ITEA 3 Office
High Tech Campus 69-3
5656 AG Eindhoven
The Netherlands
Tel : +31 88 003 6136
Fax : +31 88 003 6130
Email : info@itea2.org
Web : www.itea2.org
ITEA 2 is a EUREKA strategic ICT cluster programme

INFORMATION TECHNOLOGY FOR EUROPEAN ADVANCEMENT

D4.1: Data Analysis Platform

MEASURE

Edited by Jérôme Rocheteau on 07/05/2018
Edited by Alessandra Bagnato on 11/05/2018
Edited by Jérôme Rocheteau on 11/05/2018
Edited by Sarah Dahab on 19/05/2018
Edited by Alin Stefanescu on 25/05/2018
Edited by Jérôme Rocheteau on 04/06/2018
Reviewed by Alin Stefanescu on 06/06/2018
Reviewed by Sana Mallouli on 06/06/2018

Executive summary

This document both presents the MEASURE Analysis Platform services and the analysis tools that have already been integrated into the MEASURE Analysis Platform. These analysis tools correspond either to state-of-the-art analysis algorithms – constraint-based filtering with SOFTEAM's Quality Guard and clustering with ICAM's MELKI – and innovative analysis such as the MINT correlation tool co-developed by Montimage and IMT, the IMT's Metrics Suggester and University of Bucharest's STRACKER. It also makes possible to enrich this analysis platform by other tools thanks to the MEASURE Analysis Platform HTTP API.

In fact, the MEASURE project early objectives of 5 integrated analysis tools have been achieved. The objective of an extensible analysis platform has been achieved as well.

Table of Contents

1. Introduction

1.1. Role of this deliverable

This document aims at presenting the MEASURE Analysis Platform API and the different analysis tools that have been integrated into the MEASURE Analysis Platform yet.

1.2. Structure of this document

This document is structured as follows:

1. The MEASURE Analysis Platform API it presented throughout examples for developing analysis tool and correctly integrated it into the MEASURE Analysis Platform. Hence the section name: Analysis Tool Integration.
2. The next section is devoted to 5 Analysis Tools that have yet been integrated into the MEASURE Analysis Platform:
    1. Quality Guard, a constraint-based filtering tool by SOFTEAM
    2. MINT, a metrics correlation analysis tool by Montimage (MTI) and IMT
    3. Metrics Suggester, a metrics suggester tool by IMT
    4. M·ELKI, a clustering algorithm tool by ICAM
    5. STRACKER, a metrics tracker and suggester tool by University of Bucarest

1.3. Relationship with others MEASURE deliverables

This deliverable is closely related to deliverable D4.2 that presents the analysis tools' underlying algorithms with numerous details. It is also related to the deliverable D3.1 that presents the MEASURE Platform, as the latter provides measurements to analysis tools that are previously collected from executable measures,

1.4. Contributors

These deliverable contributors are:

- SOFTEAM for the section §2 (Analysis Tool Integration) and the section §3.1 (Quality Guard).
- Montimage for the section §3.2 (MINT)
- IMT: for the section §3.2 (MINT) and §3,3 (Metrics Suggester)
- ICAM for the section §3/4 (M·ELKI).
- University of Bucharest for the section §3.5 (STRACKER).

Reviews have been performed by SOFTEAM, Montimage, ICAM, and UniBuc.

2. Analysis Tool Integration

The Measure Platform allow to **Deploy**, **Configure**, **Collect**, **Store**, **Combine**, and **Display** measures and metrics in relation with software development process.

Services provided by the metric platform are completed by the **Analysis Tools**, a set of external services which work on the historical measures values in order to provide advanced and valuable analysis function to the platform.

In order to support a large set of analyses services and do not limit to it a specific technology, the Analysis Tools are external processes. Although external, we wanted a deep integration between the platform and the analysis tools. We solved this issue in the following way:

- The Measure platform provide a REST API which allows an analysis tool to register it on the platform, to receive notification from the platform and access to information related to project defined and measure collected by the platform.
- On its side, the analysis tool provides some web pages which will be embedded into the platform web application.

This Measure platform is organised by Projects. It must be the same for analysis tools that can be deployed or not in each project. When a project decides to activate an analysis tool, the tools have to provide specific analysis services to the project scoped by the project configuration.

Embedded View

In order to integrate deeply the analysis tool to the Measure Platform, the analysis tools have to provide some web pages which will be embedded to the platform web application.

Each of these views are defined on the platform side by a specific URL. For project specific views, this URL is different for each project.

Global Configuration Page (optional)

If the analysis tool requires a way to provide some configuration interface which will be shared by all project, it can provide a global configuration web page.

Project Specific Configuration Page

Configuration page which are specific for each project.

Project Specific Main View

Main view of the analysis tool which are specific for each project.

Dashboard Card

Optional small view which can be integrated to projects dashboards in order to provide some key information to project managers related to the service provided by the analysis tool.

Integration Life Cycle

- **Registration**: At startup of the Analysis Tool, it must register itself to the platform using the Registration service. This would allow the project to activate the analysis tools.
- **Wait for Notifications**: The Analysis Tool must listen to notifications from the platform in order to know when a project requests the usage of the analysis tool. The notification (Alert) system is based on

pooling system. The Analysis tool pool the platform periodically using the alert service to received notifications.

- **Configure Analysis**: When a project activates an analysis tool, the analysis tool must configure it for the project and provide URLs for the project-specific configuration page, the project main view and optionally the dashboard cards.

- **Analyse the Project**: When configured, the analysis tool can start its analysis work for the specific project. In order to perform this work, the analysis tool can explore the project configuration using the various services provided by the Measure platform. It can also configure new Alerts to receive notifications when the project configuration has changed.

Registration Service

This service registers an analysis tool to the Measure platform.

HTTP: PUT /api/analysis/register

Input Data (application/json):

```
{
  "configurationURL": "string",
  "description": "string",
  "name": "string"
}
```

Java Client Implementation:

```
public class AnalysisService {
    private String name;
    private String description;
    private String configurationURL;

    public AnalysisService(){
    }
    ... get and set
}
....
public void registerAnalysisTool(AnalysisService service){
    RestTemplate restTemplate = new RestTemplate();
    try {
        restTemplate.put(serverURL +"/api/analysis/register",service);
    } catch (Exception e) {
        e.printStackTrace();
        return;
    }
}
```

Alert Service

The Measure Platform alert system is based on a pooling system. The analysis tool must pull periodically the Measure platform to receive all new notification arrived between the last pool and now.

Notification related to analysis tool activation and deactivation are automatically configure by the platform when a new Analysis tool register itself to the platform. The analysis tool can subscribe to others kind of notifications using the Alert REST API.

Alert Type: ANALYSIS_ENABLE

Description: A Project sends an activation request for the Analysis Tool. It's not required for analysis tool to subscribe to this alert, the subscription is automatic.

Properties:

- ANALYSISID: Id of the instance of analysis associated with this request on platform side

Alert Type: ANALYSIS_DESABLE

Description: A Project indicate that the analysis service is not required anymore. It's not required for analysis tool to subscribe to this alert, the subscription is automatic.

Properties:

- ANALYSISID: Id of the instance of analysis associated with this request on platform side

Alert Type: MEASURE_ADDED

Description: A new Measure is added the the project

Properties:

- MEASUREID : Id of the Measure

Alert Type: MEASURE_REMOVED

Description: A Measure is removed form the project

Properties:

- MEASUREID : Id of the Measure

Alert Type: MEASURE_SCHEDULED

Description: A Measure is not collected periodicly for the project

Properties:

- MEASUREID : Id of the Measure

Alert Type: MEASURE_UNSCHEDULED

Description: A Measure is not collected anymore by the project

Properties:

- MEASUREID : Id of the Measure

Retrieve Platform Alerts

This service retrieves the alerts form the platform for a specific analysis tool

HTTP: GET /api/analysis/alert/list/{AnalysisToolName}

Output Data :

```
{
  "alerts": [
    {
      "alertType": "string",
      "projectId": 0,
```

```
      "properties": [
        {
          "property": "string",
          "value": "string"
        }
      ]
    }
  ],
  "from": "2018-03-13T12:16:33.164Z"
}
```

Java Client Implementation:

```
public class AlertReport {
    private Date from;
    private List<AlertData> alerts = new ArrayList<>();

    public AlertReport(){
    }
      ... get and set
}

public class AlertData {
    private String alertType;
    private Long projectId;
    private List<AlertProperty> properties = new ArrayList<>();

    public AlertData() {
    }
      ... get and set
}

public class AlertProperty {

    private String property;
    private String value;

    public AlertProperty(){
    }
      ... get and set
}

public AlertReport getAlerts(String analysisTool){
    RestTemplate restTemplate = new RestTemplate();
    try {
        return restTemplate.getForObject(serverURL
+"api/analysis/alert/list/?id="+analysisTool,AlertReport.class);
    } catch (Exception e) {
```

```
            e.printStackTrace();
        }
        return null;
    }
```

Configuration Service

This service configures the Analysis Tool on the Measure Platform level for a specific project. This configuration consists to define URL of embedded visualisation provided by the analysis tool for the project.

HTTP: PUT /api/analysis/configure

Warning: The analysis configuration input data required an projectAnalysisId. This id is provided by the platform as properties of the ANALYSIS_ENABLE and ANALYSIS_DESABLE notification message.

Input Data :

```
{
  "cards": [
    {
      "cardUrl": "string",
      "label": "string",
      "preferedHeight": 0,
      "preferedWidth": 0
    }
  ],
  "configurationUrl": "string",
  "projectAnalysisId": 0,
  "viewUrl": "string"
}
```

Java Client Implementation:

```
public class AnalysisConfiguration {
    private Long projectAnalysisId;
    private String viewUrl;
    private String configurationUrl;
    private List<CardConfiguration> cards = new ArrayList<>();

    public AnalysisConfiguration() {
    }
    ... get and set
}

public class CardConfiguration {

    private String cardUrl;
    private String label;

    private Integer preferedWidth;
```

```
    private Integer preferedHeight;

    public CardConfiguration(){

    }
... get and set
}

public void configureAnalysisTool(AnalysisConfiguration service){
    RestTemplate restTemplate = new RestTemplate();
    try {
        restTemplate.put(serverURL +"/api/analysis/configure", service);
    } catch (Exception e) {
        e.printStackTrace();
        return;
    }
}
```

Subscribe to Alerts

This service allows an analysis tool to subscribe to a new alert related to a specific project,

HTTP: PUT /api/analysis/alert/subscribe

Input Data :

```
{
  "analysisTool": "string",
  "eventType": "ANALYSIS_ENABLE",
  "projectId": 0,
  "properties": [
    {
      "property": "string",
      "value": "string"
    }
  ]
}
```

Unsubscribe to Alerts

This service allows the analysis tool to unsuscribe to an alerte.

HTTP: PUT /api/analysis/alert/unsubscribe

Input Data:

```
{
  "analysisTool": "string",
  "eventType": "ANALYSIS_ENABLE",
  "projectId": 0,
  "properties": [
```

```
    {
      "property": "string",
      "value": "string"
    }
  ]
}
```

Platform Querying Services

The platform provides several others services which can be used by the analysis tools to retrieve platform and project configurations data, information related to measures and measurements and more.

The list of available services can be consulted via Swagger directly on deployed Measure platform. To access this specification, one must be connected as Administrator to the platform. The complete API specification is available on Administration > API menu

Some example of available HTTP services:

- GET /api/measure/findall : List all measures
- GET /api/measure/{id} : information related to a specific measure
- GET /api/measure-properties/{id} : List of scope properties associated with one measure
- GET /api/projects : List all projects
- GET /api/projects/{id} : Information related to a specific project
- GET /api/phases/byproject/{id} : Get phases of a specific project
- GET /api/phases/{id} : Information of a specific phase
- GET /api/measure-instances : List of all measure instances
- GET /api/measure-instances/{id} : Information of a specific measure instance
- GET /api/project-measure-instances/{id} : List of measure instances of a specified project
- GET /api/measure-instance/scheduling/execute/{id} : Execute a specific measure
- GET /api/measure-instance/scheduling/start/{id} : Activate scheduling of a specific measure
- GET /api/measure-instance/scheduling/stop/{id} : Deactivate scheduling of a specific measure

3. Integrated Analysis Tools

3.1. Quality Guard (SOFTEAM)

Quality Guard Tool

The Measure Platform allows us to collect of measures on various aspect of a development process. In order to monitor these measurements, the Quality Guard Tool allows project managers and quality experts to define quality constraints which allow to compare in real-time measures collected by the platform to predefined measure thresholds.

The defined constraints are based on a simple expression language which supports logic operators like “AND”, “OR”, “NOT”, comparison operators like “>”, ”<”, ”<=”, ”<=”, constants and measurements value collected by the platform.

This constraint expression is interpreted by the Constraint Evaluator component whose role is to detect constraints violations. This violation is managed by a Violation Manager component that will apply the strategy defined by the quality gate in this eventuality.

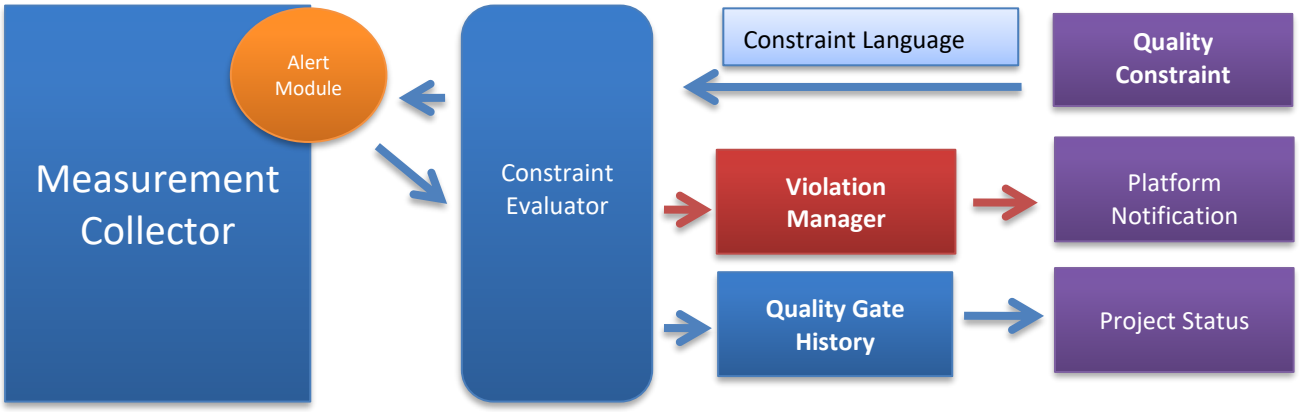


Figure 1 : Constraints violations detection in Measure Platform

The Quality Guard Tool has been released as an open source tool available on GitHub.

- Last tool release can be download at this address:
- A user manual is available online:
- The source code itself can be downloaded from GitHub:

Business Added Value

The Measure Platform is data collection and aggregation platform which collects information from the entire product development chain. This data provides a good picture of the state of a software development process but can’t reveal their full potential if they are not part of an overall quality approach. A Quality Gate is a process which reviews the quality of all factors involved in production. A part of quality management process, quality controls focused on fulfilling quality requirements.

The Quality Guard approach is a standard way to enforce a quality policy in an organization. The goal of Quality Guard is to answer several questions related to the actual state of a software product:

- What is the actual status of my product for each development phases?
- Can i move to the next development phases of my product?
- Can I deliver a project to production today or not?

- Is there a critical issue which appear during the past week?

The answer of these key questions can be summarized as Quality Gate, acceptance criteria reviews that can be used throughout any project. It can be seen as a set of predefined quality criteria that a software development project must meet in order to proceed from one stage of its lifecycle to the next.

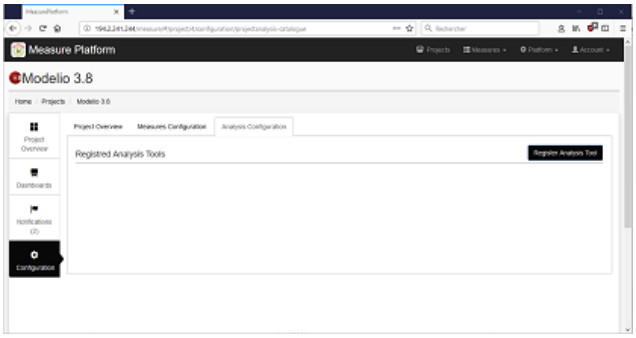
In order to integrate data collected by the platform in a quality process, this data must be constantly compared to threshold values, the quality criteria, identified by quality engineer relying on his expertise and a history of data previously collected. The Quality Guard tool allow then quality engineer to formalize quality criteria as quality rules and, integrated to a notification system which report all quality violation in a synthetic way, allow the project manager to easily monitor the evolution of the state of his project based on this quality criteria.

Main Analysis Services

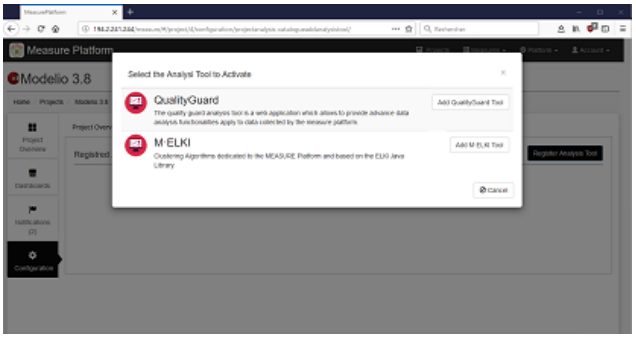
Activate the Quality Guard function in Measure Project.

As for others analysis tools, a Measure project which would like to use Quality Guard services must activate it in Project Configuration view.

- As Project Administrator, go to Configuration view of the project and select the Analysis Tools tab.

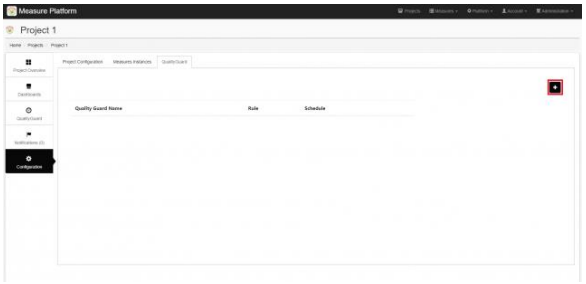


- Click on the *Register Analysis Tool* button.
- Select the Quality Guard tool in the list of available analysis tools.

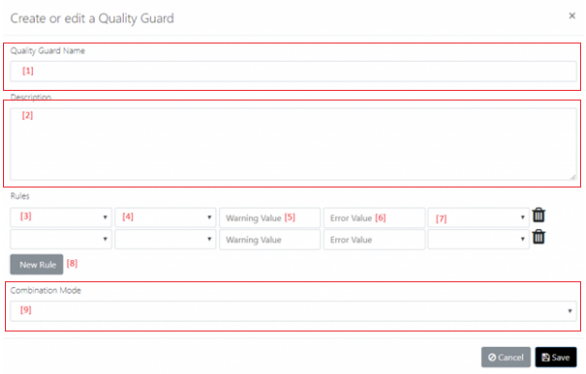


Configure a new Quality Guard Rules

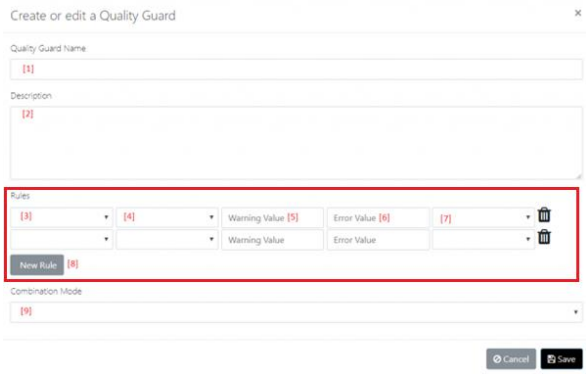
- Create new Quality Guard Rule



- Name and provide a description for the rule. (1,2)
- Chose the aggregation mode: The aggregation model defines how the tool will combine the different conditions. (9)

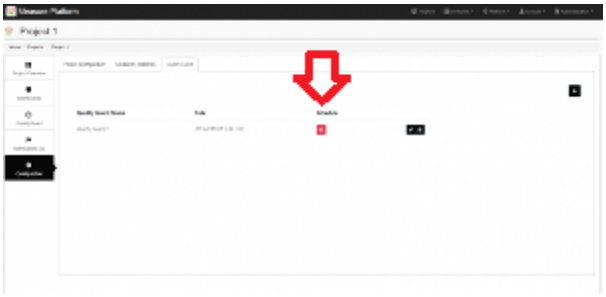


- Define and configure a new Quality Condition
  - Select the monitored measure (3)
  - Select the condition operation (Superior or Inferior) (4)
  - Define the Alert and the Error threshold (5,6)
  - Define the aggregation interval (The measure value is compare to the threshold based on the average value of all collected value during the interval) (7)



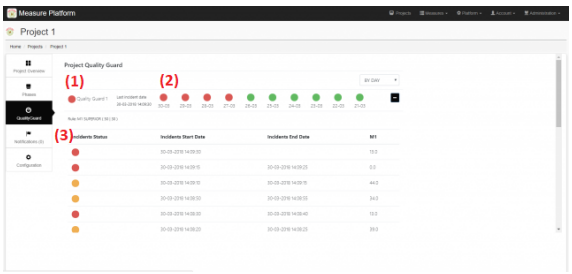
Activate and Deactivate a Quality Rule

- Quality Rules can be activated or deactivated independently using the *Scheduling* button



Visualise Quality States

- In Quality View, show current state of all quality rules. (1)
- Show the history of quality states. (2)
- Show the list of last quality violations. (3)



Tool Architecture Overview

The Quality Guard Analysis Tool is an independent web application based on the Spring Boot framework. Constraints evaluations, violation management and history analysis are implemented using spring services. A full integration with the Measure Platform is ensured using the dedicated API defined by the platform.

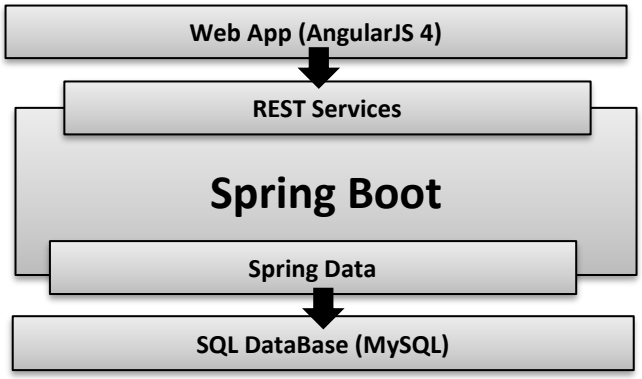


Figure 2 : Quality Guard Tool Technological stack

Technology stack on the client side

- Angular 4 or AngularJS v1.x
- Responsive Web Design with Twitter Bootstrap
- HTML5 Boilerplate

Technology stack on the server side

- Spring Boot for easy application configuration
- Spring Security
- Spring MVC REST + Jackson
- Spring Data JPA + Bean Validation

Deployment and Configuration

Prerequisites

The Quality Guard Analysis Tool can be deployed in both Linux or Windows systems. To be executed, the tool requires the installation of a MySQL database, and Java 1.8.

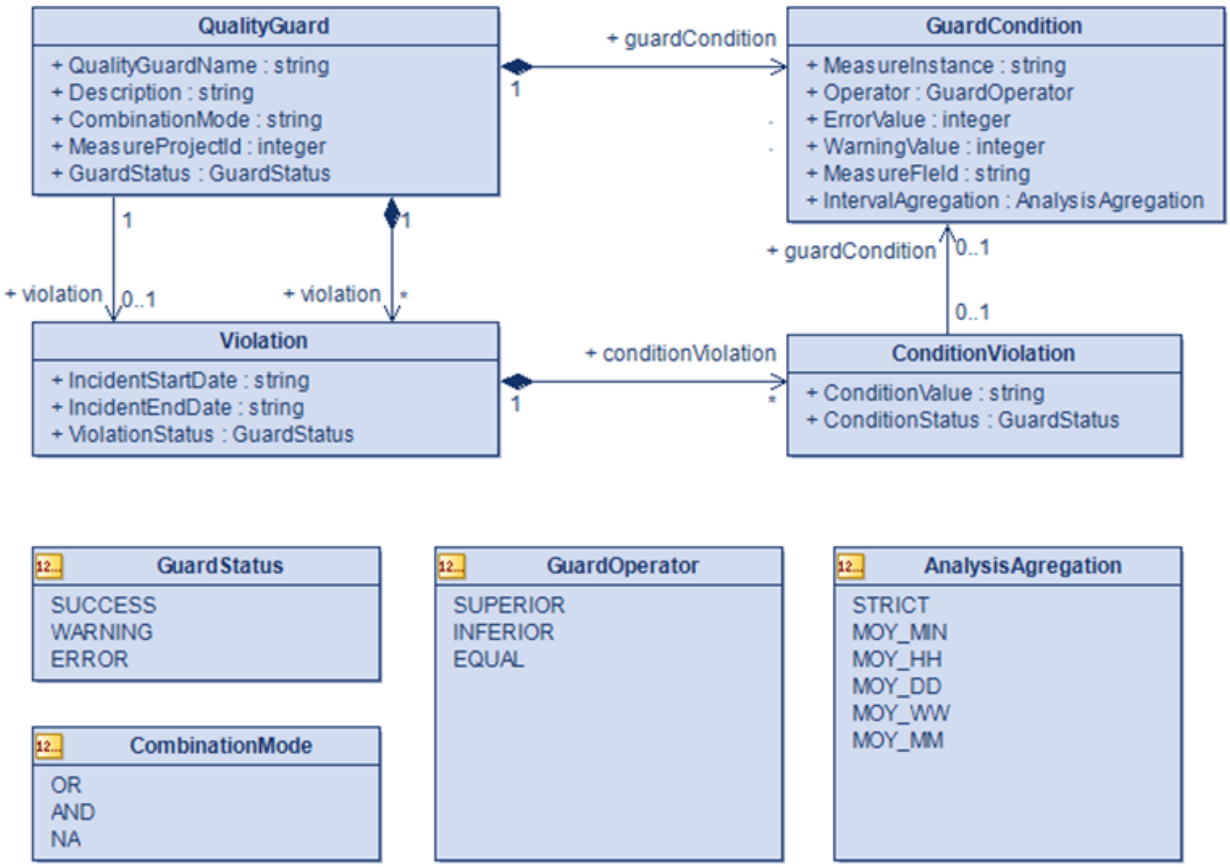


Figure 3 : Quality Guard Data Model

- MySQL Installation
  - Download MySQL Community Server ver. 5.7 or above: https://dev.mysql.com/downloads/mysql/
  - Install MySQL using these instructions: https://dev.mysql.com/doc/refman/5.7/en/installing.html
  - Create a new database named "qualityguardanalysis".
- Java 1.8 Installation
  - Download and install the jdk8 in your environment: http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html
- Measure Platform V0.8.1 or above
  - Download : https://github.com/ITEA3-Measure/MeasurePlatform/releases
  - Installation : https://github.com/ITEA3-Measure/MeasurePlatform/wiki/Platform-Installation

Tool *Installation*

- Download the last released version of the QualityGuardAnalysisTool : https://github.com/ITEA3-Measure/QualityGuardAnalysis/releases
- Unzip the project in your tool directory.
- Using MySQL administration tool, create a new empty database named “qualityguardanalysis”.

Tool *Configuration*

The Quality Guard Tool is parametrize using a property file named “**applicaton.properties**”. This property file has to be put in the same folder of application binary.

Property	Description	Default Value
measure-platform.url	URL of the measure platform	http://localhost/
analysis-tool.ws.url	URL of the quality guard analysis tool	http://localhost:8585/
analysis-tool.url	URL of the quality guard analysis tool	http://localhost:8585/#/
spring.datasource.url	JDBC URL of the database ex: "jdbc:mysql://"+ ip of computer in which is installed MySQL database name.	jdbc:mysql://localhost:3306/qualityguardanalysis
spring.datasource.username	Loign MySQL.	root
spring.datasource.password	Password MySQL.	root
spring.datasource.driver-class-name	Driver JDBC for MySQL	com.mysql.jdbc.Driver
server.port	Port of the QualityGuardAnalysisTool web application	8585

Run the Quality Guard Analysis Tool

Start the Quality Guard Analysis Tool:

```
java -jar quality-guard-analysis-0.0.1-SNAPSHOT.jar
```

3.2. MINT (MTI-IMT)

Description

Metrics Intelligence Tool (MINT) is a software solution designed to correlate metrics from different software development life cycle in order to provide valuable recommendations to different actors impacting the software development process. MINT considers the different measurements collected by the MEASURE platform as events occurring at runtime. The correlation is designed as extended finite state machines (EFSMs) allowing to perform Complex Event Processing (CEP) in order to determine the possible actions that can be taken to improve the diverse stages of the software life cycle and thus the global software quality and cost.

This tool is available as web application with NodeJs and uses Montimage's MMT-correlator to implement the extended finite state machines.

The integration to the Measure Platform is made using the provided API to register and configure Mint as an analysis tool.

Installation

Node.js is cross-platform meaning that Mint can work on Windows, OSX and Linux.

Mint requires Node.js 8.9.0 or above and the installation of a MySQL database and a database named mint_db, tables are created automatically if they do not exist.

Once the project is cloned or downloaded to the machine, it's required to install the packages that Mint depends on, using:

$ yarn install

Edit the config.ini file under config directory to specify the mysql data and measure platform url. The default values are:

measure-platform.url=http://localhost:8085

mysql.datasource.database=mint_db

mysql.datasource.host=localhost

mysql.datasource.username=root

mysql.datasource.password=root

The first time is indispensable to populate the database with the machines description, for this run the command:

$ node_modules/.bin/sequelize db:seed:all

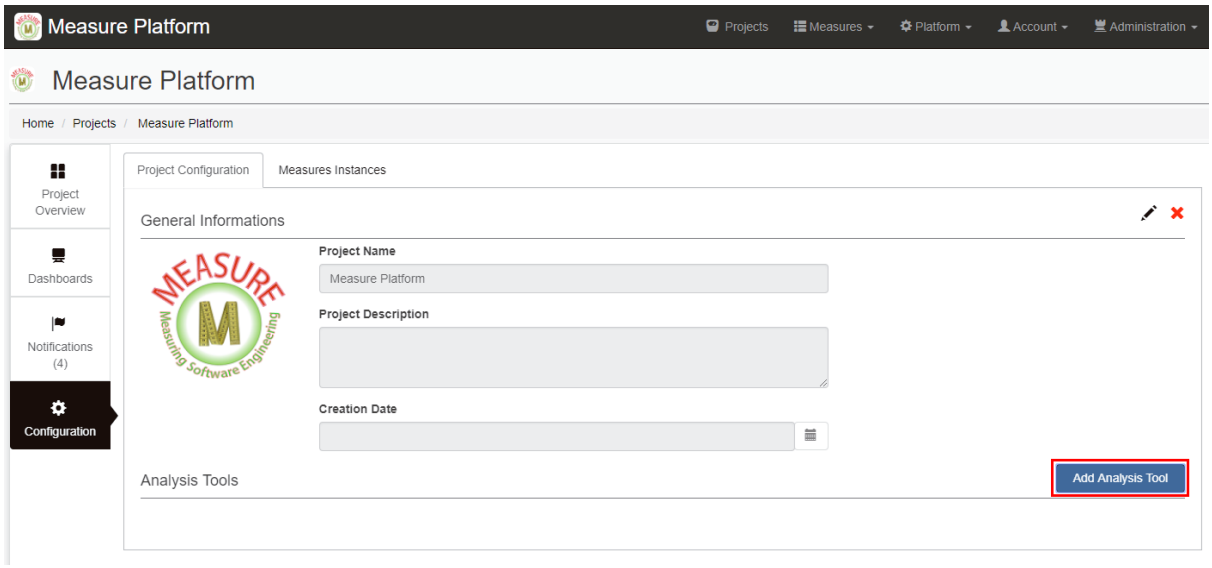
Finally run the tool using:

$ yarn run start

Registration

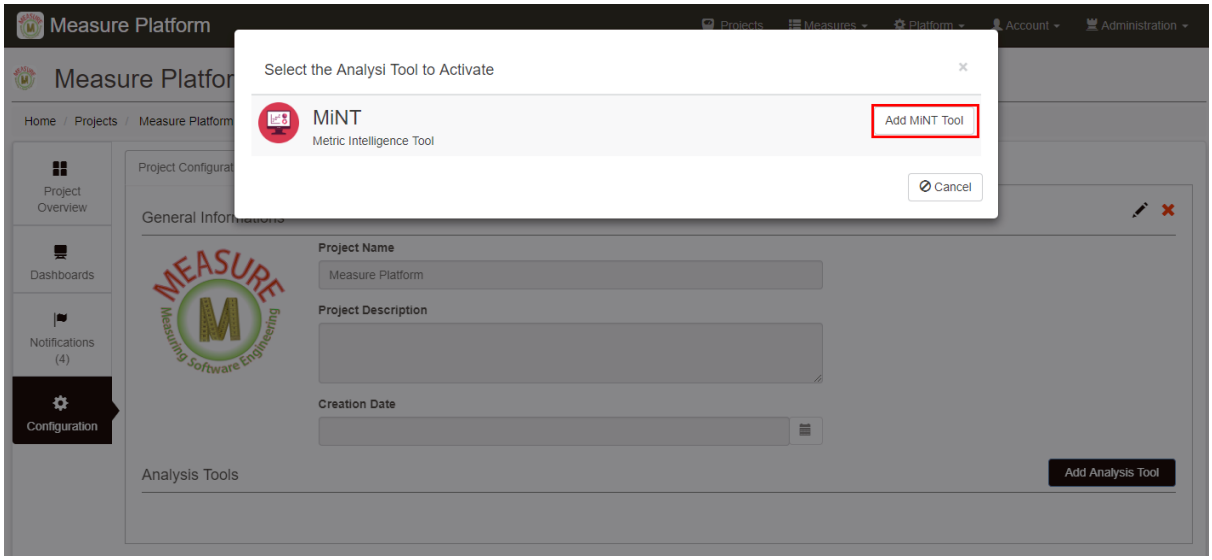
Once the tool is running it can be registered into the Measure platform.

Access to the configuration tab of the project page configuration and click on "Add Analysis tool"

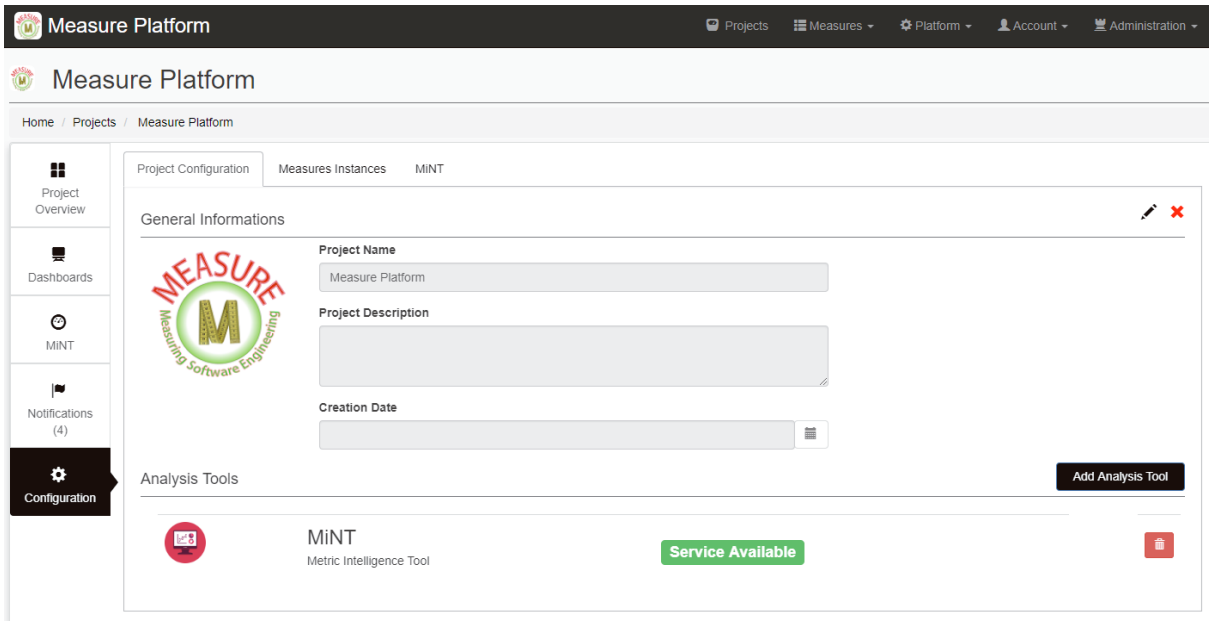


Mint should be visible in the list of analysis tools to activate.

Click on “Add Mint Tool”.



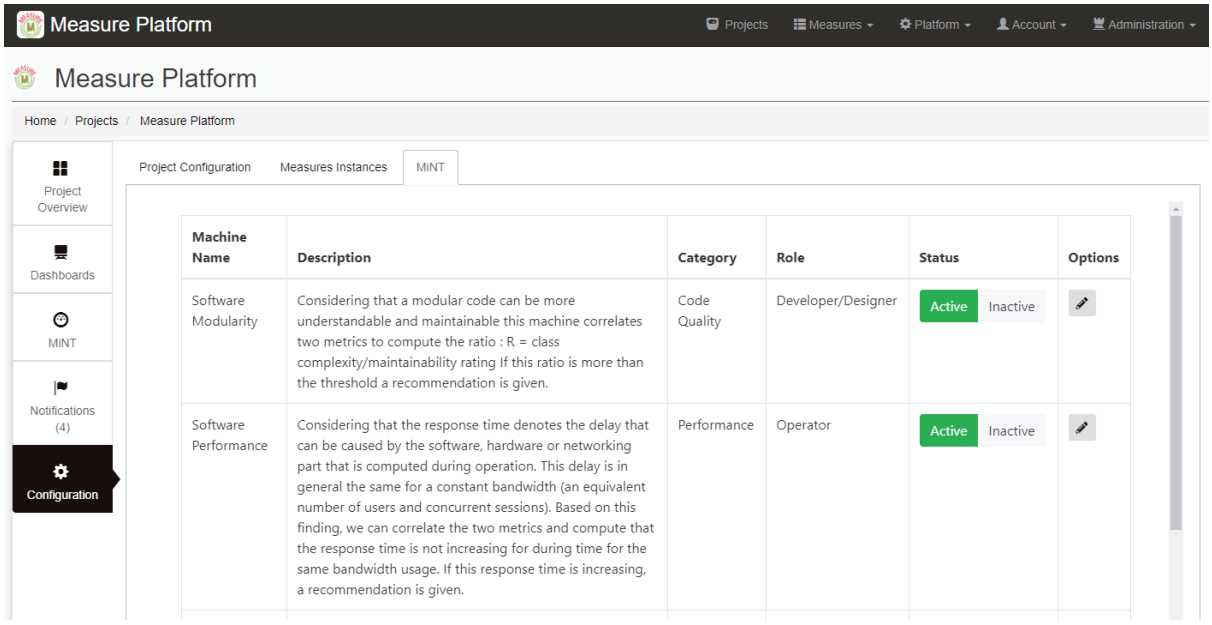
If Mint was added successfully you should see it in the list of available analysis tools with the status in green.



Configuration

Access to the Mint tab of the project page configuration.

There is a table with the available EFSMs (Extended Finite State Machines) displaying name, description, category, role to which the recommendation is guided, status of the machine (Active or Inactive) and options.

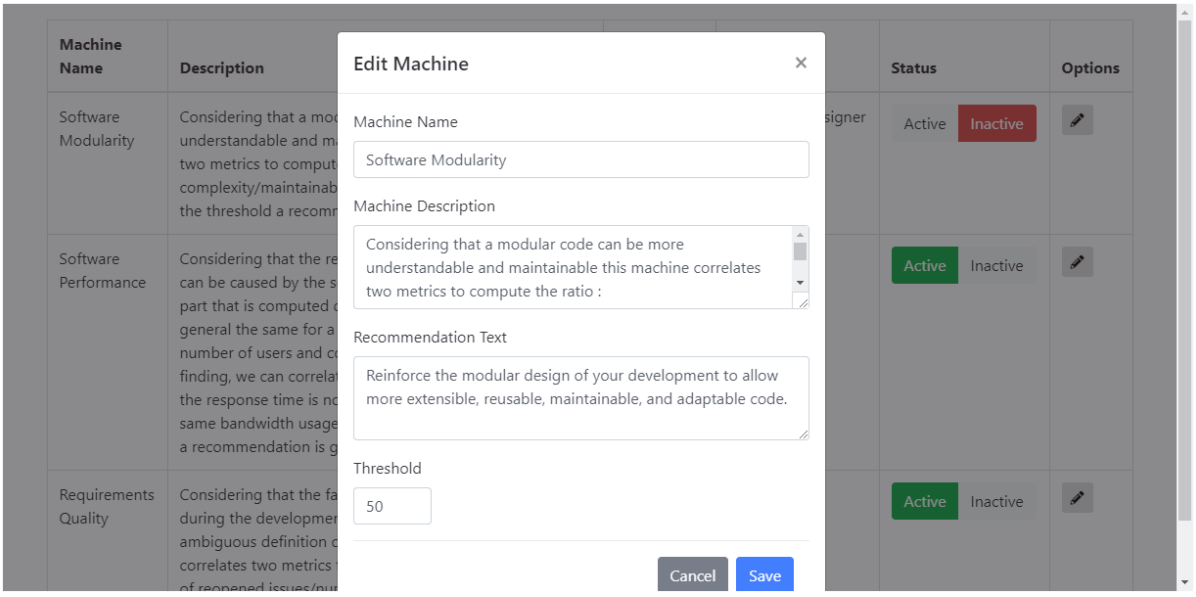


The state of each of the machines can be changed, this determines if the analysis is performed and the corresponding recommendations are received.

Machine Name	Description	Category	Role	Status	Options
Software Modularity	Considering that a modular code can be more understandable and maintainable this machine correlates two metrics to compute the ratio : R = class complexity/maintainability rating If this ratio is more than the threshold a recommendation is given.	Code Quality	Developer/Designer	Active Inactive	
Software Performance	Considering that the response time denotes the delay that can be caused by the software, hardware or networking part that is computed during operation. This delay is in general the same for a constant bandwidth (an equivalent number of users and concurrent sessions). Based on this finding, we can correlate the two metrics and compute that the response time is not increasing for during time for the same bandwidth usage. If this response time is increasing, a recommendation is given.	Performance	Operator	Active Inactive	

The name, description and text of the recommendation can also be modified, as well as the threshold value (if applicable) as required.

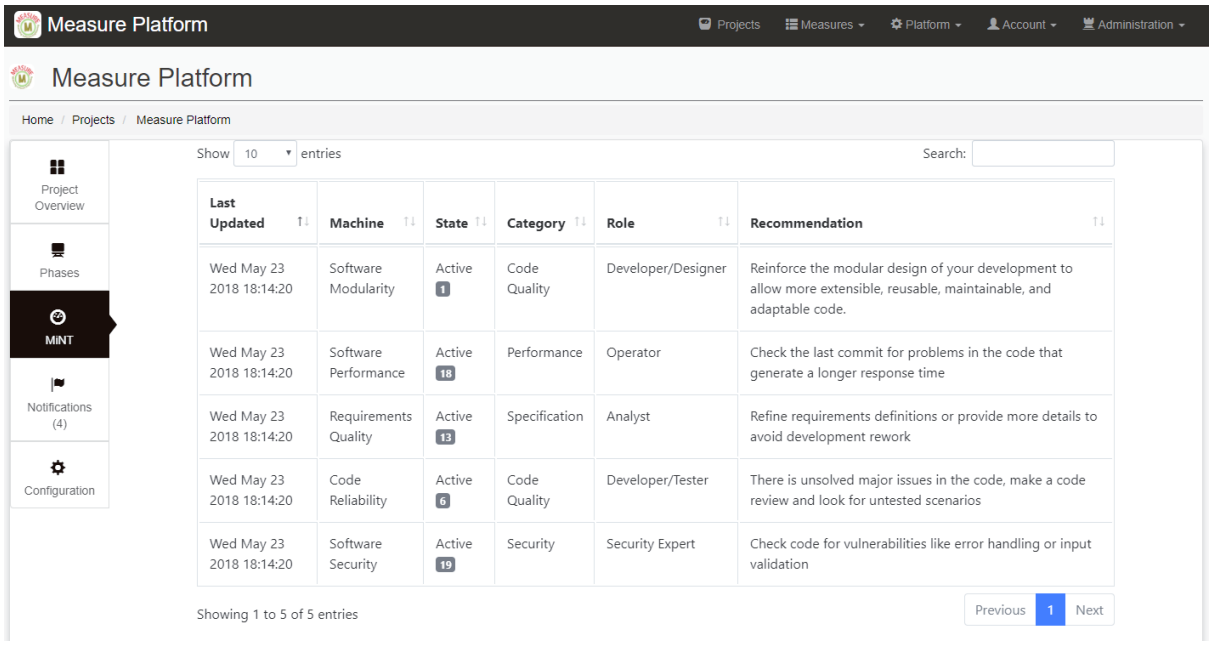
These changes are only applied to the project where the modifications are being made and does not affect the rest.



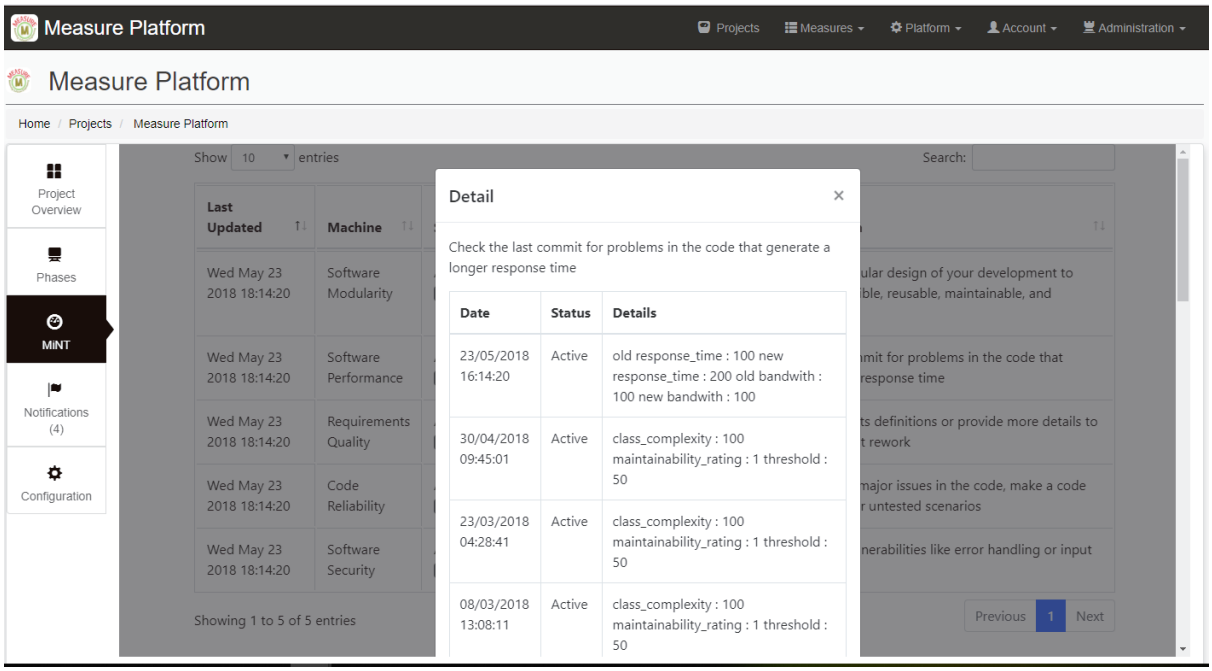
Visualization

The list of recommendations can be accessed from the Mint page within the project.

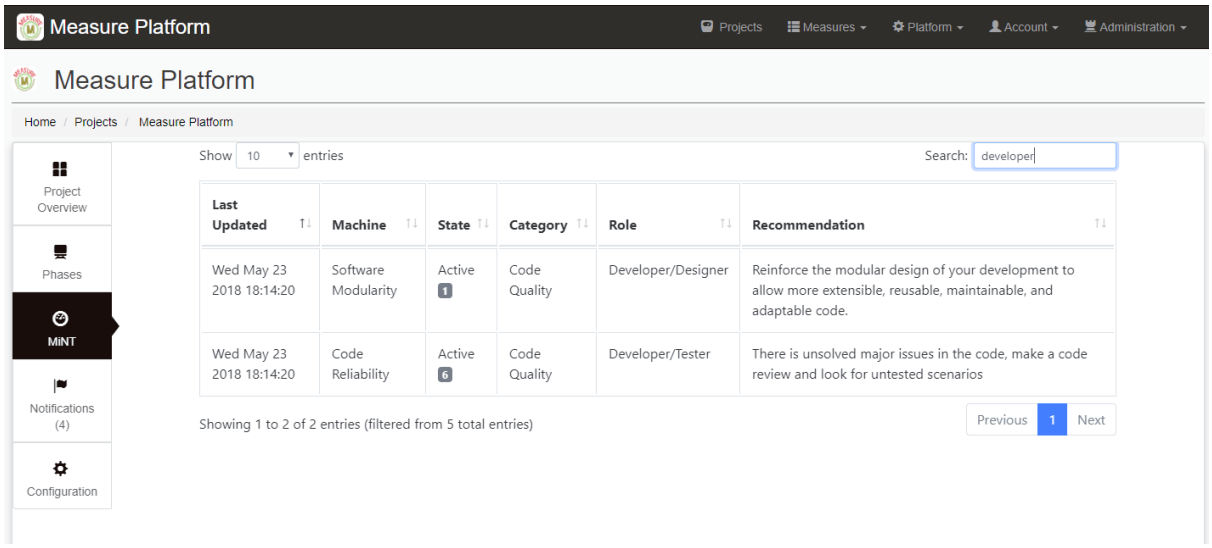
The recommendations are found in a table sorted by date, with the columns last updated, machine name, status of the recommendation (and number of recommendations made), category, role to which the recommendation is directed and recommendation text.



By clicking on any of the recommendations, a model is displayed with the details of each of the recommendations made: date and time, status and details.



The recommendations table can be ordered by any of its columns and the results can also be filtered by searching for some text



Business added value

This tool contributes to improve software quality development identifying and designing correlations between metrics and providing recommendations that help developers to take actions and decisions about the development process. The proposed correlations cover all aspects of the system like functional behavior, security, green computing, and timing. For instance, correlations covering different phases of development and correlations of two metrics from the same development phase at different times.

3.3. Metrics Suggester (IMT)

Description

The Metrics Suggester provides a framework to automatize the suggestion of software metrics based on an initial measurement plan. To do so, the framework needs an initial configuration from the user to determine the metrics range to be analysed and the classifier. This framework is available as web application written in Python and using the Python Scikit-learn library for the analysis process.

This tool is based on learning techniques, The SVM (Support Vector Machine) algorithm for the classification or in other words the analysis of the measurements and the RFE (Recursive Feature Elimination) one for the suggestion of a new measurement plan.

This application is integrated to the Measure Platform as the analysis tool: SuggesterTool, by using the MEASURE platform analysis tool integration protocol (MeasurePlatform/Analysis-Tool-Integration).

Installation

The source code is freely available on its GitHub repository: https://github.com/jjhp02/suggester_tool. It has been released according to the GNU General Public License. It is also available already deployed and running on the server http://157.159.233.70/tool/.

The framework is built as shown in the above figure and the deployment steps of the tool in another server is available in the wiki of the GitHub repository.

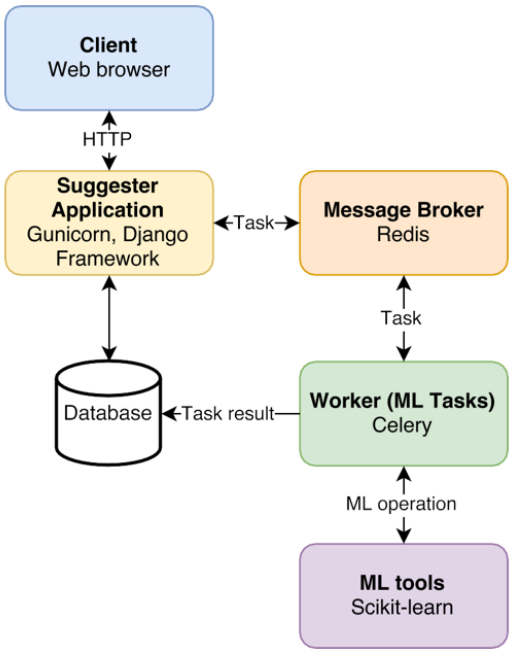


Figure 4: Tool diagram displaying the parts the tool is built from and how each part is connected with the other parts.

Configuration

To do the suggestion, the framework needs an initial configuration from the user to determine the metrics and classifier. The whole process consists of the following steps:

1. Configuration of the initial classifier and the tool. The user must provide the necessary inputs to train the initial classifier, the current measurement plan and how often the suggestion task must be executed.
2. With the initial configuration, our tool can start the automated process of classifying new data that was accumulated during the defined period. The classification is done with the classifier of the tool, and SVM.
3. The result of the classification process is then used as input for the feature selection process. During this process the class that appeared the most during the classification is identify and the relevant features for that class are determined.
4. The relevant features are then used to suggest changes in the measurement plan. This process, based on the relationships between metrics and features, finds a new metric set for a new measurement plan. The new plan is then suggested to the user.

Each step saves its results in a database shared between all the processes of the framework.

Processing

The analysis and the suggestion are automatically launched when the analysis file is uploaded. Then a new measurement plan is created and added to the list of the last generated measurement plans.

Visualization

Metrics Suggester has its own interface allowing to maintain the tool, configure and upload multiple measurements plans or to upload diverse (training) datasets to configure the learned models.

The tool is also embedded in an iFrame to the MEASURE platform (see the Figure below). This frame allows to visualize the collected measures that are analysed. The model is defined in 'Classifiers', the measurements plans suggested and the different detailed suggestions are also illustrated into the frame.

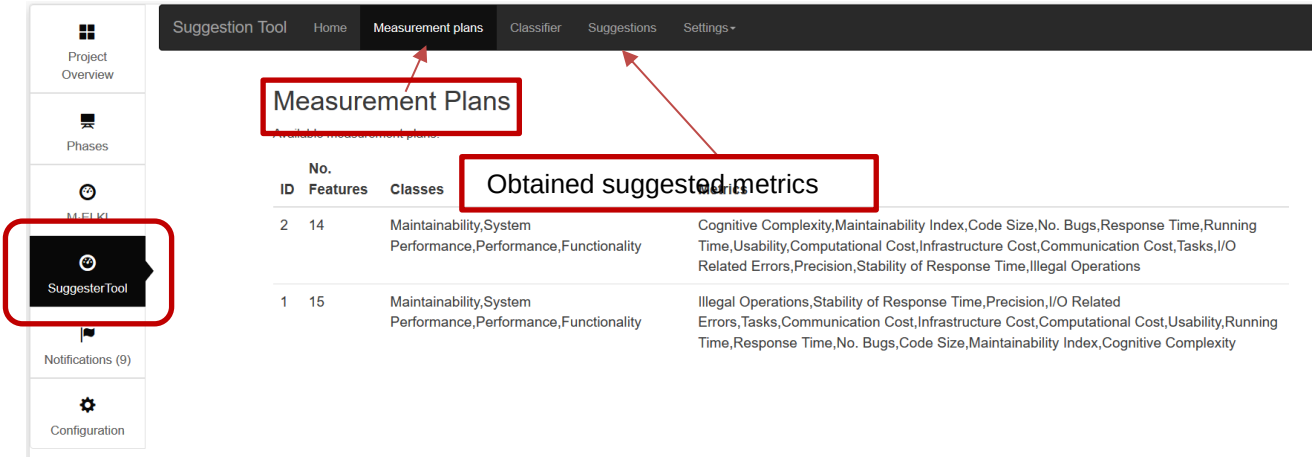


Figure 5- The Metrics Suggester tool embedded in the MEASURE platform

Business added value

Metrics Suggester is an interesting tool allowing to determine (based on the analysis of collected data) and visualise relevant measures, metrics and measurement plans during a specific period of time. This tool is first expected to be used during courses at Telecom SudParis but also to be enriched by other probes, learning techniques (unsupervised) and applied through other diverse metrics (e.g., emotional). IMT also aims to make it sustainable for measuring concrete activities at Telecom SudParis.

M·ELKI (ICAM)

Description

M·ELKI is a set of web services that makes possible to select, configure, process and visualize results of 4 clustering algorithms provided from the ELKI Java library (https://elki-project.github.io). This library is a state-of-the-art and clustering-focused library in Java. It has been chosen because of these 3 features. It has been chosen instead of the Weka library (https://www.cs.waikato.ac.nz/ml/weka) because its integration within web services has been proved easier than Weka, its memory footprint and its processing efficiency better than Weka[1]. The 4 selected algorithms are drawn out from 4 different clustering algorithm families:

1. DBSCAN, a density-based clustering algorithm
2. KMEANS, a centroid-based clustering algorithm
3. EM, a distribution-based clustering algorithm
4. SLINK, a connectivity-based clustering algorithm

1 Hans-Peter Kriegel, Erich Schubert, Arthur Zimek: *The (black) art of runtime evaluation: Are we comparing algorithms or implementations?* In Knowledge and Information Systems 2016, DOI: 10.1007/s10115-016-1004-2 (https://elki-project.github.io/benchmarking)

This set of web services are dedicated to the MEASURE platform. In fact, these web services are designed to closely fit the MEASURE analysis tool integration protocol (MeasurePlatform/Analysis-Tool-Integration). Hence the name: M·ELKI.

Installation

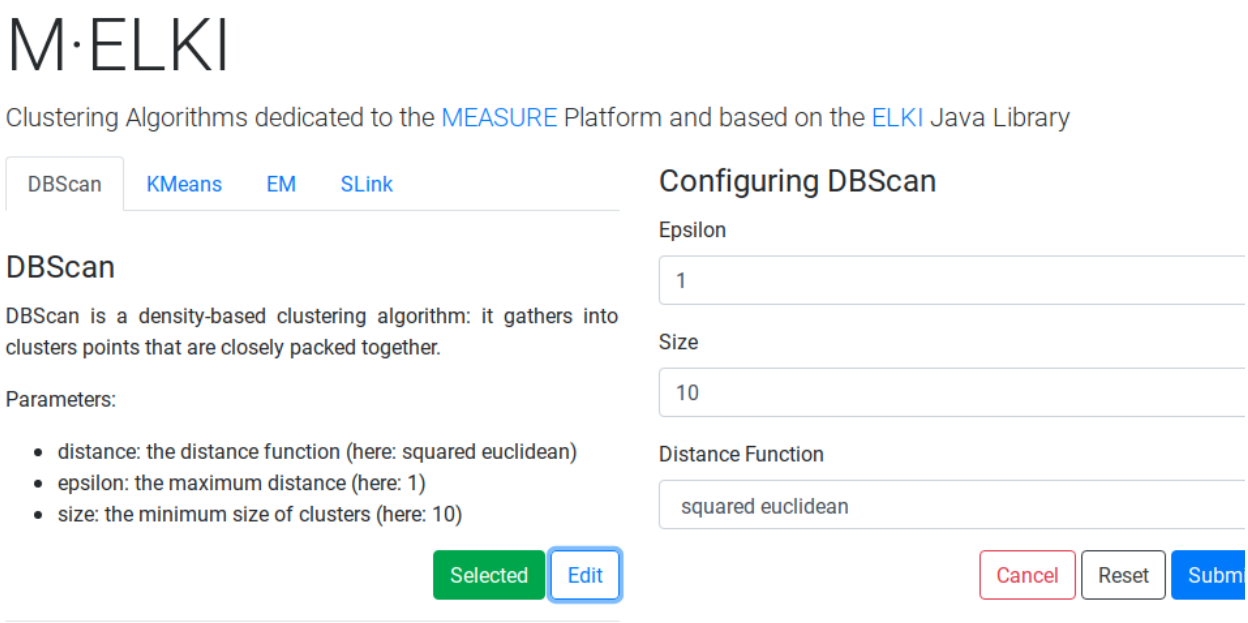
M·ELKI source code is freely available on its GitHub repository: https://github.com/JeromeRocheteau/m-elki. It has been released according to the Apache 2,0 license.

The default M·ELKI instance has been deployed and is running on the server http://app.icam.fr/elki. Thus, there is no need to compile and install it in order to use it within the MEASURE platform. However, instances of M·ELKI can be deployed on other servers as follows:

1. install a JEE application container (e.g. Tomcat, Jetty, TomEE, GlassFish, JBoss, WebSphere, etc)
2. deploy the M·ELKI web archive (https://github.com/JeromeRocheteau/m-elki/raw/master/elki.war) onto the JEE application container,

Configuration

The M·ELKI configuration pannel consists in 4 tabs that correspond to the 4 clustering algorithms. Each tab describes its related algorithm and its parameter. It makes possible to select the current algorithm against the others. It also possible to define parameter values for every algorithm as illustrated by the Illustration.



Every algorithm should be parametrized by a list of measures. It then makes possible to subscribe to notifications about these measures to the MEASURE platform.

A demonstration in available in French on YouTube: https://youtu.be/yBQHTcdeziQ.

Processing

M·ELKI algorithms are automatically launched using according to the configuration previously defined.

Visualization

Result visualization of the M·ELKI analysis tool are not available currently. However, the ELKI library provides facilities to export results as images in SVG format that can easily be embedded in the MEASURE platform as iFrames.

Business added value

M·ELKI provides state-of-the-art and efficient clustering algorithms to the MEASURE platform. In fact, these are basic algorithms for datamining and bigdata processing. This analysis tool can easily be extended to other algorithms that belong to the ELKI library that same way as the 4 selected algorithms.

3.4. STRACKER (UNIBUC)

Description

STRACKER is a web application that aims to increase the quality of software development by tracking and suggesting (thus, the acronym STRACKER) values of various software metrics during the software development process. More precisely, it helps you see the status of the metric values using different charts, and also shows scores assigned to each new record. It also includes a module that predicts future metric values based on values recorded so far.

In the future, new metrics, an increase in prediction accuracy and new algorithms for metrics correlation are planned.

Installation

If you want to run the standalone version of the Stracker tool, you need to install the following:

- **Python3** https://www.python.org/downloads/ + libraries (flask, pygal, numpy, pandas, matplotlib, sklearn, statsmodels)
- **Elasticsearch** https://www.elastic.co/downloads/elasticsearch
- **Sonarqube** https://www.sonarqube.org/
- Download **Stracker** from GitHub https://github.com/CostiCTI/Stracker

Steps:

- Start Elasticsearch

- Start Sonar

- In Stracker folder, run app.py (*python app.py*)

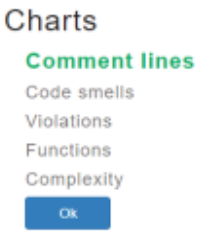
- Application is running on http://localhost:5000

How to use the tool

1. Track software metrics values

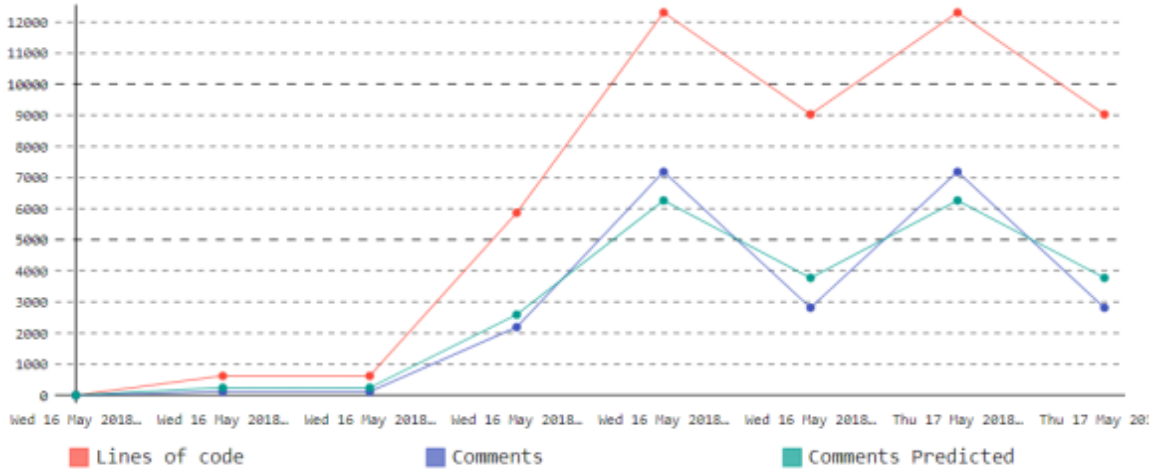
Select a metric that you want to see.

Let us say that we want to see the comment lines metric. We select the comment lines option (option that matches the desired metric) and click ok button.



Then, you can see two charts on the page (the charts are created using pygal library).

First chart may look like the one below:

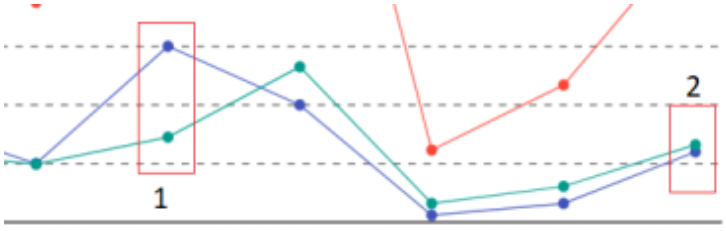


The red line shows the number of lines of code of the project.

The blue line shows the number of comment lines of the project.

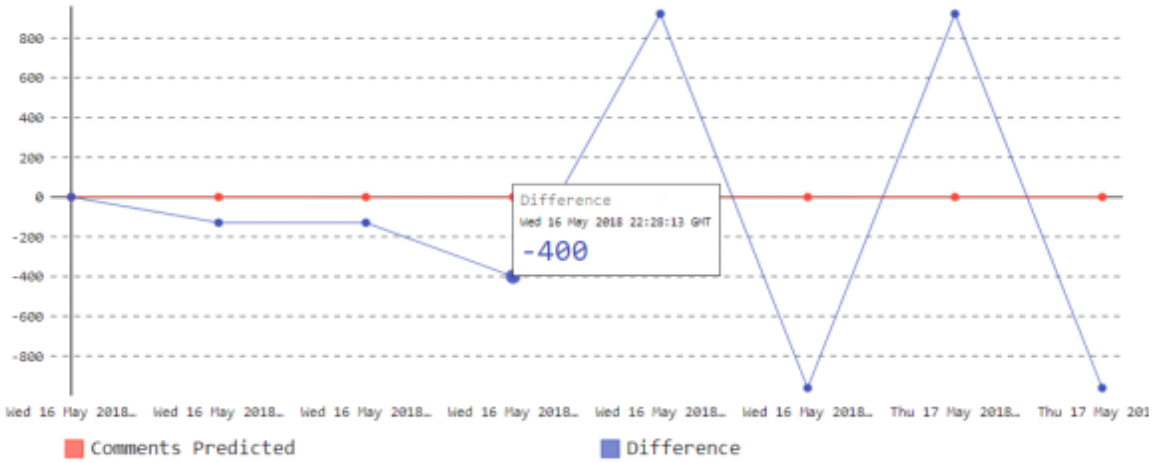
The green line shows the predicted number of comment lines of the project.

The scope of the chart is to show to a software project manager the difference between the current metric (blue line) and the predicted metric (the green line). Since the prediction is based on a model taking into account many software projects, the predicted value is, in a sense, showing the „best practices".



Zooming in (see figure above), we notice that the red box 1 shows a situation when the difference is rather big whereas in red box 2, the metric values and predicted values are almost similar.

The second chart looks uses the predicted metric as a baseline:

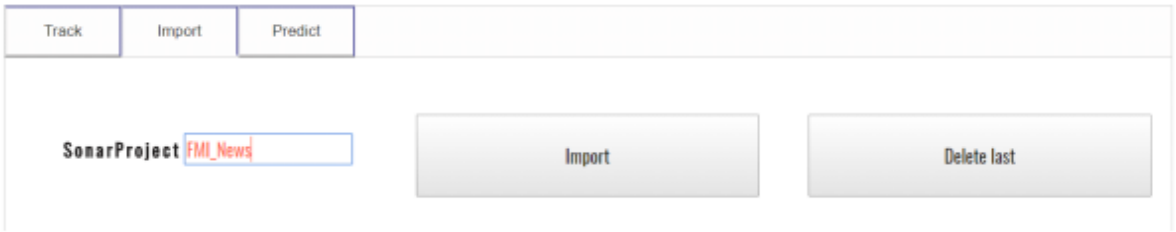


Here we can more clearly see the difference between the number of predicted comment lines and the number of comment lines of the project.

Also, if we put the cursor over a point we can see the exact value of that point.

2. Import new metrics values

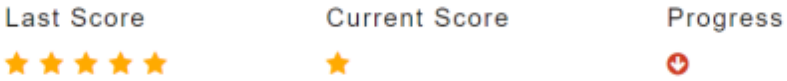
In order to add a new project to the current project, you need to upload your project in SonarQube and then Stracker will make the import from there.



Go to “Import tag”. Write the project name from SonarQube and click Import button. Now your new record is added.

Also, here you can delete your last record by clicking the Delete last button.

Also, you can see if your new record is better or not than the last one, by looking at the score.



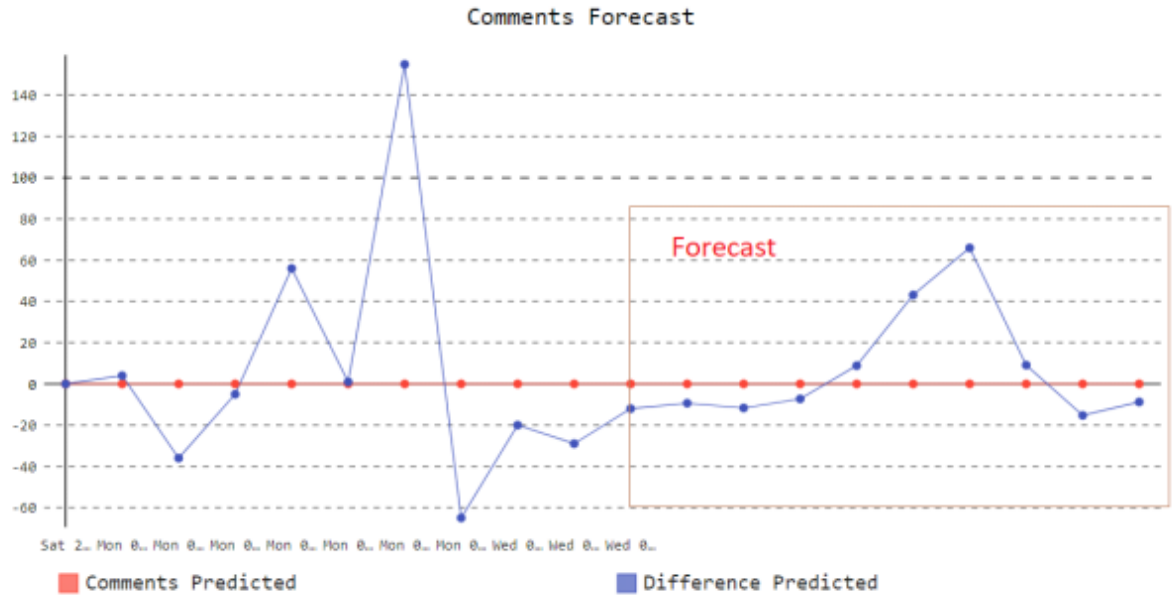
Current score represents the score of the last added value, calculated based on the difference between our metric value and predicted value.

Last score is the previously added score.

Progress is positive if the current score is better than the previous one, and negative otherwise.

3. Forecast software metric values

We can also see a graph with forecasted values of our project metrics data.



Thus, you can get the possible values of your metric in the future (using time series prediction algorithms).

Business added value

There are two important features of the Stracker tool that may be valuable to a software project manager.

First, she can compare various values of the metrics with “expected” values from a “best practices” point of view. Large deviations may raise some alarms (we will implement this feature in the future) and check the situations where the difference is large.

Second, if she receives predicted values for her metrics, she can better plan the project (e.g. if the predicted number of tests is high you may increase the man-power from testing point of view).

4. Conclusion

This document presented, on the one hand, the MEASURE Analysis Platform services and, on the other hand, the analysis tools that have already been integrated into the MEASURE Analysis Platform. These analysis tools correspond as well as to state-of-the-art analysis algorithms – constraint-based filtering with SOFTEAM's Quality Guard and clustering with ICAM's MELKI – as innovative analysis such as the MINT correlation tool co-developed by Montimage and IMT, the IMT's Metrics Suggester and University of Bacarest's STACKER. It also makes possible to enrich this analysis platform by other tools thanks to the MEASURE Analysis Platform HTTP API.

The set of integrated analysis tools will increase before the end of the project in M39. For instance, Turkish consortium analysis tools are planned to integrate the MEASURE Analysis Platform at M35, Moreover, the 5 already integrated analysis tools will be continuously improved until the end of the project.