Percerons: A web-service suite that enhance software development process

Percerons is a list of web services, see http://www.percerons.com, that helps software developers to adopt established software engineering approaches in their product development. At this point there are three main service categories that are available inside Percerons:

- Percerons Reuse Repositories,
- Percerons Quality Dashboard, and
- Percerons Design Expert

All the provided services are available through *Percerons Client*, i.e. a desktop application, which locally inspects your Java byte code, performs all available forms of analysis and stores data such as structural quality metrics on a web repository. Respecting the copyrights of your application, neither the source code, or the byte code of your applications are stored online. Finally, the results of analyzing your application are available only to you through a private (authentication protected) web interface.

We note that the public interface of Percerons, is populated with data from open source projects, only for demonstration reasons. The layout and the information provided by the public web-interface is completely the same as the private one.

Percerons Reuse Repositories

Percerons Reuse Repositories aim at promoting systematic in-house reuse for software development companies. Software reuse is a well-known technique that is proposed so as to increase the productivity of software development and testing. Reusing already developed and tested components is significantly reducing the time to market of a new product. The two most widely used pools of reusable software components are:

- Open source software components, and
- In-house software components

The major issues that hinder from reusing open source software components is that their internal structure is vague, that their structure is not familiar to the software developers, and there is no established way of accessing and searching for components that might fit the needs of a software development company. On the other hand, in-house reuse, which solves the abovementioned problems is usually hurdled by the lack of systematic documentation, while developing a system, which leads to the lack of in-house components repositories.

The abovementioned problems are solved by using *Percerons Reuse Repositories* as follows:

- We provide the infrastructure for building an in-house reuse repository by components, reverse engineered from binary code. These components include domain specific knowledge, in the sense that they are retrieved from projects of the company itself, and therefore are more

understandable to your software engineers. Additionally, these components are automatically retrieved and offered to you without own cost, from a search engine that enables filtering and sorting the retrieved components.

- We provide a search engine that enables access to 250.000 open source components. These components are qualified as easy to compile and understandable. Also, these components are reversed engineered to class diagram, and assessed with all possible evaluations offered by the *Percerons Quality Dashboard*.

Percerons Quality Dashboard

Percerons Quality Dashboard is a "one-stop-quality" set of tools for Java software. We provide first class reliable assessments covering all aspects of source quality. Through our repertoire of on-line scalable services you will be able to assess different perspectives of quality. The quality attributes that are being assessed at this point are:

- Design Quality,
- Reuse Capabilities, and
- Instability

Software *design quality* refers to how the product meets non-functional requirements that support the delivery of the functional requirements, such as robustness or maintainability, and therefore the degree to which the software was produced correctly. *Design quality* is assessed by the use of one of the most known OO metrics suite, the one introduced by Chidamber and Kemerer (CK metrics suite). The suite is based on the following metrics:

- *NOC*: Count of number of classes in the system.
- *WMC*: A class's weighted methods per class is the average of the complexities of its methods. As a measure of complexity we use the cyclomatic complexity.
- *DIT*: The depth of inheritance tree (DIT) metric provides for each class a measure of the inheritance levels from the object hierarchy top.
- *NOCC*: A class's number of children (NOC) metric measures the number of immediate descendants of the class.
- *CBO*: The coupling between object classes (CBO) metric represents the number of classes coupled to a given class. This coupling can occur through method calls, field accesses, inheritance, arguments, return types, and exceptions.
- *RFC*: The metric called the response for a class (RFC) measures the number of different methods that can be executed when an object of that class receives a message (when a method is invoked for that object).

- *LCOM*: A class's lack of cohesion in methods (LCOM) metric counts the sets of methods in a class that are not related through the sharing of some of the class's fields. The definition of this metric considers all pairs of a class's methods. In some of these pairs both methods access at least one common field of the class, while in other pairs the two methods to not share any common field accesses. The lack of cohesion in methods is then calculated by subtracting from the number of method pairs that don't share a field access the number of method pairs that do.

Software *reuse capability* refers to the extent to which the artifacts of a specific product can be reused in different products of the company. *Reuse capability* is synthesized by three very important factors, while reusing software:

- *Structural Reusability*: Is calculated based on the QMOOD model, where reusability is defined as the extent to which a product can be reused in different software.
- *Functionality*: Represents the portion of the current system to which the specific component is used. As much functionality the component offers to the system, the more probable is for the component to be reused in a different application.
- *External Dependencies*: Represents the hardness to compile the specific component, if reused in a different project. A zero value of this variable represents components that can be reused “as is” in different systems.

Finally, software *instability* refers to the probability of a class to change in the next version of the system, due to the ripple effect. Instability is calculated according to a published model in a top journal of the software engineering domain. The above-mentioned evaluations can be used for:

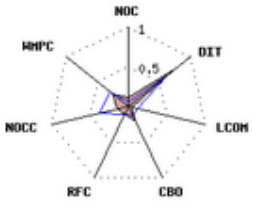
- *Perfective maintenance prioritization*: Classes that lack design quality and are more change prone should be prioritized in company’s refactoring activities
- *Preventive maintenance prioritization*: Classes that provide the more functionality to the system, and classes that are change prone should be more seriously considered, while performing activities to identify bugs.
- *Adaptive maintenance*: The extendibility (related to design quality) of classes that provide the most functionality to the system and are change prone should have priority in refactoring activities, so as to ease future adoptions to the system and decrease technical debt.
- *Applying the reuse strategy*: The classes/components/project that are characterized from high reuse capabilities should have higher priority when selecting classes from the *Percerons Reuse Repositories*.

Home | Reuse Repositories | Quality Dashboard | Design Expert | FAQ | About

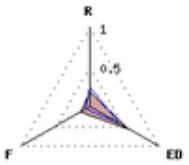
Bolzpatz2006, version: 1.0.3 from application domain: Game

Quality Evaluation at System Level

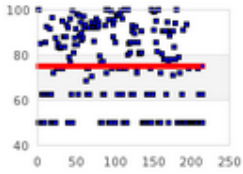
CK Metrics Suite



Reuse Capabilities



Instability



Rating: ★★☆☆☆

Number of Classes
Depth of Inheritance Tree
Lack of Cohesion of Methods
Coupling Between Objects
Response for a Class
Number of Class Childeren
Weighted Method per Class

Rating: ★★☆☆☆

Structural Reusability
External Dependencies
Funcionality

Rating: ★½☆☆☆

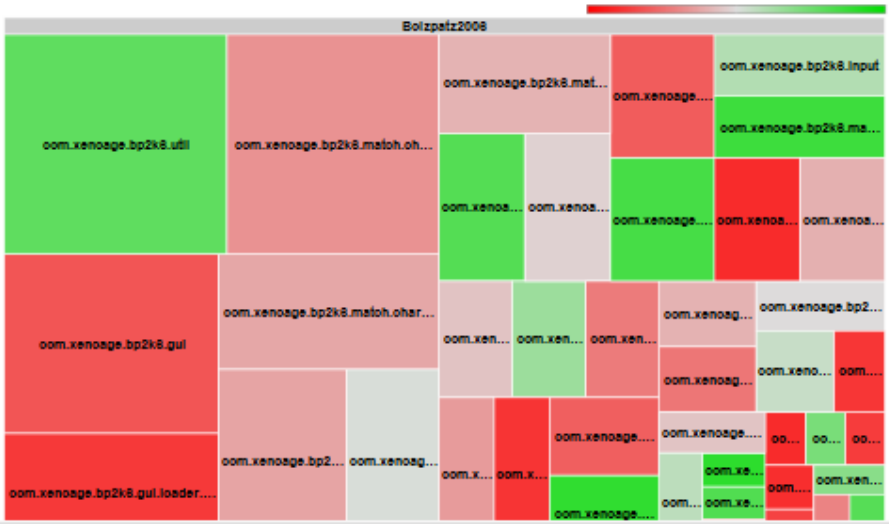
Change Proneness

Overall Quality Evaluation per Package/Class

Back to root

Quality Indicator

Poor — Excellent



Percerons Design Expert

Percerons Design Expert provides a set of tools that help software engineers and more specifically designers and architects, to perform the optimum design decisions. At this point the *Design Pattern Advisor* service, enables designers to consider the quality trade-offs while using GoF design patterns.