Discovery of Components and Services within the ProActive/Fractal Framework

Diego Puppin

Institute for Information Sciences and Technology
Pisa, Italy

October 10 2005

Outline

Service-Oriented Architecture

Thanks to their ability to be dynamically assembled, services can provide a much required layer of integration between the different global computers that [...] are supporting the operation of the future Information Society. As a result, service-oriented computing is bound to play the role of an ideal overlay computer for Global Computing.[a]

[a] from Martin Wirsing, “Software Engineering for Service-Oriented Overlay Computers: SENSORIA”, FP6-2004-IST-FET

Why Searching?

- A progamming model based on components
- Developing and publishing code as a component
- Component search: general problem
  - since day 1 of programming
- FORTRAN
  - several libraries available for the same problem
- .NET
  - assemblies can be (down)loaded at run-time

Why Searching?

- E-science and e-business applications are evolving, chasing technological progress
  - Monolithic applications are gone
  - Multi-domain, heterogeneous applications
  - Big choice of tools, computing resources etc.
- Real-world applications are getting more complex


When Searching?

- More and more difficult and important for the GRID
  - Resources are not dependable
  - Heterogeneous platform $\rightarrow$ adaptation
  - Extremely variable environment
- Old problem, new challenge

When Searching?

- At design-time
  - Choice of tools that are most fit to the task
  - Minimizing prototyping time
  - Improve SW quality with stable solutions
- At run time
  - Adaptation to guarantee performance contract
  - C2C Interaction
  - Fault management


An open market of components

- Several vendors will sell competing components
  - Different prices
  - Different QoS
  - Different trust
- How to choose?
  - Economy-based scheduling
  - Component search engine

Components: Models

- In general, we can think of a SW component as
  - Independently developed, composable, clear interface, well-defined black-box behavior
- Several choices
  - Deployed Web Service
  - CCM Component
  - CCA use-provide model
  - Java Class, .NET assembly
  - Fractal Component


Goals

- A programming model suitable (that is user friendly and efficient) to program individual components is needed
- Component definition, usage and composition must be arranged according to standards that allow interoperability to be achieved [...]
- Component composition must support and, in the meanwhile, guarantee that scalability is achieved
- Semantics must be defined, precisely modelling both the single component semantics and the semantics of composition[...]
- Performance/cost models must be defined, to allow the development of tools

CONGRATULATIONS FRACTAL!!!

GRID.it

A programming environment is needed that [1]:

1. supports structured exploitation of parallelism
2. allows interoperability with existing SW
3. supports reuse of legacy code

[1] M. Aldinucci, S. Campa et al., title = ”Components for high-performance grid programming in Grid.IT”, in ”Component Models and Systems for Grid Applications ed. Vladimir Getov and Thilo Kielmann, Springer, 2004

Outline

Our proposal: GRIDLERank

- Similar to Google PageRank
- Static analysis of code links
- INTERFACES ONLY
- Every time a class is used, there is a rank boost
- No source code is needed
- No runtime information is needed
- Only public interfaces

Why only interfaces?

- Commercial component will hide source code
- Will also hide runtime profile information
- Interfaces must be public, in order to use a component
- We suggest to base ranking on this
- A composed application should also make public the composition structure
- use/provide ports

Why composition should be public?

- To improve trust
- To become more popular
- To support a standard
- Open Source...
  - Compare with digital libraries
  - bib. references VS full-text


Class Graph: Static vs Dynamic

- Abstract static information $\rightarrow$ Progr. Interface:Class Rank
- Executable information $\rightarrow$ Program Code:Comp. Rank
- Dynamic information $\rightarrow$ Process:Scale Free

A Model for Ranking

Assumptions:

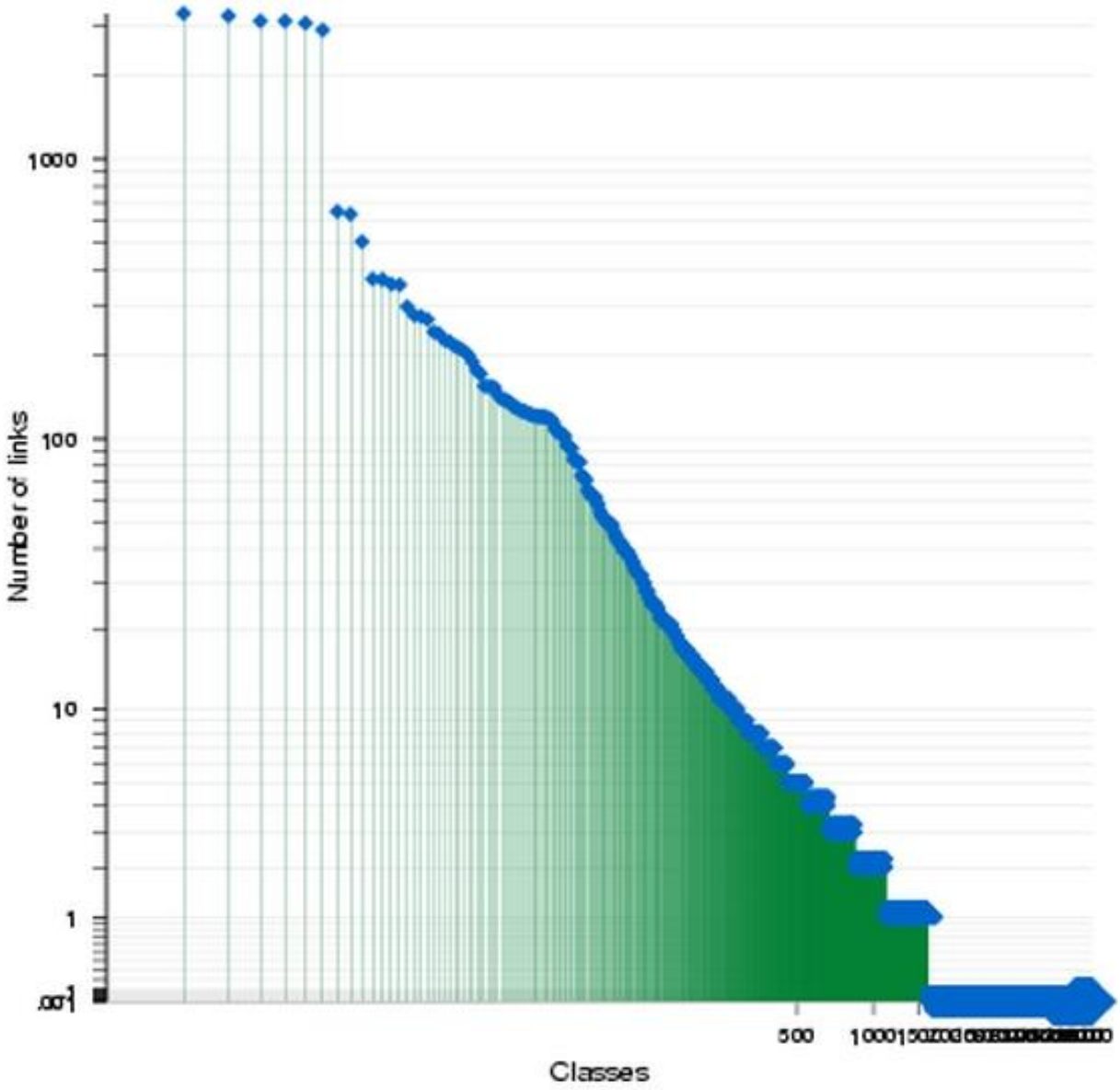
- Avoid ontologies
  - An ontology for the whole Grid?
- Heavy emphasis on LINKS
  - Positive experience on the Web
- No use of source code and dynamic data
  - Hidden in commercial apps
- Use only public interfaces
  - Maybe semantic info can be used
    - Sub/super-types...

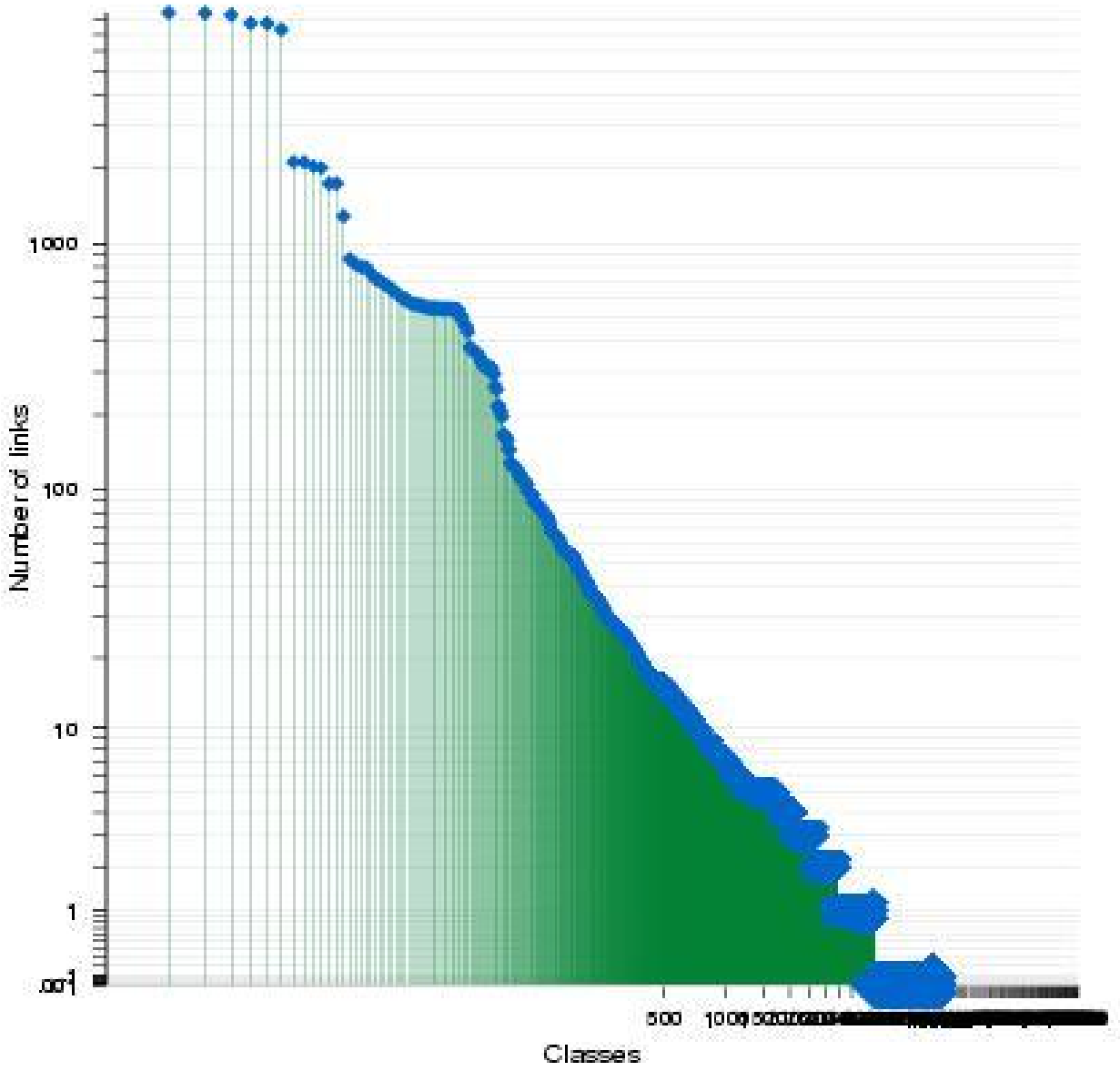
Initial Experiments

- 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*

Social Network in Java Classes

- Java programmer $\leftrightarrow$ component user
  - S/he chooses most general and useful classes
- Power-law behavior
  - Web pages, blogs, social networks sociali etc
  - see On Power-Law Relationships of the Internet Topology, by Faloutsos et al.
- component usage $\leftrightarrow$ Web linking!!!
- Component search $\leftrightarrow$ Web Search


1000
100
10
1
Number of links
500
Classes


Number of links
1000
100
10
1
500
Classes

Class Rank

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

Top-rank classes

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


Outline

Fractal and Proactive

- The Fractal framework was recently re-implemented under Proactive
- Proactive's Objects now have the Fractal component interface

Fractal

Fractal is a modular and extensible component model that can be used with various programming languages to design, implement, deploy and reconfigure various systems and applications, from operating systems to middleware platforms and to graphical user interfaces.

Goals

The goal of Fractal is to reduce the development, deployment and maintenance costs of software systems. It uses some well known design patterns, such as separation of interface and implementation and, more generally, separation of concerns, in order to achieve this goal.

Main features

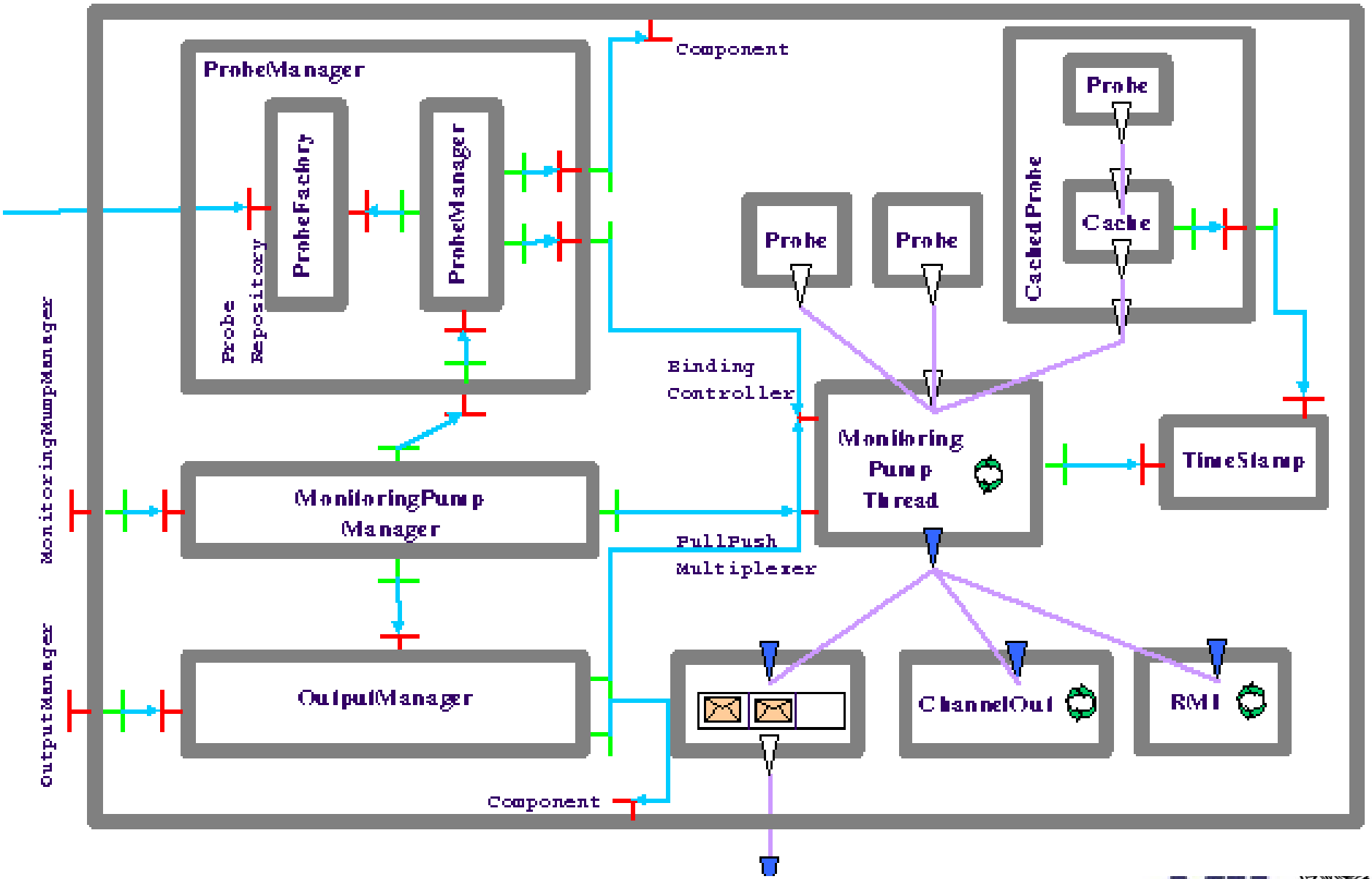
A Fractal component is composed of two parts:

1. a content that manages the functional concerns,
2. a controller that manages zero or more non functional concerns (introspection, configuration, security, transactions, ...).

The content is made of other Fractal components, i.e. Fractal components can be nested (and shared) at arbitrary levels.

More comments

- Reusable object-oriented abstract classes, components and frameworks have lifecyles of their own that are distinct from those of the applications that incubate them
- Objects evolve within and beyond the applications that spawned them
- Structure emerges as objects evolve
- Because the pattern in which they evolve is similar at each level, the overall pattern can be thought of as a fractal curve


ProbeManager
ProbeFactory
ProbeManager
Probe Repository
Component
Probe
Probe
Probe
CachedProbe
Cache
Binding Controller
MonitoringPump Manager
Monitoring Pump Thread
TimeStamp
PullPush Multiplexer
OutputManager
ChannelOut
RMI
Component
MonitoringMumpManager
OutputManager

Outline

Documentation

- No standard for documentation file or distribution.
  - Only interfaces have a standard description (ADL)
- Detailed description of the interfaces is usually released in the form of a Java documentation file for the corresponding Java interface.
- When more detailed information about a given implementation is needed,
  - No easy way to document composite components
  - Basic components = Java classes, documented as such


Limitations of the Java Doc model

1. Methods and fields that are peculiar to the framework (e.g. the *bindFc* mechanism to bind components) are listed among the methods that implements the class logic.
2. Component dependencies (client interfaces) to other components can be inferred only through a detailed analysis of class fields, among which the binding variables are listed.
3. Redundant information is present about, for instance, methods inherited through the Java class structure (e.g. *clone, equals, finalize, getClass, toString...*) or interfaces implemented for other reasons related to implementation details (e.g. *javax.swing.Action, java.lang.Cloneable, java.io.Serializable...*).

Searching?

- Component interface describes the *services* offered
  - We could search and choose services!
- Implementations are not well described
  - We cannot easily choose among competitors!

Integrating GRIDLE and Fractal

- We run GRIDLE over the documentation of Proactive and Fractal
- We studied the Julia Fractal implementation
- and the Proactive Fractal implementation
- We wrote a tool to inject components into the GRIDLE database
  - it takes Fractal ADLs and returns the chain of (recursive) component dependencies

Is Search Possible?

- Component interfaces are well described
  - We could search and choose among different services
- No standard description for implementation
  - How can we choose among competitors?
- What about dependencies?
  - Should we choose the one with fewer dependencies?

Packaging is fundamental

- A package component should include ADL, exec. files, description, list of dependencies, maybe dependent modules
- We should look for packaged components (with few dep.)
- Rank: description, usage within other appls.

Right now...

- Applications = big Fractal components
- Dependencies are not well managed
- WE CAN do some mining on final basic components
  - Is the right level?
- WE CAN see what interfaces are most used
- WE CANNOT match description and components!

Outline

Conclusions

- Growing use of component-oriented frameworks
  - Vision: open-market of COTS components
- Strong need for searching tools
- New metrics needed
  - based on social graph
- Strong potential
  - Use of class $\leftrightarrow$ Linking of pages
  - Public compositions
- Fractal is an interesting comp. model but not clear std. for distribution and packaging
- At what level do we search?

Open issues

- Components depend from interfaces
  - Java and .NET depend from specific classes
- Used components are listed in the *Composition*
  - *Is the ADL always available?*
- *Apparently... there aren't many components available so far*

Acknowledgments

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

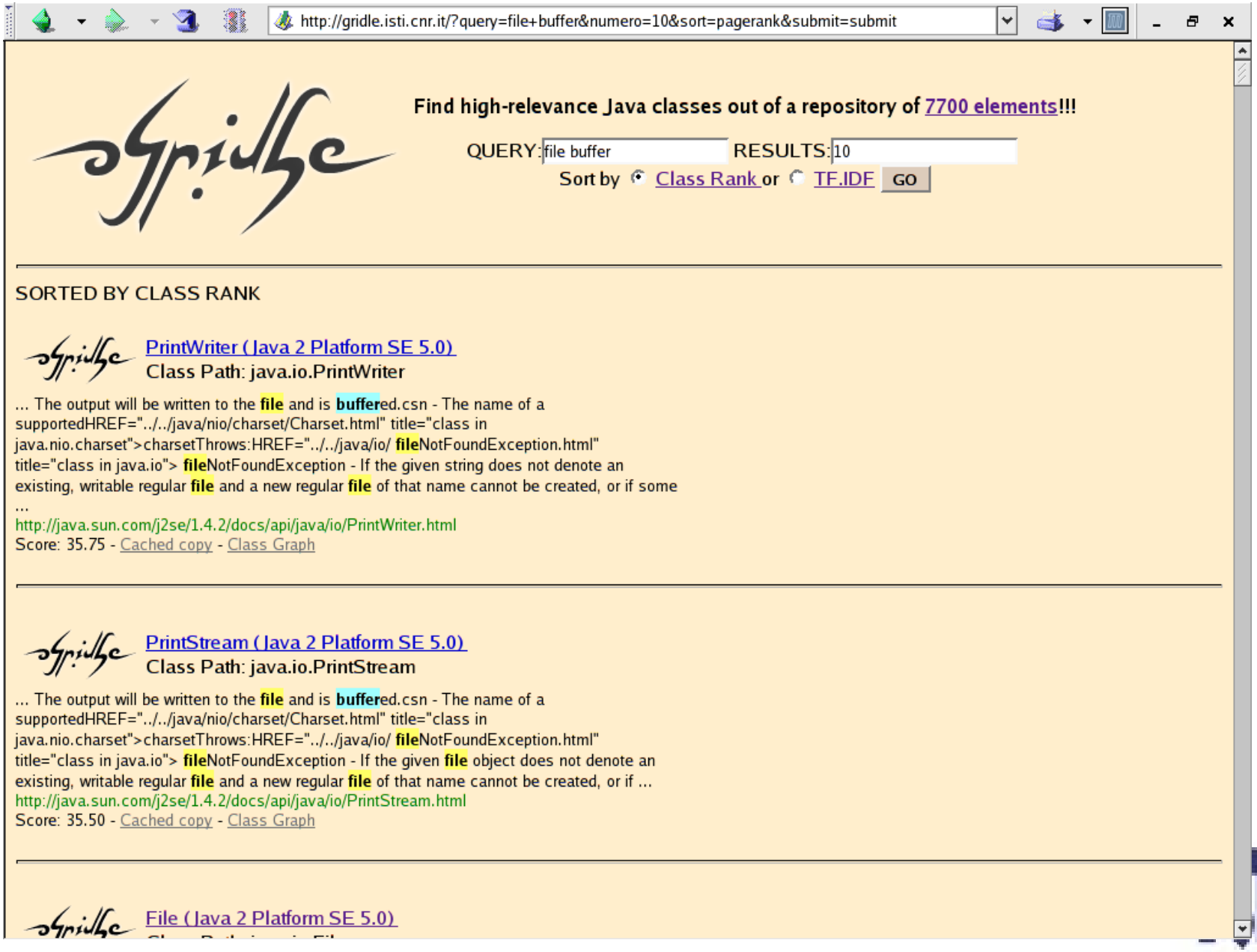
Outline

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 buffered.csn - The name of a supportedHREF="../../java/nio/charset/Charset.html" title="class in java.nio.charset">charsetThrows:HREF="../../java/io/ fileNotFoundException.html" title="class in java.io"> fileNotFoundException - 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 buffered.csn - The name of a supportedHREF="../../java/nio/charset/Charset.html" title="class in java.nio.charset">charsetThrows:HREF="../../java/io/ fileNotFoundException.html" title="class in java.io"> fileNotFoundException - 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
File (Java 2 Platform SE 5.0)