Implementation and Use of Data Structures in Java Programs

Syed S. Albiz and Patrick Lam

September 2009

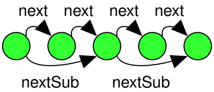
Dark Matter

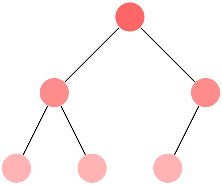
NGC 1300, NASA, ESA, and The Hubble Heritage Team (STScI/AURA)

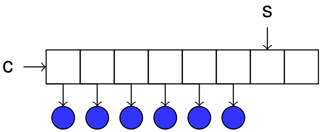
What makes up 90% of matter in galaxies?

Data Structures

Data structures are key to Computer Science.







Libraries

JDK Collections are extensive.
Do developers implement data structures?

Alternative to Libraries

Or do they roll their own?

Goal

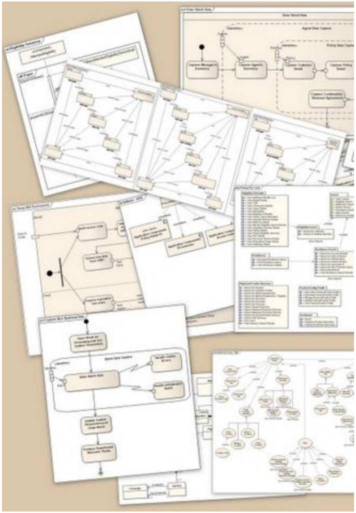
Empirically investigate data structure implementation and use in Java programs.

- How do programs organize information on the heap?

Motivation
(Students sleeping through the bell, http://www.flickr.com/photos/portablematthew, CC-BY-NC3)

- Program Understanding
- Shape Analysis
- Parallelization
- Library Sufficiency

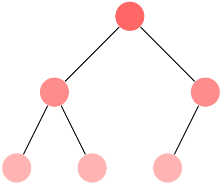
Motivation: Program Understanding
(“UML_Diagrams”, kishorekumar62, Wikimedia Commons, GFDL/CC-BY-SA3)

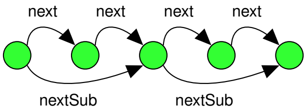


Goal: match the code to the model.

```
public NodeVisitor enter(Node parent, Node n) {
 if (n instanceof LocalDecl || n instanceof Formal) {
  List<LocalInstance> li = decls.get(parent);
  if (li == null) {
   li = new LinkedList();
   decls.put(parent, li);
  }
  if (n instanceof LocalDecl)
   li.add(((LocalDecl)n).localInstance());
  else
   li.add(((Formal)n).localInstance());
 }
```

Motivation: Shape Analysis





Shape analysis is expensive and must be quarantined.

Motivation: Parallelization

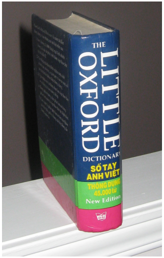
Parallelization motivated much shape analysis research.

Motivation: Library Sufficiency
(20060513_toolbox, Per Erik Strandberg, Wikimedia Commons, CC-BY-SA2.5)

Do libraries contain enough tools or do developers often need to supplement library implementations?

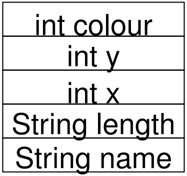
Outline

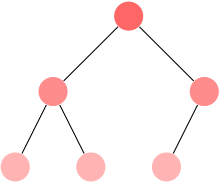
Defining Data Structures

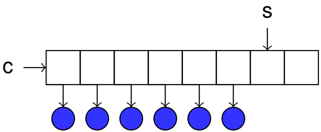


What are we looking for, exactly?

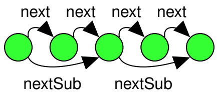
One of these is Not Like the Others





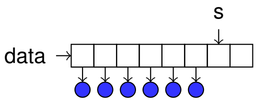


Example: Linked Data Structures




```
class Node {
  Node next, nextSub;
  // etc.
}
```

Example: Arrays




```
class A {
  Object[] data;
  int s;
}
```

Working Definition

Intuitively: a data structure is a mutable set which can contain arbitrarily many elements.

Corner Cases

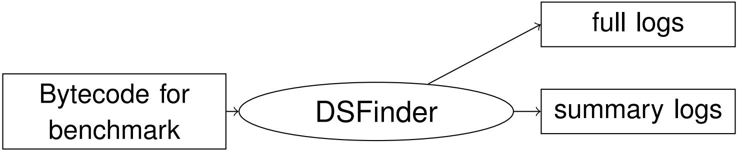
We consider data structure *implementations*, not *interfaces*. Hence:

- Singleton sets: not data structures.
- Fixed-universe sets: data structures.

Outline

Our Approach

Implemented the `DSFinder` tool for detecting data structure implementations in Java programs.



`http://www.patricklam.ca/dsfinder`

java.util.LinkedList from OpenJDK-7

```
public class LinkedList<E>
    extends AbstractSequentialList<E>
    implements List<E>, Deque<E>, Cloneable, Serializable
{
  private transient Entry<E> header =
        new Entry<E>(null, null, null); // etc

  private static class Entry<E> {
    E element;
    Entry<E> next;
    Entry<E> previous; // etc
```

java.util.LinkedList from OpenJDK-7

```
public class LinkedList<E>
    extends AbstractSequentialList<E>
    implements List<E>, Deque<E>, Cloneable, Serializable
{
  private transient Entry<E> header =
        new Entry<E>(null, null, null); // etc

  private static class Entry<E> {
    E element;
    Entry<E> next;
    Entry<E> previous; // etc
```

`Entry<E>` is an exact recursive type definition.

java.util.LinkedList from OpenJDK-7

```
public class LinkedList<E>
    extends AbstractSequentialList<E>
    implements List<E>, Deque<E>, Cloneable, Serializable
{
  private transient Entry<E> header =
        new Entry<E>(null, null, null); // etc

  private static class Entry<E> {
    E element;
    Entry<E> next;
    Entry<E> previous; // etc
```

Fields `next`, `previous` are whitelisted as linked list fields.

java.util.LinkedList from OpenJDK-7

```
public class LinkedList<E>
    extends AbstractSequentialList<E>
    implements List<E>, Deque<E>, Cloneable, Serializable
{
  private transient Entry<E> header =
        new Entry<E>(null, null, null); // etc

  private static class Entry<E> {
    E element;
    Entry<E> next;
    Entry<E> previous; // etc
```

We count linked list implementations, e.g. `Entry`, and linked list uses, e.g. `LinkedList`.

Apache Tomcat

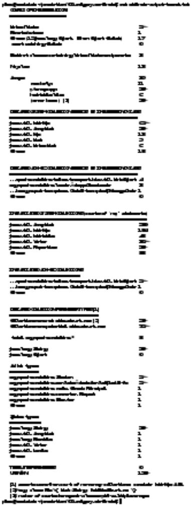
Apache Tomcat

We present `DSFinder` results from Apache Tomcat 6.0.18.

> *Apache Tomcat is an open source software implementation of the Java Servlet and JavaServer Pages technologies.*

`http://tomcat.apache.org`

DSFinder Output



Summary logs include:

- Linked Data Structure Counts
- Array Data Structure Counts
- Uses of System Collections
- Uses of Ad-Hoc Collections
- Parametric Type Information

Full logs list all potential data structures.

Data Structure Implementation Counts

```
COUNTS OF IMPLEMENTATIONS
=========================

Linked lists                                    2
Parents/outers                                  1
Others (12 Object, 5 non-Object fields)        17
 exact matching fields                          0

Distinct classes w/linked lists and parents:    3

N-cycles                                       13

Arrays                                         39
         read-only:                            11
         w/arraycopy:                          25
         hashtable-like:                        6
         (error bars:) [3]                     20
```

`DSFinder` counts recursive type definitions.

- Linked Lists: `next`, `prev`;
- Parents: `parent`, `outer`;
- Other recursive types.

Type-based Classification

```
class C
{
  C next;             // Exact recursive type definition
}

class D extends C
{
  C prev;             // Non-exact recursive type definition
  String name;        // Not a recursive type definition
  Object value;       // What should this be?
}
```

Name-based Classification
(Santa_Claus-SL, Shawn Lea, Wikimedia Commons, CC-BY2.0)

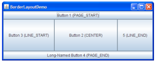
Blacklists and Whitelists

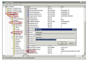
Field names show developer intent.
To automatically identify data structures, we:

1. Blacklist common false positives.
2. Identify linked lists and trees.
3. Identify other data structures.

Type and Name-based Blacklists

We blacklist the following field types:

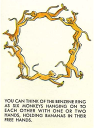
-  subclasses of `Throwable`.
- 

subclasses of AWT or Swing.
-  subclasses of `Properties`.

We also blacklist the following field names:

- name contains `lock` or `key`.
- name contains `value`, `arg`, `data`, `dir`, `param` or `target`.

N-Cycles
(http://www.coronene.com/blog/?p=276)



Classes could conceivably collaborate to create data structures:

```
public abstract class Object3D {
  protected MaterialMapping matMapping; // etc

public abstract class MaterialMapping {
  Object3D object; Material material; // etc
```

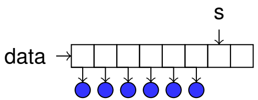
In this case, the undocumented invariant

$$\forall x.\texttt{x.matMapping.object = x}$$

prevents data structures.

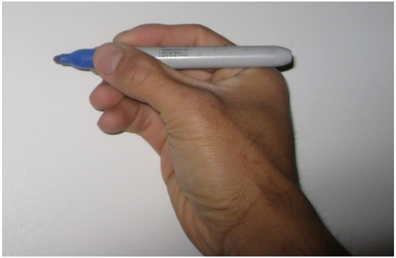
Arrays

Can also store unbounded amounts of data.



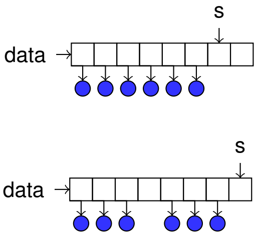
We therefore count the number of arrays in our applications.

Read-only Arrays



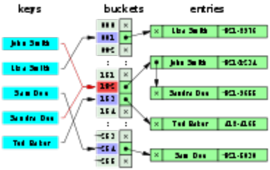
Arrays without writes aren't going to be mutable, hence not data structures.

Arraycopy



`System.arraycopy` denotes a likely array-based (list-like) data structure.

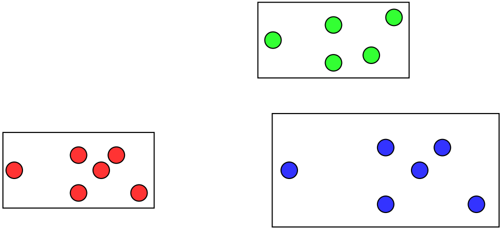
Hash tables
(Hash_table_5_0_1_1_1_1_1_LL.svg, Jorge Stolfi, Wikimedia Commons, CC-BY-SA3.0)




```
private Item get (final Item key) {
  Item tab[] = table;
  int hashCode = key.hashCode;
  int index = (hashCode & 0x7FFFFFFF) % tab.length;
  for (Item i = tab[index]; i != null; i = i.next) {
    if (i.hashCode == hashCode && key.isEqualTo(i)) {
      return i; } }
  return null; }
```

We identify uses of the % and `hashCode()` as potential data structures.

Error Bars



Our measurements conflate accesses to all arrays in a class. Error bars constrain the inaccuracy by reporting the maximum number of arrays per class.

Data Structure Implementation Counts

```
COUNTS OF IMPLEMENTATIONS
=========================

Linked lists                                        2
Parents/outers                                      1
Others (12 Object, 5 non-Object fields)            17
 exact matching fields                              0

Distinct classes w/linked lists and parents:        3

N-cycles                                           13

Arrays                                             39
         read-only:                                11
         w/arraycopy:                              25
         hashtable-like:                            6
         (error bars:) [3]                         20
```

`DSFinder` counts recursive type definitions.

- Linked Lists: `next`, `prev`;
- Parents: `parent`, `outer`;
- Other recursive types.

(Calgary parking sign, greatergreaterwashington.org)

We also count data structure usage, both system and ad-hoc.

Declared versus Instantiated Data Structures

Declared:

```
class C {
  List l;
}
```

Instantiated:

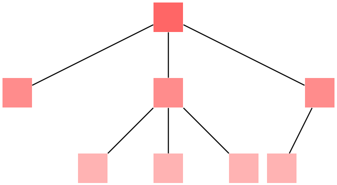
```
class D {
  public void foo() {
    LinkedList l = new LinkedList();
  }
}
```

Recall that we count data structure containers (`LinkedList`), not nodes (`LinkedList$Entry`).

Tomcat Data Structure Usage Report

```
DECLARED SYSTEM COLLECTION FIELDS, BY IMPLEMENTING CLASS
========================================================
java.util.HashMap                                        62
java.util.ArrayList                                      20
Others                                                   46


DECLARED AD-HOC COLLECTION FIELDS, BY IMPLEMENTING CLASS
========================================================
...apache.catalina.tribes.transport.bio.util.LinkObject 4
org.apache.catalina.loader.WebappClassLoader            3
...bes.group.interceptors.OrderInterceptor$MessageOrder 1
Others                                                  0


INSTANTIATED SYSTEM COLLECTIONS (counts of 'new' statements)
============================================================
java.util.ArrayList                                      230
java.util.HashMap                                        184
java.util.Hashtable                                      48
Others                                                   148

INSTANTIATED AD-HOC COLLECTIONS
===============================
...apache.catalina.tribes.transport.bio.util.LinkObject 2
...bes.group.interceptors.OrderInterceptor$MessageOrder 1
Others                                                  0
```

Building Composite Data Structures



This tree calls out for a list of children at each node.

Building Composite Data Structures

DairuggerPromo.jpg, Wikipedia / 365.080827, s4ints@Flickr, CC-BY-NC-SA

(i.e. List<Node>)

and not

(i.e. List<String>)

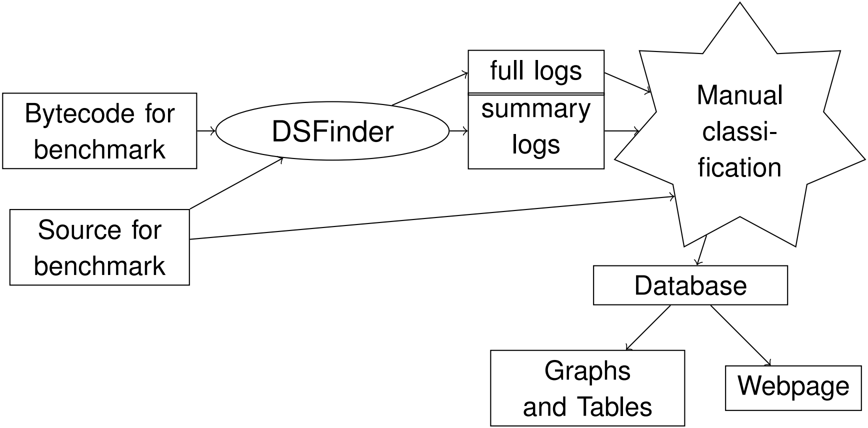
Composite Data Structures Report

```
DECLARED COLLECTION PARAMETER TYPES [1]
=======================================
Collections are not data structures [2]                  23
Collections are potential data structures                72

 total org.apache.catalina.*                             8
 java.lang.String                                        20
 java.lang.Object                                        0

Ad-Hoc types:
=============
org.apache.catalina.Session                              2
org.apache.catalina.servlets.WebdavServlet$LockInfo      2
Others                                                   4

System types:
==============
java.lang.String                                         20
java.util.ArrayList                                      1
Others                                                   4

TEMPLATE PARAMETERS                                      0
UNKNOWN                                                  128
```

In `tomcat`, only 1 of the 72 potential data structures is actually a data structure.

Outline

Full workflow



Provenance of the Results
http://www.townlinehatchery.com/33PICT0004.JPG

Manual classification of DSFinder results.

Summary of Results I

Benchmark	Ver	Classes	Graphs	Lists	Trees	DS	Declarations SYS	Declarations AH	Instantiations SYS	Instantiations AH
aglets	2.0.2	413	0	3	0	3	59	11	135	18
antlr-gunit (l)	3.1.3	147	0	0	0	0	2	0	20	0
aoi	2.7.2	680	0	11	5	16	26	81	247	209
argoUML	0.28	2068	0	2	9	11	87	34	1118	29
asm (l)	3.2	176	0	10	0	10	49	7	92	0
azureus	r1.97	5651	0	3	8	11	645	27	2077	15
bloat (l)	1.0	332	0	7	16	23	128	23	361	4
cglib (l)	2.2	226	0	0	0	0	11	0	58	0
colt (l)	1.2.0	554	0	0	0	0	0	0	9	0
columba	1.4	1850	0	0	4	3	101	24	429	129
derby	10.5.1.1	1812	0	8	14	22	282	45	585	594
drjava	r4932	3155	0	5	15	19	107	31	951	70
fop	0.95	1314	0	4	9	12	302	45	911	41
ireport	3.0.0	2451	0	0	3	3	179	20	490	17
jchem	1.0	914	0	3	0	3	212	9	567	13
jcm	r133	353	0	0	2	2	10	1	175	1
jedit	4.3pre16	1109	0	15	4	19	71	188	370	97
jfreechart (l)	1.0.13	819	0	0	3	3	144	8	326	2
jre	1.5.0_18	712	0	16	9	25	92	116	284	251
junit	4.7	110	0	0	0	0	13	0	33	0
jython	2.2.1	953	0	4	3	7	66	12	284	154
lucene (l)	2.4.1	795	0	15	0	15	155	48	412	21
megamek	0.34.2	1799	0	0	0	0	23	0	978	0
poi (l)	3.2	1059	0	1	2	3	85	2	215	0
sandmark	3.4.0	1087	4	10	22	36	272	27	996	18
tomcat	6.0.18	656	0	3	5	8	128	11	305	3

Summary of Results I

Benchmark	Ver	Classes	Graphs	Lists	Trees	DS	Declarations SYS	Declarations AH	Instantiations SYS	Instantiations AH
aglets	2.0.2	413	0	3	0	3	59	11	135	18
antlr-gunit (l)	3.1.3	147	0	0	0	0	2	0	20	0
aoi	2.7.2	680	0	11	5	16	26	81	247	209
argoUML	0.28	2068	0	2	9	11	87	34	1118	29
asm (l)	3.2	176	0	10	0	10	49	7	92	0
azureus	r1.97	5651	0	3	8	11	645	27	2077	15
bloat (l)	1.0	332	0	7	16	23	128	23	361	4
cglib (l)	2.2	226	0	0	0	0	11	0	58	0
colt (l)	1.2.0	554	0	0	0	0	0	0	9	0
columba	1.4	1850	0	0	4	3	101	24	429	129
derby	10.5.1.1	1812	0	8	14	22	282	45	585	594
drjava	r4932	3155	0	5	15	19	107	31	951	70
fop	0.95	1314	0	4	9	12	302	45	911	41
ireport	3.0.0	2451	0	0	3	3	179	20	490	17
jchem	1.0	914	0	3	0	3	212	9	567	13
jcm	r133	353	0	0	2	2	10	1	175	1
jedit	4.3pre16	1109	0	15	4	19	71	188	370	97
jfreechart (l)	1.0.13	819	0	0	3	3	144	8	326	2
jre	1.5.0_18	712	0	16	9	25	92	116	284	251
junit	4.7	110	0	0	0	0	13	0	33	0
jython	2.2.1	953	0	4	3	7	66	12	284	154
lucene (l)	2.4.1	795	0	15	0	15	155	48	412	21
megamek	0.34.2	1799	0	0	0	0	23	0	978	0
poi (l)	3.2	1059	0	1	2	3	85	2	215	0
sandmark	3.4.0	1087	4	10	22	36	272	27	996	18
tomcat	6.0.18	656	0	3	5	8	128	11	305	3

Summary of Results I

Benchmark	Ver	Classes	Graphs	Lists	Trees	DS	Declarations SYS	Declarations AH	Instantiations SYS	Instantiations AH
aglets	2.0.2	413	0	3	0	3	59	11	135	18
antlr-gunit (l)	3.1.3	147	0	0	0	0	2	0	20	0
aoi	2.7.2	680	0	11	5	16	26	81	247	209
argoUML	0.28	2068	0	2	9	11	87	34	1118	29
asm (l)	3.2	176	0	10	0	10	49	7	92	0
azureus	r1.97	5651	0	3	8	11	645	27	2077	15
bloat (l)	1.0	332	0	7	16	23	128	23	361	4
cglib (l)	2.2	226	0	0	0	0	11	0	58	0
colt (l)	1.2.0	554	0	0	0	0	0	0	9	0
columba	1.4	1850	0	0	4	3	101	24	429	129
derby	10.5.1.1	1812	0	8	14	22	282	45	585	594
drjava	r4932	3155	0	5	15	19	107	31	951	70
fop	0.95	1314	0	4	9	12	302	45	911	41
ireport	3.0.0	2451	0	0	3	3	179	20	490	17
jchem	1.0	914	0	3	0	3	212	9	567	13
jcm	r133	353	0	0	2	2	10	1	175	1
jedit	4.3pre16	1109	0	15	4	19	71	188	370	97
jfreechart (l)	1.0.13	819	0	0	3	3	144	8	326	2
jre	1.5.0_18	712	0	16	9	25	92	116	284	251
junit	4.7	110	0	0	0	0	13	0	33	0
jython	2.2.1	953	0	4	3	7	66	12	284	154
lucene (l)	2.4.1	795	0	15	0	15	155	48	412	21
megamek	0.34.2	1799	0	0	0	0	23	0	978	0
poi (l)	3.2	1059	0	1	2	3	85	2	215	0
sandmark	3.4.0	1087	4	10	22	36	272	27	996	18
tomcat	6.0.18	656	0	3	5	8	128	11	305	3

Summary of Results II

Benchmark	Arrays				
	ARR	RO	w/AC	HS	ERR
aglets	10	6	1	0	2
antlr-gunit (l)	0	0	0	0	0
aoi	103	9	20	36	33
argoUML	19	3	1	1	2
asm (l)	10	1	7	5	4
azureus	169	53	41	41	47
bloat (l)	22	2	7	5	4
cglib (l)	16	17	4	0	10
colt (l)	12	0	4	3	0
columba	59	26	1	4	4
derby	151	74	39	13	74
drjava	50	36	0	15	17
fop	30	5	5	4	2
ireport	33	6	0	2	0
jchem	43	3	11	17	8
jcm	18	1	0	8	7
jedit	44	17	7	3	0
jfreechart (l)	34	7	12	14	19
jre	33	10	10	9	6
junit	1	0	0	0	0
jython	71	47	18	25	36
lucene (l)	62	19	9	8	2
megamek	82	11	8	15	8
poi (l)	41	22	8	7	15
sandmark	86	49	10	20	34
tomcat	36	8	25	6	20

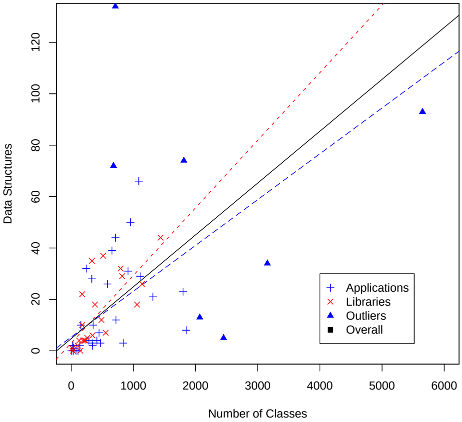
Some Random Observations: Data Structures

- Max number of lists: 16 (jre)
- Max number of data structures: 36 (sandmark)
- # of declared system collections $>$ # of ad-hoc collections
  - exception: `jedit`
- # of instantiated system collections $\gg$ # of ad-hoc collections

Some Random Observations: Arrays

- Numerically-intensive benchmarks declare more arrays (e.g. `jcm`, `artofillusion`).
- Lots of read-only arrays (up to half).
- Many apparent data structures.
- Reasonably low error bars.

Correlations



Usage of Ad-Hoc Data Structures

When did developers implement their own data structures?

- when they only use limited functionality (add, iterate); or
- when they are implementing chaining for hash tables; or
- when they need more features (dual-function hash tables)

Clearly, developers use system data structures extensively.

Outline

Hypotheses

We explored the four following hypotheses:

1. Number of data structure implementations correlates with number of classes.
2. Libraries implement more data structures than applications, per capita.
3. Developers use system data structures more often than ad-hoc data structures.
4. Data structure implementations are concentrated in a small portion of applications and libraries.

Hypothesis 1: Correlation

The number of data structure implementations in a program correlates with its number of classes.

On our benchmark set, the Pearson correlation coefficient is 0.58, which provides some support for our hypothesis.

Hypothesis 2: Libraries v. Applications

Libraries implement more data structures than applications on a per-class basis.

Our libraries implemented approximately 0.026 data structures per class (correlation 0.74), while our applications implemented approximately 0.18 data structures per class (correlation 0.52).

Hypothesis 3: System v. Ad-Hoc

Developers use system data structures more often than ad-hoc data structures.

YES!

Both instantiation and declaration data support this hypothesis.

- All but 3 benchmarks declare more system data structures than ad-hoc data structures, often by a factor of 2×.
- All benchmarks instantiate system data structures more often than ad-hoc data structures.

Hypothesis 4: Data Structure Confinement

Data structure implementations are concentrated in a small portion of applications and libraries.

YES!

No benchmark declares more than 24 data structures. Data structure manipulation is also confined to a limited number of classes.

Outline

Confounding Factors
(NYC Street Cleaning, Salim Virji, CC-BY-SA2.0)

Confounding Factors

- Application domain;
- Number and type of libraries used;
- Developer characteristics.

Our benchmarks come from many domains with heterogeneous developer pools.

External Validity

Is our corpus representative?

- All programs are open-source Java programs.

Our corpus includes 62 programs with up to 5000 classes.

Melton and Tempero report on Java program characteristics; proprietary applications are generally similar to open-source applications.

Outline

Abstract Data Types

VHLL74 Liskov and Zilles invented data abstraction, in the form of operation clusters.

OOPSLA86 Snyder describes how OO programs encapsulate clusters using classes.

POPL03 Ownership types (e.g. Boyapati) can statically guarantee encapsulation.

Empirical Studies

SPE07 Collberg et al. study empirical properties of Java programs: most commonly used classes, field types, bytecode sequences.

OOPSLA06 Baxter et al. study whether various metrics fit power-law distribution for Java programs.

ASEC09 Tempero investigates field visibility and access: fields usually encapsulated.

ESE07 Tempero and Melton count cyclic dependencies.

OOPSLA03 Dufour et al. investigate dynamic metrics for Java programs, including data structure uses.

Shape Analysis

Shape analysis statically identifies data structures and verifies data structure manipulations.

PALE, TVLA verify list and tree insertion, concatenation, sorting, reversal, removal, etc.

Problem: scalability!

Separation Logic enables local reasoning about data structure implementations;

Hob, Jahob enable developers to modularize their code and provide effective and verified interfaces between modules.

Outline

Conclusion

- Defined notion of a data structure.
- Described `DSFinder` tool to count data structures.
- Presented empirical results on 62 open-source Java applications.
- Found few data structure implementations (< 24).
  - Correlates with program size; 0.020 data structures / class.

Data structure implementations don't constitute the dark matter behind Java programs. What does?

http://www.patricklam.ca/dsfinder