The Social Network of Java Classes

Diego Puppin, Fabrizio Silvestri

Institute of Information Science and Technologies
Pisa, Italy

SAC 2006, Dijon

Outline

Social Networks

- Behavior emerging from linked entities.
- Ranging from biology, to computer science, from economics to epidemiology.
- First studies in late 1930s.
- Then, studies on random graph by Pál Erdős.

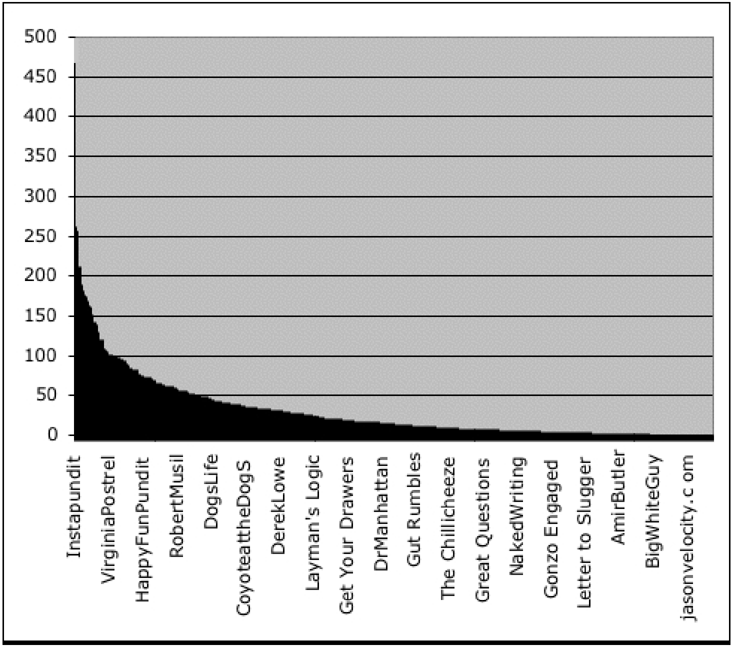
Emerging Behavior

- *Rich* Web pages getting *richer*
- Stocks raising or falling quickly
- Ants building nests

Emerging Behavior

- *Rich* Web pages getting *richer*
- Stocks raising or falling quickly
- Ants building nests
- Traffic jams

When counting links, visitors, references in a natural network, large events are rare, but small ones are quite common.
Next picture by Clay Shirky: number of connections to popular blogs.


500
450
400
350
300
250
200
150
100
50
0
Instapundit
VirginiaPostrel
HappyFunPundit
RobertMusil
DogsLife
CoyoteattheDogS
DerekLowe
Layman's Logic
Get Your Drawers
DrManhattan
Gut Rumbles
The Chillicheeze
Great Questions
NakedWriting
Gonzo Engaged
Letter to Slugger
AmirButler
BigWhiteGuy
jasonvelocity.c om

Zipf, Pareto, Power Law

- Zipf's law: the 'size' y of an occurrence of an event is inversely proportional to its 'rank': $y \propto r^{-b}$, with b close to unity.
- Pareto: the number of events larger than x is an inverse power of x: $P[X > x] \propto x^{-k}$.
- Power law: the number of events equal to x: $P[X = x] \propto x^{-(k+1)} = x^{-a}$.

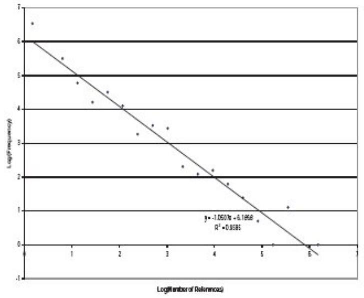
Also called scale-free distribution, because the distribution doesn't change with scale.

Scale-free Networks from Design Choices

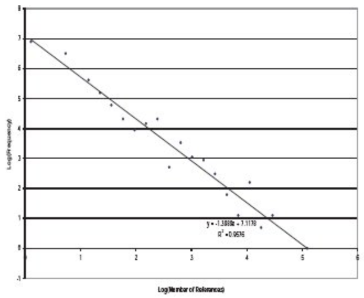
- In “Scale-free Networks from Optimal Design”, analysis of class graphs of JDK and a computer game.
- Power-laws appear in the number of class links.
- Coherent SW engineering choices (modularity, orthogonality, interface contracts) lead to a scale-free network with power-law distributions.
- Internal property of individual projects.

More Power-laws in Big OO Programs

- In “Power Law Distribution in Class Relationships”, power-laws found in: number of sub-classes, number of fields, number of methods, number of constructors.
- Class graphs built out of interfaces, inheritance, aggregation, return type and parameter type.
- Verified for JDK, Ant and Tomcat.
- Important impact on software engineering, code maintenance, garbage collection.



(a) Field members

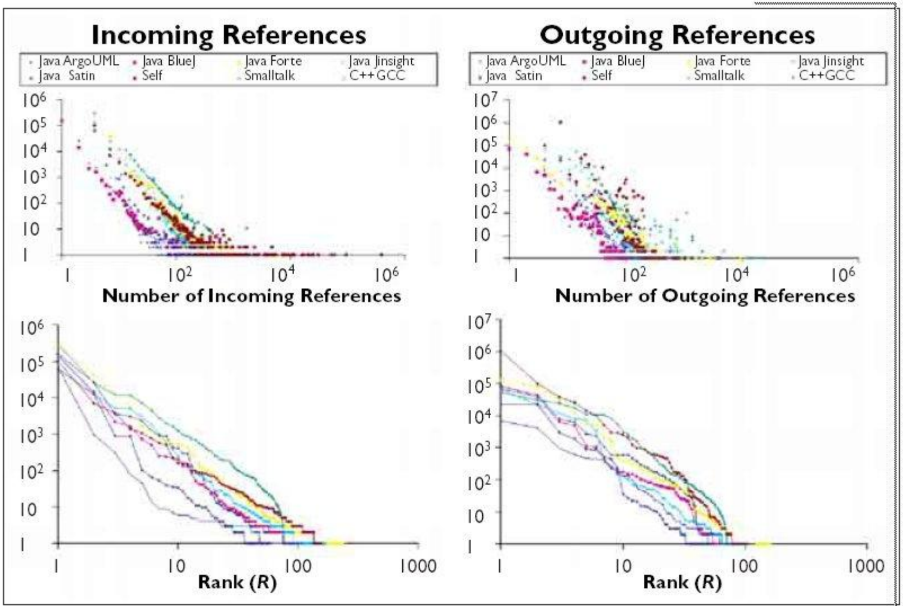


(b) Containing classes

Figure 7. Log-Log plots showing power law distributions in (a) the number of classes referenced as field variables and (b) in the number of classes which contain references to classes as field variables.

Scale Free Geometry in OO Programs

- Authors get a snapshot of the memory at run time.
- They counts the number of references (link) among objects.
- Number of inlinks and outlinks are distributed with power law.
- No typical size of an object.
- Few objects are very important.
- Not *classes*, but *objects*.


Incoming References
Java ArgoUML
Java BlueJ
Java Forte
Java Jinsight
Java Satin
Self
Smalltalk
C++GCC
Number of Incoming References
Outgoing References
Number of Outgoing References
Rank (R)
Rank (R)

Power-Law from Social Behavior

- Is this just an emergent effect of engineering choices?
- Does this happens also *naturally*, in the ecosystem of OO programs and libraries?
- Is this true at run-time, source-code level, interface level?
- Can this be useful for developers?

$\longrightarrow$ We want to study the behavior that is *naturally emerging* from the social network of programs and libraries, at the level of *static* references.

Outline

Goal

- We would like to create a tool for developers, to help them choose among software libraries
- We would like to mine code to extract information about code quality and reputation
- We would like to let a developer query and get the best library available
- We would like to support him/her with documentation and code examples

An Idea to Rank Software

- Let's have a look at Web pages
  - link $\rightarrow$ PageRank
- Links among classes:
  - Parameter type, return type, interfaces, inheritance, class fields
- Static analysis of interfaces is enough

Why only interfaces?

- Commercial component will hide source code
- Will also hide runtime profile information
- Interfaces must be public, if you want to share your code

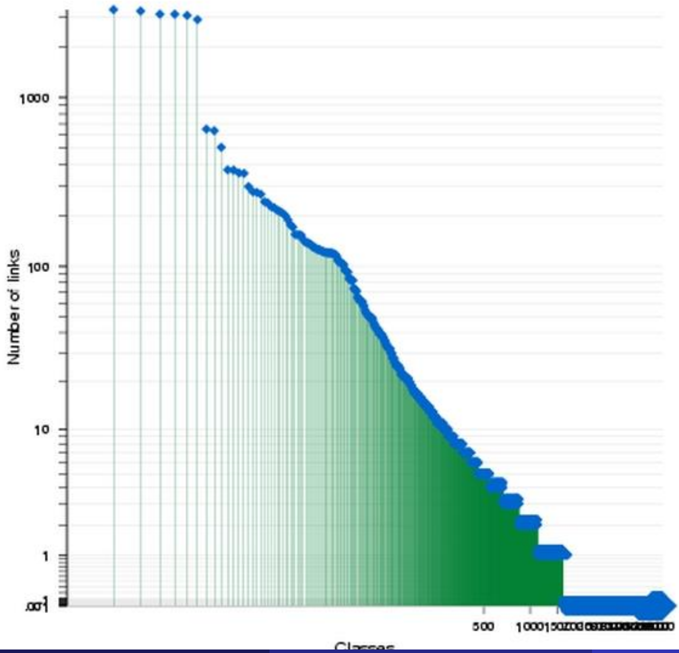
Social Network in Java Classes

Will this work?

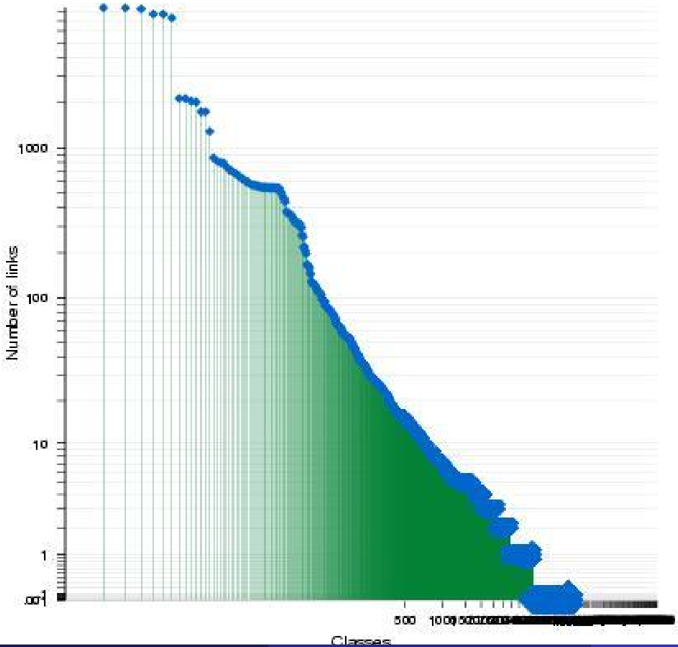
- Java programmer $\leftrightarrow$ Web user
- Class links $\leftrightarrow$ Web links
- Java class search $\leftrightarrow$ Web Search

Initial Experiments

- Java classes, simple composition model
- Strong documentation (Java Docs)
- Unique ID (Package names)
- Class links are very easy to see
- We collected 49000 classes
- We parsed and built a *social graph*



Number of links
1000
100
10
1
.001
500
1000
Classes


Number of links
1000
100
10
1
.001
500
1000
Classes

Power-laws Everywhere

- We observed a power-law distribution of inlinks.
- ...like the Web.
- This is a natural emerging property, from independent projects.
- Independent developers shape a scale-free ecosystem.

Class Rank

To determine the rank of a class C, we iterate the following formula:

$$rank_C = \lambda + (1 - \lambda) \sum_{i \in inlinks_C} \frac{rank_i}{\#outlinks_i}$$

where $inlinks_C$ is the set of classes that use C (with a link into C), $\#outlinks_i$ is the number of classes used by i (number of links out of i), and λ a small factor, usually around 0.15.

Top-ranking classes

- String, Object, Class, Exception
- #7: Apache MessageResources
- #11: Tomcat CharChunk
- #14: DBXML Value
- #73: JXTA ID

Ranking better than Pure TF.IDF

TF.IDF for “file writer”:

1. javax.jnlp.JNLPRandomAccessFile, from JNLP API Reference 1.5;
2. javax.swing.filechooser.FileSystemView, from Java 2 Platform SE 5.0;
3. java.io.FileOutputStream, from Java 2 Platform SE 5.0;
4. java.io.RandomAccessFile, from Java 2 Platform SE 5.0.

Class Rank for “file writer”:

1. java.io.PrintWriter;
2. java.io.PrintStream;
3. java.io.File;
4. java.util.Formatter.

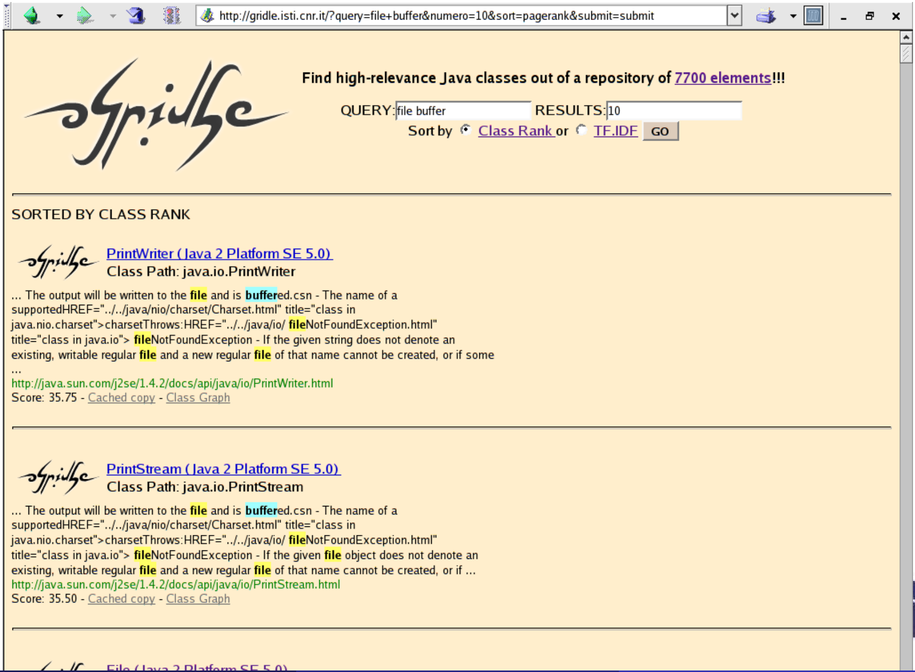
all from Java API.

GRIDLE 0.1

- Ranking using two metrics:
  - TF.IDF (term frequency times inverted document frequency)
  - GRIDLE Rank
- Bells and whistles:
  - Snippets, Links and Reverse Links
- http://gridle.isti.cnr.it

A search engine for SW components

GRIDLE: Google-like Ranking, Indexing and Discovery service for a Link-based Eco-system of software components


http://gridle.isti.cnr.it/?query=file+buffer&numero=10&sort=pagerank&submit=submit

Find high-relevance Java classes out of a repository of 7700 elements!!!

QUERY: file buffer RESULTS: 10

Sort by [✓ Class Rank] or TF.IDF GO

SORTED BY CLASS RANK

PrintWriter (Java 2 Platform SE 5.0)
Class Path: java.io.PrintWriter

... The output will be written to the **file** and is **buffer**ed.csn - The name of a supportedHREF="../../java/nio/charset/Charset.html" title="class in java.nio.charset">charsetThrows:HREF="../../java/io/ **file**NotFoundException.html" title="class in java.io"> **file**NotFoundException - If the given string does not denote an existing, writable regular **file** and a new regular **file** of that name cannot be created, or if some ...
http://java.sun.com/j2se/1.4.2/docs/api/java/io/PrintWriter.html
Score: 35.75 - Cached copy - Class Graph

PrintStream (Java 2 Platform SE 5.0)
Class Path: java.io.PrintStream

... The output will be written to the **file** and is **buffer**ed.csn - The name of a supportedHREF="../../java/nio/charset/Charset.html" title="class in java.nio.charset">charsetThrows:HREF="../../java/io/ **file**NotFoundException.html" title="class in java.io"> **file**NotFoundException - If the given **file** object does not denote an existing, writable regular **file** and a new regular **file** of that name cannot be created, or if ...
http://java.sun.com/j2se/1.4.2/docs/api/java/io/PrintStream.html
Score: 35.50 - Cached copy - Class Graph

Outline

Conclusions

- Usage of Java classes follows the pattern of links among Web pages
- We can rank classes according to this finding
- ...towards a search engine for developers!

Future Work

- Enlarge the code base.
- Compare with results which include source code.
- Change granularity from classes to packages.
- Try different ranking strategies (HITS).

Acknowledgments

- MIUR CNR Strategic Project L 499/97-2000 (5%)
- NextGrid
- CoreGRID
- Università degli Studi di Pisa
- ISTI-CNR