Dynamic Analysis and Design Pattern Detection in Java Programs

Lei Hu
Kamran Sartipi

{hul4, sartipi}@mcmaster.ca

Department of Computing and Software
McMaster University
Canada

McMaster University
Inspiring Innovation and Discovery

SEKE' 08
July 3, 2008

1

Outline

- Motivation
- Research Problem Definition and Solution
- Proposed Framework for Feature-Oriented Design Pattern Detection
  - Feature-oriented Dynamic Analysis
  - Two-phase Design Pattern Detection Process
- Case studies on Three Versions of JHotDraw Systems
- Contribution

2

Motivation

- Software product line is a group of software-intensive systems that share a common set of features to satisfy the specific needs from the market.
  - Developed based on a reference architecture which consists of common parts and variable parts.
- An evolutionary development of a software product line starts from reverse engineering activities.
  - Understanding the existing systems
  - Locating common features to reuse
- Design pattern recovery can support the construction of software product line.
  - Understanding the existing system at design level
  - Reusing the existing system's design artefacts

3

Research Definition

- **Research Agenda**:
  - To devise a methodology and supporting tools for recovering the instances of design patterns from the implementation of software system's behavioural features, by the means of a high level pattern description method.
- **Provided Solution**:
  - We propose a reverse engineering framework which combines
    - feature-oriented dynamic analysis with
    - two-phase design pattern detection technique

    to identify the instances of design patterns for different software behavioural features.

4

Foundation of Reverse Engineering

- Reverse Engineering

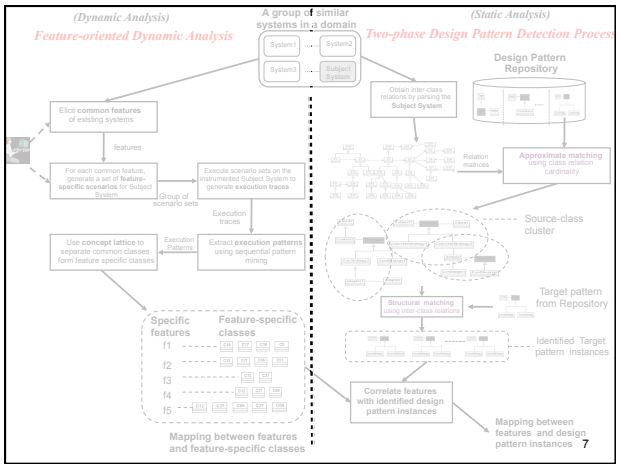
> A process of analyzing a software system to identify a system's components and their interrelationships, and create representation of the system at a higher level of abstraction. [Chikofsky&Cross]

- Two major sub-areas in Reverse Engineering
  - Static Analysis
    - Clustering
    - Visualization
    - **Pattern Matching (Design Pattern Recovery)**
  - Dynamic Analysis
    - **Feature identification**
    - Behavioural design model extraction

5

Proposed Framework for Feature-oriented Design Pattern Detection

6



Dynamic Analysis

Feature-Oriented Dynamic Analysis

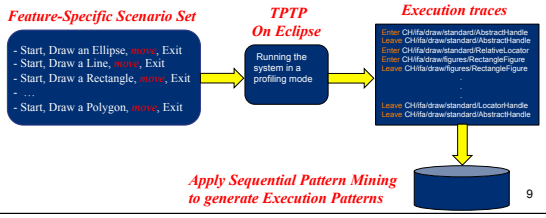
8

Feature-Oriented Dynamic Analysis

----Execution Traces Generation

Using Eclipse Test and Performance Tools Platform (TPTP) to collect the execution traces generated by running the scenarios in the feature-specific scenario set.

- Reducing execution trace size using ***filter set*** mechanism



9

Feature-Oriented Dynamic Analysis

----Execution Pattern Extraction

Execution Traces for 3 Feature-specific Scenario Sets

Feature 1	Feature 2	Feature 3
C1, C4, C3, C8, C4, C15 C1, C2, C3, C8, C16, C15 C1, C5, C3, C8, C4, C10, C18, C20 C1, C7, C3, C8, C20, C13, C15 C1, C4, C3, C8, C9, C15 C1, C3, C8, C4, C10, C17, C18, C20 C1, C3, C8, C4, C10, C18, C20	C1, C4, C23, C28, C20 C1, C2, C23, C28, C15 C1, C5, C23, C28, C4, C10, C18, C20 C1, C7, C23, C28, C20, C13, C15 C1, C4, C23, C28, C9, , C4, C10, C15 C1, C23, C28, C4, C10, C17, C18, C20	C1, C4, C33, C38, C4, C15 C1, C2, C33, C38, C16, C15 C1, C5, C33, C38, C15 C1, C7, C33, C38, C20, C13, C15 C1, C4, C33, C38, C9, C15 C1, C9, C33, C38, C10, C15

feature	1	2	3	
Execution Patterns	C1 C15 C4, C10 C18, C20 C3, C8	C1 C15 C4, C10 C18, C20 C23, C28	C1 C15 C33, C38	Common pattern Noise pattern Feature-specific pattern

10

Static Analysis

Two-Phase Design Pattern Detection

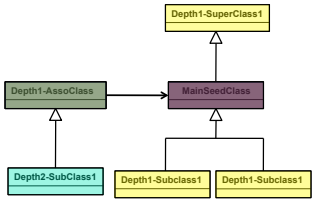
11

Describe Design Pattern using PDL

- Different types of the classes in PDL
  - Main-seed class, Depth1 class, Depth2 class and Seed-depth1 class

Example:

Class diagram of a target design pattern

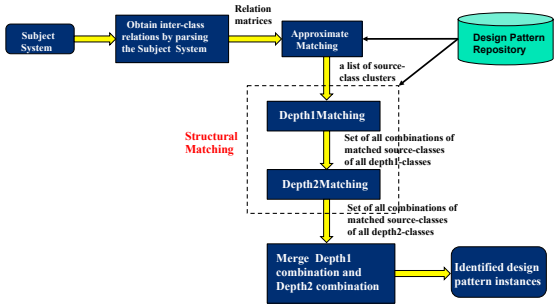


PDL Representation of a target design pattern

```
1  Begin-PDL
2    Pattern: TargetPattern
3      Main-seed class: MainSeedClass
4        Depth1:
5          Inherit_From:
6            Depth1-SuperClass1
7          Inherited_By:
8            Depth1-SubClass1;
9            Depth1-SubClass2
10         in_Association:
11           Depth1-AssoClass
12       Depth2:
13         Seed-Depth1 : Depth1-AssoClass
14           Inherited_By:
15             Depth2-SubClass1
16     End-Pattern
17 End-PDL
```

12

Two-phase Design Pattern Detection Process



13

Two-phase Design Pattern Detection

----Approximate Matching

- Attribute Vector
  - The attribute vector includes the following items:
    - Number of Inherit _From / Inherited_By relation
    - Number of in _Association / out _Association relation
    - Number of isAbstract relation (0 or 1)
- Similarity Function

Given the attribute vector $Attr\_Vec(c^{ms}) = [a_1, ..., a_k]$ of the main-seed class c^{ms} and the attribute vector $Attr\_Vec(c_i) = [b_1, ..., b_k]$ of and a source-class c_i, the approximate similarity function is defined as:

$$sim_{apx}(Attr\_Vec(c_i), Attr\_Vec(c^{ms})) = \begin{cases} \Delta(Attr\_Vec(c_i), Attr\_Vec(c^{ms})) & Attr\_Vec(c_i) \geq Attr\_Vec(c^{ms}) \\ 0 & otherwise \end{cases}$$

where $\Delta(Attr\_Vec(c_i), Attr\_Vec(c^{ms})) = 1 - \frac{\sum_{j=1}^{k}(Attr_j(c_i) - Attr_j(c_t))}{\sum_{j=1}^{k} Attr_j(c_i)}$

- Result: a group of source-class clusters

Two-phase Design Pattern Detection

----Structural Matching

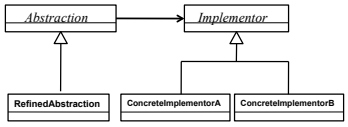
Identifying all the instances of the target design pattern within a source-class cluster.

- Depth1Matching
  - Input: a source-class cluster, a candidate main-seed class, and a target design pattern .
  - Output: set of all combinations of matched source-classes of all the depth1-classes.
- Depth2Matching
  - Input: a source-class cluster, a combination of matched source-classes of all the depth1-classes and a target design pattern .
  - Output: set of instances of the target design pattern.

15

An Example

Class diagram of Bridge Design Pattern



Attribute Vector

Attr_Vec (Implementor) = [0, 2, 1, 0, 1]

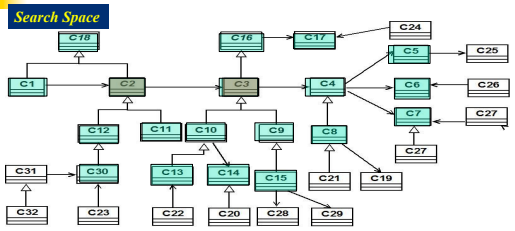
PDL representation of Bridge Design Pattern

```
1  Begin-PDL
2    Pattern: Bridge
3      Main-seed class: Implementor
4        Depth1:
5          Inherited_By:
6            ConcreteImplementorA;
7            ConcreteImplementorB
8          in_Association:
9            Abstraction
10       Depth2:
11         Seed-Depth1 : Abstraction
12           Inherited_By:
13             RefinedAbstraction
14         AbstractClass:
15           Implementor;
16           Abstraction
17   End-Pattern
18 End-PDL
```

16

An Example…

Search Space



Through applying approximate matching on the search space, we obtain two candidates of main-seed class C2 and C3.

Attr_Vec(C2)=[1, 2, 1, 1, 1]

Attr_Vec(C3)=[1, 2, 1, 1, 1]

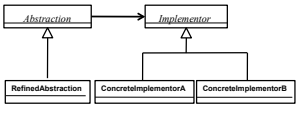
Attr_Vec (Implementor) = [0, 2, 1, 0, 1]

17

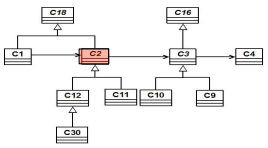
An Example…

- Depth1Matching

Class diagram of Bridge Design Pattern



A source-class cluster



Implementor (main-seed class)	Abstraction	Concrete-ImplementorA	Concrete-ImplementorB
c_2	c_1	c_{11}	c_{12}
c_2	c_1	c_{12}	c_{11}

18

Experiments with JHotDraw System

- Statistics of three versions of JHotDraw systems

Systems	Version	# Classes	#Files	#LOC
JHotDraw	5.1	172	144	8419
JHotDraw	6.0b1	405	289	21091
JHotDraw	7.0.7	331	309	32122

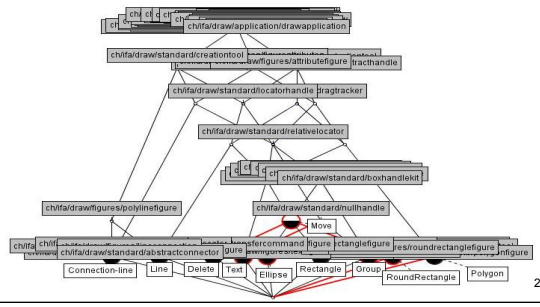
- The experimental results of execution pattern extraction

Specific Feature of JHotDraw	Number of Scenarios	Average Trace Size	Average Pruned Trace Size	Number of Extracted Patterns	Average Pattern Size
Rectangle	4 / 4 / 4	2494 / 4889 / 11962	927 / 2165 / 2110	13 / 24 / 23	126 / 170 / 220
Round Rectangle	4 / 4 / 4	2369 / 5040 / 10620	927 / 2327 / 1864	15 / 25 / 19	153 / 138 / 183
Ellipse	4 / 4 / 4	2104 / 5492 / 10580	773 / 2226 / 1915	15 / 22 / 24	112 / 185 / 175
Polygon	4 / 4 / 4	4553 / 15769 / 17130	1654 / 4029 / 3142	21 / 41 / 38	199 / 192 / 130
Line	4 / 4 / 4	1439 / 4253 / 9882	546 / 2224 / 2123	7 / 24 / 27	157 / 170 / 126
Move	4 / 4 / 4	2599 / 4930 / 11341	774 / 2688 / 2487	18 / 34 / 52	31 / 89 / 37
Delete	4 / 4 / 4	1323 / 5739 / 8540	623 / 2456 / 969	16 / 32 / 24	36 / 89 / 49
Group	5 / 5 / 5	4579 / 12978 / 33921	1397 / 4675 / 4842	36 / 66 / 57	26 / 85 / 49
LineConnection	4 / 4 / 4	5238 / 10356 / 24075	1681 / 4158 / 4437	38 / 53 / 56	36 / 78 / 73
Text	4 / 4 / 4	1524 / 6074 / 18629	781 / 2435 / 2204	11 / 35 / 12	62 / 105 / 288

Legend : A / B / C A: data for JHotDraw 5.1 B: data for JHotDraw 6.0b1 C: data for JHotDraw 7.0.7

Experiments with JHotDraw System

Concept lattice representation of features and classes in JHotDraw 5.1



Experiments with JHotDraw System …

Results of feature-specific classes assignment for 10 features of JHotDraw 5.1

Specific Features	Feature-specific Classes
Draw a Rectangle	RectangleFigure
Draw a RoundRectangle	RoundRectangleFigure, RadiusHandle
Draw an Ellipse	EllipseFigure
Draw a Polygon	PolygonHandle, PolygonTool, PolygonFigure, PolygonScaleHandle, SelectAreaTracker
Draw a Text	TextTool, FontSizeHandle, FloatingTextField, TextFigure
Group Figures	GroupHandle, GroupFigure, GroupCommand
Move a Figure	SouthEastHandle, SouthHandle, NorthHandle, EastHandle, WestHandle, SouthWestHandle, NorthEastHandle, NorthWestHandle, RelativeLocator, BoxHandleKit
Delete a Figure	FigureTransferCommand, DeleteCommand
Draw a Line	PolylineLocator, PolylineFigure, LineFigure, PolylineHandle
Draw a LineConnection	Lineconnection, AbstractConnector, ConnectionTool, ArrowTip

Experiments with JHotDraw …

Results of identified Adapter design pattern instances and related features in JHotDraw 5.1 system

Design Pattern Instance	Related Feature
CH/ifa/draw/figures/PolyLineFigure (Target) CH/ifa/draw/figures/LineConnection (Adapter) CH/ifa/draw/framework/Connector (Adaptee)	Draw a Line
CH/ifa/draw/standard/AbstractHandle (Target) CH/ifa/draw/figures/RadiusHandle (Adapter) CH/ifa/draw/figures/RoundRectangleFigure (Adaptee)	Draw a RoundRectangleFigure
CH/ifa/draw/standard/AbstractHandle (Target) CH/ifa/draw/contrib/PolygonHandle (Adapter) CH/ifa/draw/framework/Locator (Adaptee)	Draw a Polygon
CH/ifa/draw/standard/AbstractTool (Target) CH/ifa/draw/contrib/PolygonTool (Adapter) CH/ifa/draw/contrib/PolygonFigure (Adaptee)	Draw a Polygon
CH/ifa/draw/util/Command (Target) CH/ifa/draw/figures/GroupCommand (Adapter) CH/ifa/draw/framework/DrawingView (Adaptee)	Group figures
CH/ifa/draw/framework/Connector (Target) CH/ifa/draw/standard/AbstractConnector (Adapter) CH/ifa/draw/framework/Figure (Adaptee)	Draw a LineConnection
CH/ifa/draw/framework/ConnectionFigure (Target) CH/ifa/draw/figures/LineConnection (Adapter) CH/ifa/draw/framework/Connector (Adaptee)	Draw a LineConnection
CH/ifa/draw/figures/PolyLineFigure (Target) CH/ifa/draw/figures/LineConnection (Adapter) CH/ifa/draw/framework/Connector (Adaptee)	Draw a LineConnection

Summary

We presented:

- A methodology to identify individual design pattern instances from the implementation of system behavioural features.
- A new design pattern representation, PDL (Pattern Description Language), which enables users to describe the structural information of design patterns efficiently and conveniently.
- A two-phase design pattern process (approximate matching & structure matching) to reduce the complexity of the matching process
- A prototype toolkit for the proposed approach on the Eclipse open platform.

Future Work

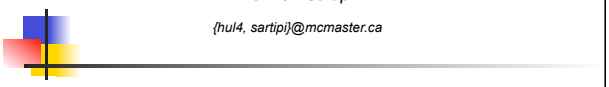
Our future work will mainly concentrate on the following directions:

- Extending the pattern repository to support more design patterns identification.
- Extracting more inter-class relations, such as delegation and method invocation, to improve the accuracy of the technique.
- Validating our approach on large-scale software systems.
- Tracking the evolution of software systems at design level by analyzing the evolution of design patterns.

Dynamic Analysis and Design Pattern Detection in Java Programs

Lei Hu
Kamran Sartipi

{hul4, sartipi}@mcmaster.ca



Department of Computing and Software
McMaster University
Canada

McMaster University
Inspiring Innovation and Discovery

SEKE' 08
July 3, 2008

25