Nuttige sites:

<http://deors.wordpress.com/2014/03/20/sonarqube-plugins-1/>

<http://docs.codehaus.org/display/SONAR/Coding+a+Plugin>

<http://www.docfacto.com/blogs/2013/05/29/the-adventures-of-sonar-plugin-development/>

SonarQube provides a clean way of developing the polymorphic view extension. SonarQube is built in such a way that it can be easily extended. You can add new plugins (custom metrics, sensors, widgets). This is also what we need to do, build a new visualization plugin, that can show a visualization of some collection of resources according to one or more user-configurable metrics.

But the code is not documented, what makes it very hard to know how and where everything should be implemented.

The entry point for every SonarQube plug-in is a class that extends the org.sonar.api.SonarPlugin abstract class. A plug-in is modelled as a list of extensions (implementations of the interface org.sonar.api.Extension).

Probably need of metrics, a sensor/a decorator and a widget

Metric classes define custom metrics, Sensors are used to scan projects and calculate metric values (measures, in SonarQube terminology), Decorators are used to calculate derived values for a given project resource (typically using several measures to do the calculation) and Widgets are used to present information in SonarQube dashboard.

Plugin 🡪package and deploy it to a working SonarQube instance, as you would do with any other plug-in.

Nodige klassen

* Org.sonar.api.batch.Sensor
* Org.sonar.api.batch.Decorator
* Org.sonar.api.web.Widget
* org.sonar.api.batch.SensorContext
* org.sonar.api.resources.Project
* org.sonar.api.batch.Sensor
* org.sonar.api.measures.Metrics.
* org.sonar.api.web.RubyRailsWidget.
* org.sonar.api.web.AbstractRubyTemplate