ITEA Office
High Tech Campus 69 - 3
5656 AG Eindhoven
The Netherlands

T + 31 88 003 6136
E info@itea3.org
W www.itea3.org

ITEA 3 is a EUREKA strategic ICT cluster programme

D4.3: Assessment of improvement areas and countermeasures

MEASURE

Executive summary

The objective of WP4 is to design and develop a platform to analyse the large amount of measurement data generated by the use cases carried out in WP5. The data analysis platform will implement analytics algorithms, to correlate the different phases of software development and perform the tracking of metrics and their value.

This deliverable D4.3 describes the assessment of improvement areas and countermeasures provided by the data analysis algorithms and supporting tools that are required to identify correlations.

The deliverable is rather concise and describes basically an assessment of the tools (self-assessment and assessment by the industrial partners) and future plans for the MEASURE analysis tools: Quality Guard is described by Softeam, rule-based correlation used in MINT tool developed by IMT and relying MMT-Correlator developed by MTI are described by Montimage. The clustering provided by the M-ELKI analysis tool is described by ICAM. Machine learning provided by Metrics Suggester tool is described by IMT and finally, Stracker is presented by the University of Bucharest.

Table of Contents

1. Introduction

1.1. Role of this deliverable

This deliverable D4.3 describes the assessment of improvement areas and countermeasures provided by the data analysis algorithms and supporting tools. It contains mostly the development plans of the analysis tools.

1.2. Relationship with others MEASURE deliverables

This deliverable is linked to Deliverable D4.1, where we describe how the MEASURE analysis tools work, providing a description, installation procedure, configurations, processing, visualization, and business added value.

This deliverable is linked to D4.2, where we describe the analysis algorithms that we used and implemented in the analysis tools.

It is also related to D3.3, we provide documentation and user guidelines for the MEASURE tooling, including the analysis tools.

Finally, it is linked to D5.5, in which the industrial case study providers evaluate and assess the MEASURE tooling, including the analysis tools.

Given fact that the above deliverables provide plenty of technical information about the analysis tools in MEASURE, in this deliverable we focus on (according to the title of the deliverables), “areas of improvement and countermeasure”, i.e., how the tools will evolve from a release point of view in the next period in order to cover for missing functionalities, new features, current weaknesses, integration with other tools and frameworks.

1.3. Contributors

MTI: participated in the description of MINT tool.

IMT: contributed by the algorithms used in Metrics Suggester tool and also participated in the rule-based correlation algorithms used in MINT tool

SOFTEAM: contributed to the description of the Quality Guard tool. Assisted the deliverable editor in the coordination for the deliverable.

ICAM: contributed to the description of the M-ELKI analysis tool

UniBuc: coordinated this deliverable. It also contributed to the description of the Stracker tool.

1.4. Summary table for tool usage by case study providers

In this deliverable, since it is part of WP4, we will look at the analysis tools. However, they are part of a large ecosystem of tools, including metrics collectors, externat tools, as well as analysis tools. They were all evaluated by at least industrial partner and the results were reported in D5.5.

Below we provide a table which shows the evaluation matrix of tools and industrial case study providers (an x means the tool was evaluated in the context of the case study).

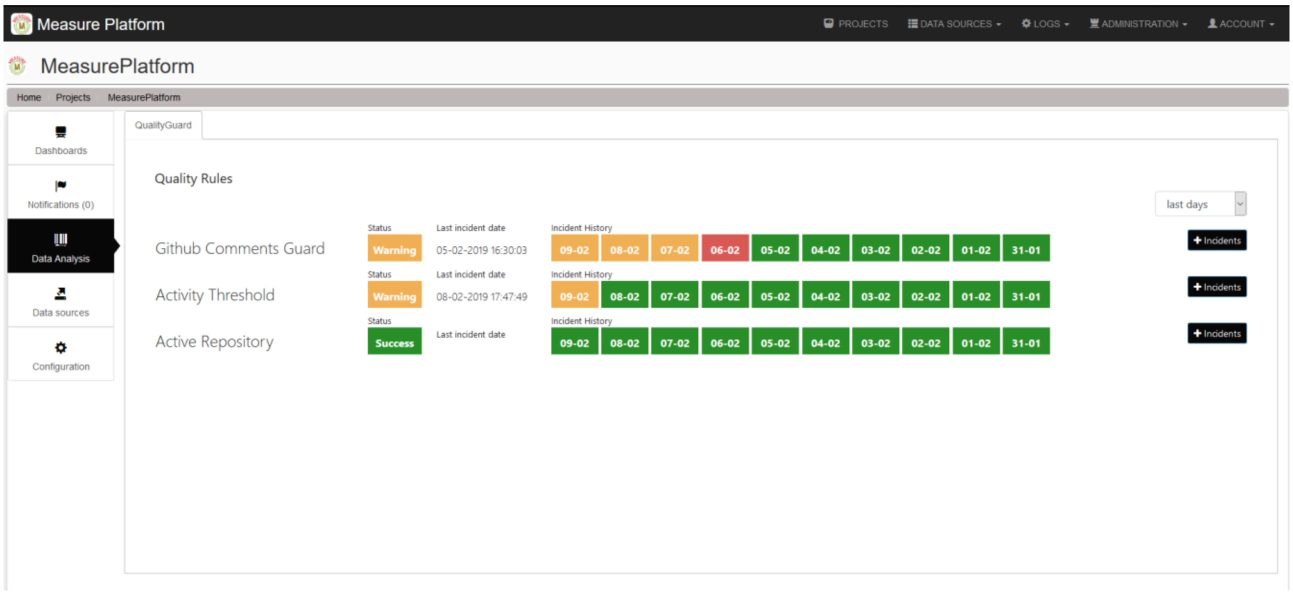
Tool type	List of tools	Softeam	Naval	Bitdefender	Ericsson+T-mob
Metrics collectors	MMT			x	
	Hawk	x	x		
	Test generator				x
	Emit	x			
	River			x	
External tools integrated in the MEASURE platform	SonarQube	x	x	x	
	SVN	x	x		
	Git	x		x	
	Mantis	x			
	Jenkins	x			
	Others	x			
Metric Analysis Tools	Quality Guard	x	x		
	Metric correlation: Mint			x	
	Metric suggester	x			
	Stracker	x		x	
	M-ELKI	x			

2. MEASURE Analysis Tools

2.1. Quality Guard

The Quality Guard analysis tool is an extension of the Measure Platform which provides mechanisms for the advance data analysis functionalities applied to data produced by the continuous measurement applied by the platform.

The main purpose of the Quality Guard tool is to provide a simple way to monitor in real time the data collected by the platform and to raise alerts and trigger-adapted countermeasures if this data drifts beyond a predetermined threshold. The tool is meant to help the everyday work of quality engineers and project managers and facilitate the communication between them. Quality engineers will be able to define constraints based on measure thresholds related to a specific project and *projects Managers* will be able to verify the state of each constraint periodically or to get the incidents history of all constraints defined by the Quality Guard Tool.

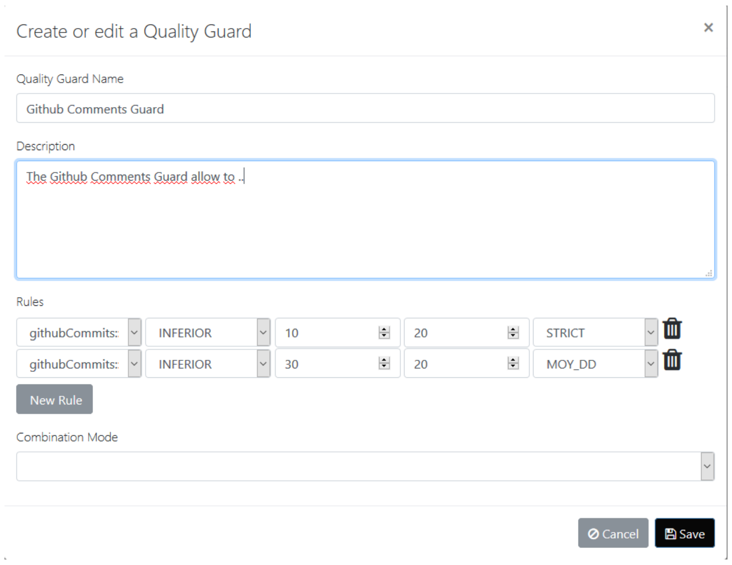


Quality Rules

	Status	Last incident date	Incident History									
Github Comments Guard	Warning	05-02-2019 16:30:03	09-02	08-02	07-02	06-02	05-02	04-02	03-02	02-02	01-02	31-01
Activity Threshold	Warning	08-02-2019 17:47:49	09-02	08-02	07-02	06-02	05-02	04-02	03-02	02-02	01-02	31-01
Active Repository	Success		09-02	08-02	07-02	06-02	05-02	04-02	03-02	02-02	01-02	31-01

Within the MEASURE Quality Guard Tool, a *quality rule* is defined on a Measure project to check that a numeric measure collected on this project stays on delimited range. The rule is composed of multiple Guard Condition which defines a binary condition to compare the measure field value with threshold values.

As a result, a quality guard can be complex and can allow the results of more than one measure to be combined in the same rule. Each violation of these rules is recorded by the tools and may be exploited later to be analyzed or contributed to an audit process. To avoid the triggering of an alarm when an abnormal value is collected, it is possible to configure the tool to make an average of the values collected over a period ranging from one second to one week instead of immediately triggering an alert when data exceeds the configured threshold.



Create or edit a Quality Guard

Quality Guard Name: Github Comments Guard

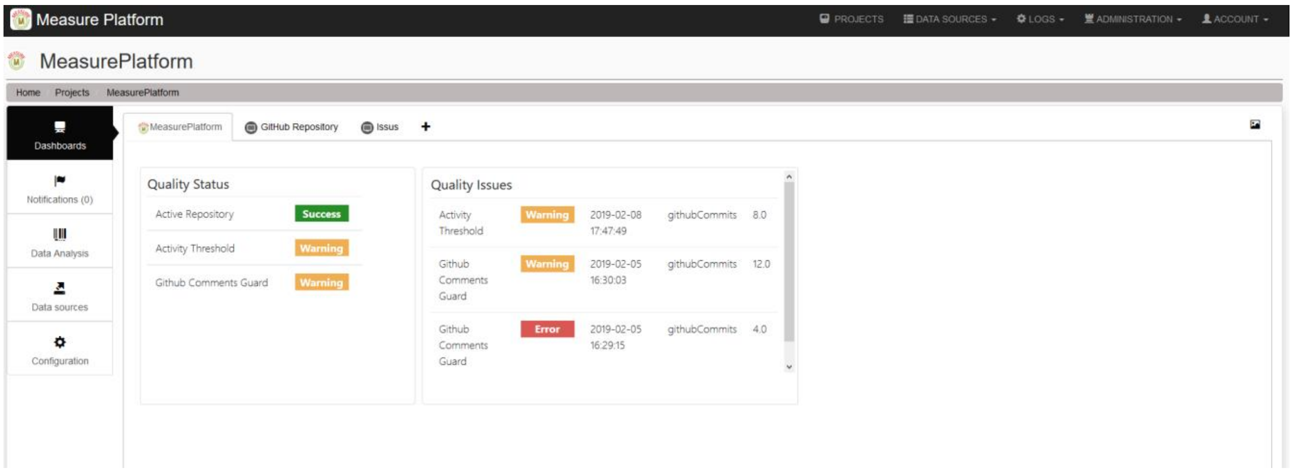
Description: The Github Comments Guard allow to ..

Rules

githubCommits:	INFERIOR	10	20	STRICT
githubCommits:	INFERIOR	30	20	MOY_DD

Conditions and constraints can be based on simple or cross measures expressions and the expression language support the logic operators: “AND”, “OR” and the comparison operators: “>”,”<”. In more detail, each guard is defined by a quality guard name field, a description field, a measure instance and measure field name, a guard operator field (Superior or Inferior), a warning value field and an error value field. Constraint violations are evaluated by the tool once a new measurement is collected by the platform and periodically. i.e. average values of measurements collected in a specific interval. A rational must be specify which all quality rules to help project manager to understand the detected issue and consequently help him to take the appropriate mitigation action.

The main view provided by the Quality Guard Analysis tool allows to visualize the state of each constraints defined by the tool. For each constraint, the Quality Guard Analysis tool allows to visualize a history of the last incidents.



Quality Status

Active Repository	Success
Activity Threshold	Warning
Github Comments Guard	Warning

Quality Issues

Activity Threshold	Warning	2019-02-08 17:47:49	githubCommits	8.0
Github Comments Guard	Warning	2019-02-05 16:30:03	githubCommits	12.0
Github Comments Guard	Error	2019-02-05 16:29:15	githubCommits	4.0

The tool can also show a dashboard card that lists the last constraints violations that occurred in the monitored project. The dashboard cards provided by the Quality Guard Analysis tool allow to get an overview of the state of either the quality guards or the constraints violations occurred in each project.

2.1.1. Current state and areas of improvement

State of the tool and Integration with the Measure Platform

The current release of the Quality Guard tool is fully functional and does not have any major bugs detected that could prevent its deployment. The tool implements all the services that have been specified at the start of project.

Regarding the packaging, two versions of the tools are now available and can be downloaded on the Forge Quality Guard project, one for Linux related platforms and one for Windows. In addition, to facilitate the deployment and use of the tools, an *Installation Guide* and a *User Guide* are available in the packaging.

To protect the rights of the tool developer (i.e., SOFTEAM) on the tools and in anticipation of its future commercial exploitation, the Quality Guard analysis tool is released under the GNU GPL 3.0 licence: https://opensource.org/licenses/GPL-3.0

At this time, the Quality Guard Analysis tool is fully integrated to the Measure Platform. First, we use the mechanisms put in place by the Measure platform to register and make available the analysis services provided by the Quality Guard to all projects of the platform. This integration involves recording the tool in the platform measure, monitoring the services requests sent by the different projects and this via the REST API provided by the platform for this purpose. Next, the tool provides four views which are directly embedder into the platform web site: a configuration view, a main view which provide analysis results and two small views which can be integrated into projects dashboards.

For performance reasons, the analysis tool accesses the data collected by the platform by establishing a direct connection with the Elasticsearch database. Although the development of a standalone version of the tool is conceivable, the current Quality Guard Analysis tool has been developed specifically as an extension of the data collection platform Measure. It is therefore not currently possible to use it outside the context of the Measure ecosystem.

Self-assessment of the tool by SOFTEAM

Based on the possibility of defining complex analysis rules on the project's data, the quality guard tool provides a good answer to the problem of quickly detecting anomalies in the data of a project. Its function could however be extended to, in addition to raising alerts, trigger automatically mitigation action in case of detections of quality issues.

STRENGTHS	***WEAKNESSES***
Cross measures and cross domain analysis.	*Direct access to the Platform database is required.*
Rich quality rule configuration system including rationales.	*No automation of implementation of mitigation action when an alert is trigged.*
Historization of identified anomalies.	

OPORTUNITIES	***THREATS***
Fully integrated to the Measure Platform. *Facilitate communication between quality engineer and project managers.*	*No standalone packaging, only usable as extension of the Measure platform.*

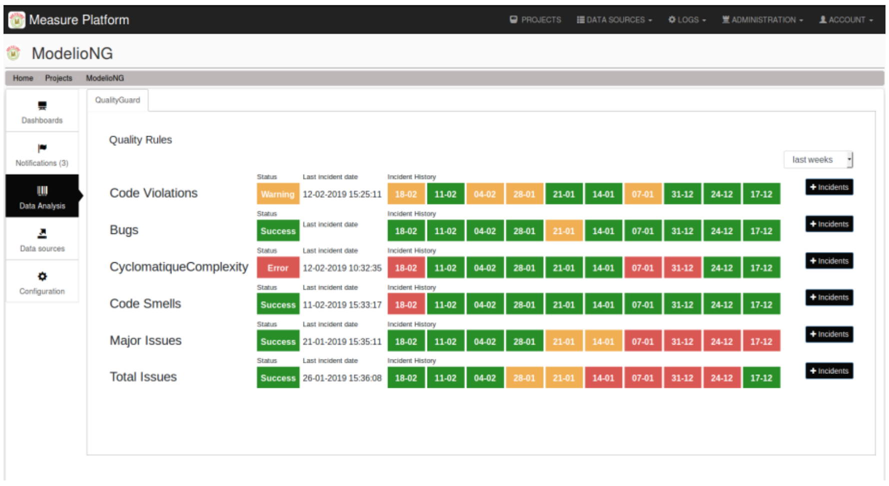
Assessment by the case study providers

At this time, the Quality Guard Analysis tool has been integrated and evaluated in the context of the project by two of the four case study: the Modelio Product Line case study by SOFTEAM and Large Naval Software System environment *case study* by Naval Group.

Assessment by The Modelio Product Line case study (Softeam)

In the second evaluation scenario of the Measure Platform who aimed to evaluate the financial gain caused by the usage of Analysis Tools applied to the development process of the Modelio 3.8 product, Softeam has evaluated the Quality Guard Tool.

Softeam has defined a list of quality rules apply to several metrics collected in context of the evaluation scenario. As expected, Softeam received several notifications of incidents during the analysed period which have covered a period of two months from December 2018 to January 2019. This period corresponded to a validation phase of the components delivered by the development team, so the results showed an increase in the number of anomalies during the validation phase then a decrease in the latter when the developer begins the correction work. These results were considered by Softeam as consistent with what we observed in the conduct of the project.



Quality Rules

	Status	Last incident date	Incident History									
Code Violations	🟧 Warning	12-02-2019 15:25:11	🟧 18-02	🟩 11-02	🟧 04-02	🟧 28-01	🟩 21-01	🟩 14-01	🟧 07-01	🟩 31-12	🟩 24-12	🟩 17-12
Bugs	🟩 Success		🟩 18-02	🟩 11-02	🟩 04-02	🟩 28-01	🟧 21-01	🟩 14-01	🟩 07-01	🟩 31-12	🟩 24-12	🟩 17-12
CyclomatiqueComplexity	🟥 Error	12-02-2019 10:32:35	🟥 18-02	🟩 11-02	🟩 04-02	🟩 28-01	🟩 21-01	🟩 14-01	🟥 07-01	🟥 31-12	🟩 24-12	🟩 17-12
Code Smells	🟩 Success	11-02-2019 15:33:17	🟥 18-02	🟩 11-02	🟩 04-02	🟩 28-01	🟩 21-01	🟩 14-01	🟩 07-01	🟩 31-12	🟩 24-12	🟩 17-12
Major Issues	🟩 Success	21-01-2019 15:35:11	🟩 18-02	🟩 11-02	🟩 04-02	🟩 28-01	🟧 21-01	🟧 14-01	🟥 07-01	🟥 31-12	🟥 24-12	🟥 17-12
Total Issues	🟩 Success	26-01-2019 15:36:08	🟩 18-02	🟩 11-02	🟩 04-02	🟧 28-01	🟧 21-01	🟥 14-01	🟥 07-01	🟥 31-12	🟥 24-12	🟩 17-12

Softeam considered that this tool addresses in a satisfactory manner its requirements in term of quality check apply to monitor data and notification mechanism when a deflect is detected.

Assessment of the Large Naval Software System environment case study by Naval Group

Following the deployment of the tool in our environment, we concluded that the Quality Guard Analysis Tool can be of great interest to monitor metrics, but might need more maturity to release its full potential. Its exploitation is not ergonomic enough to easily handle loads of Quality Guard instances. It also misses an "external trigger" feature to immediately notify developers/other tools of a Quality Guard violation.

Strong points are:

- *Easiness of Deployment*: Even though Naval Group's production environment is very constrained, deploying the tool was very easy and didn't cause much trouble.

- *Simple usage*: Configuring new rules or editing existing ones is made really easy over the MEASURE platform's interface.

Weak points are:

- *Display of analysis results*: Tool output's presentation isn't clear enough to withstand great amount of guards, tool's added value is diluted by an interface on which it is harsh to apprehend a project's global state at a glance.

- *Transmission of results*: Results are only available on the MEASURE platform; this could be improved to better blend into a Continuous Integration environment through the call of external scripts/triggers to launch external processes when a Quality Guard fails (to call other tools, to send information to developers).

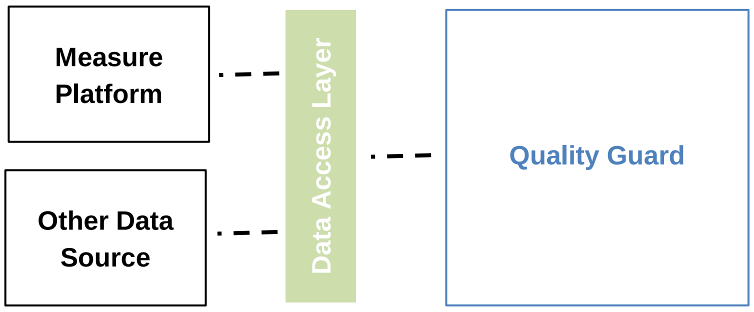
2.1.2. Future development plans

In the near future, we are considering two axes to improve and extend the functionality of the tool: the development of a standalone version able to work with multiple data sources and the establishment of a mechanism allowing automation of the triggering of mitigation actions.

Standalone version of the Quality Guard Tool

For this first release, the Quality Guard tool is intrinsically linked to the Measure platform. The use of this tool independently of the platform, in order to transform it into generic monitoring tool able to work with several data sources, is a possible improvement for the Quality Guard tool. We consider this evolution as a prerequisite for the adoption of the tool by a wider spectrum of users than the clients of the Measure platform.

Although simple to implement, this evolution will require a change of approach related to the architecture of the solution. All of the work required for this evolution is focused on separating the access layer from the data to the monitory to achieve independent components of the main platform that can be specialized for each of the data sources supported.



Standalone architecture of Quality Guard Tool

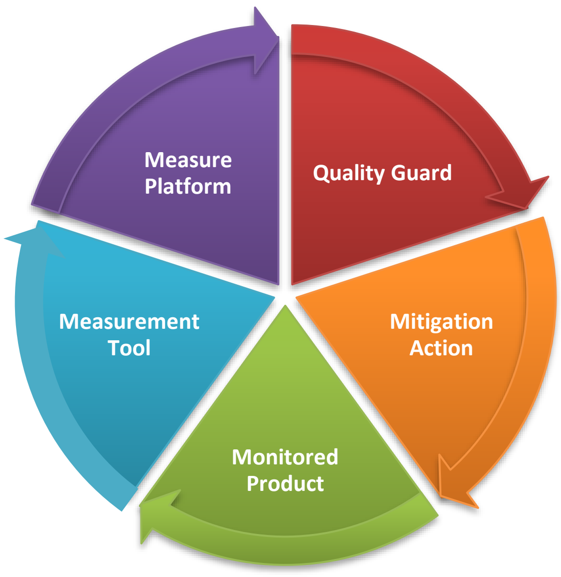
We do not have a roadmap yet for the implementation of this evolution because we are waiting to see how this tool will be welcomed by our customers beyond the perimeter of the Measure project. However, we consider this evolution as important because it would allow us to open a market related to the development of specialized data access components for our customers. However, we consider this evolution as important because it would allow us to open business opportunities related to the development of specialized data access components for our customers.

Feedback loop of product quality improvement

The second change proposed for the Quality Guard tools is the integration of an automatic mechanism for triggering corrective actions when an alert is raised. The underlying objective of this evolution is the establishment of a feedback loop aiming at a constant improvement of the quality of the projects monitored by the platform measure.

The **Product** is monitored by the **Measure Platform** using a **Measurement Tool**. If an issue is detected by the **Quality Guard** tool, a **Mitigation Action** is automatically trigged. An improvement in the quality of the **Product** result from the implementation of this action mitigations and this improvement will be monitored by the Measure Platform.

This mitigation actions will be configured by the quality engineer when this one will work on the specification of the Quality Rules and we plan to offer the possibility of defining several types of corrective actions according to the different configurable fault detection thresholds.



The Measure Platform and Quality Guard feedback loop

We plan to support several types of corrective actions ranging from the integration of new notification rings linked to the company's information system to the automatic generation of corrective tasks towards the project management tools used by our customers.

Here are some examples of mitigation actions:

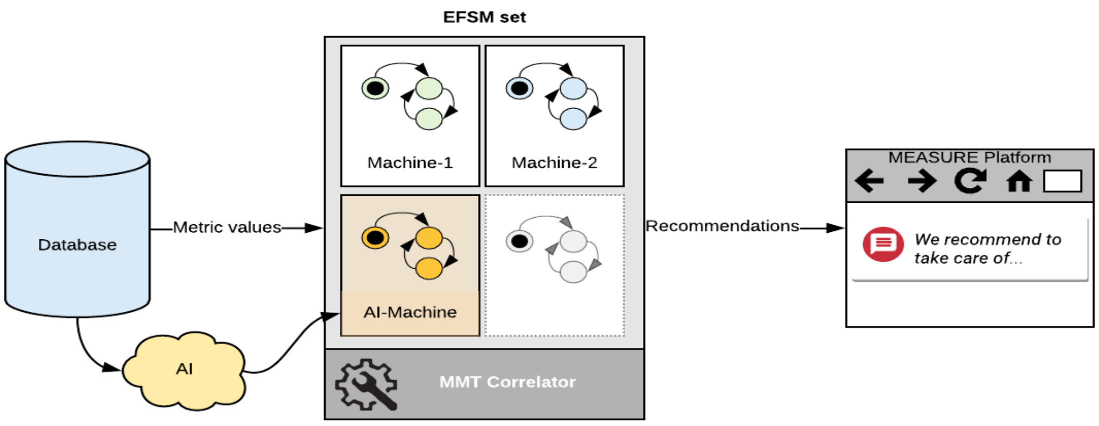
- Send an email to the project manager
- Create a new issue in a bug tracking tool
- Generate a report which will be used in a weekly management meeting
- Create a new task in a project management tool.

This evolution is already integrated in our development roadmap and we plan to release this functionality in the V 2.0 of the Quality Guard analysis tool plan in September 2019. This roadmap will be able to evolve according to the feedback following the release of the product and future uses that we want to do with this tool.

2.2. MINT

Metric correlation is one of the most widely used and widely misunderstood statistical concepts. It refers to a mutual relationship or association between measurements. It is in general useful for predicting one measurement from another.

To improve software quality, it is necessary to introduce new metrics with the required detail and increased expressive power, in order to provide valuable information to the different actors of software development. For this reason, we define an approach based on metrics that contribute to improve software quality development. This approach focuses on the combination, reuse and correlation of metrics. It suggests to the user indications of how to reuse metrics and provide recommendations after the application of metrics correlation.



MINT general concept

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. It considers the different measurements collected by the MEASURE platform as events occurring at runtime. The correlations are designed as extended finite state machines (EFSMs) allowing to perform Complex Event Processing in order to determine the possible actions (recommendations) that can be taken to improve the diverse stages of the software life cycle and thus the global software quality and cost.

2.2.1. Current state and areas of improvement

The basic idea behind MINT approach is to specify a set of correlation rules based on the knowledge of an expert of the software development process. These rules can rely on one or different sets of metrics (seen as inputs) and allow to provide different recommendations (seen as outputs) to different kinds of actors:

Actors from the DevOps team: Analysts, designers, modellers, architects, developers, tester, operators, security experts, etc.

Actors from the management plan: product manager, project manager, responsible of human resources, responsible of financial issues etc.

The tool is automatic and triggers recommendations according to the default correlation rules specified at the beginning. Now the set of correlation rules is small (10 rules) and needs to be extended. The tool is fully integrated to the MEASURE platform and can manage different measurements collected by the different measuring tools. More work is to be done to write new correlation rules or generate them from a previous data set.

The evaluation the tool in the context of MEASURE case studies provided the following output: Mint module is a great idea to derive a measure from at least two measures, it gives many opportunities for experimentation and improvement for subject matter experts who deeply understand what is measured and how to correlate data. In Bitdefender context, Mint can be used to monitor the software development process. The real value of this tool can be achieved collaborating with many other Software Engineering Managers in order to derive enough rules. This can be achieved in the Software Development community meetings. There is also a way to improve this module in the next releases by providing feedback on different metrics value at certain points and integrate user feedback in some machine learning algorithms in order to classify future statuses.

2.2.2. Future development plans

Extension	**Planned date**	**Responsible**	**Context**
New design of correlation rules (reach the threshold of 50 rules)	Q2 of 2019	Montimage	Pre-exploitation
New design of correlation rules (reach the threshold of 100 rules)	Q3 of 2019	Montimage	Pre-exploitation
Industrialization of MINT product	Q4 of 2019	Montimage	Direct exploitation
Automatic refinement of correlation rules (and mainly threshold for measurements) by applying AI algorithms	Q1 2020	Montimage	New research project
Automatic generation of correlation rule from previous projects and experimentations by applying AI algorithms	Q1 2021	Montimage	New research project

2.3. Metric Suggester

The Metric Suggester tool is a tool for analysis software measurement. The aim is to automatically analyse a large number of measurement results of a defined number of metrics in order to suggest a new measurement plan better adapted to the needs of the software. This suggestion is actually a prioritization of the metrics at a time *t* of the software development cycle. The purpose of the prioritization is to reduce the cost of a continuous static measurement on software properties that are of no interest at time t and thus reduce the huge amount of data which are unnecessarily collected and analyzed.

Our approach is based on three procedures:

- The initialization of the measurement plan by an expert,
- The analysis of measurements data through the supervised classification algorithm SVM,
- And the suggestion of novel measurement plans.

2.3.1. Current state and areas of improvement

This Metric Suggester trains a classifier according to a defined measurement plan which is characterized by: the executed metrics, the software properties observed by these metrics and the link between both. Then, it analyses the measurement data through a trained classifier to suggest a new measurement plan that is more efficient. Therefore, the tool is very valuable to reduce the energy and cost in gathering the metrics from different software lifecycle phases as it may reduce the number of the collected metrics according to the needs. It uses the support vector machine (SVM) algorithm, which allows to build different classiffcations and provide the relevant measuring plan.

Our analysis and suggestion tool is built as a web application. The architecture is organized around the machine learning unit (ML tool), which regroups the classification and feature selection algorithms. The first one is used to train the classifier, through the training file, and then to analyze the data by classifying it according to the trained classifier; the second one is used to determine the necessary features (herein metrics) to the classification. We use the latter to determine dynamically the mandatory metrics for the next analysis. The library used to develop the learning algorithms is *scikit-learn*. Its implementation has been integrated in the MEASURE industrial platform as an analysis tool.

The Metrics Suggester tool collects the data from the platform for the analysis, then return the results to the platform as dashboard.

However, this approach is still dependent to the expert for the initialization step, especially for the elaboration of the training file. As a reminder, this file is used to train the classifier and it defines the correlation between vectors and classes. So, the analysis model is manually done. Thus, the cost time of this step is high when the samples to classify are numerous.

In order to tackke this problem, we propose to use an unsupervised learning algorithm to generate automatically an analysis model. From an unlabeled software measurements sample, a labeled one is generated. This output is then used as training file to train the classifier used for the analysis and suggestion steps. The purpose is to use a clustering algorithm. It will group in clusters the similar vectors of measurements then according to the clustering result, the expert will associate to each cluster a set of metrics to suggest. Herein, the expert intervention only appears to determine the correlation between classes corresponding to the clusters and set of metrics.

Our improved suggestion approach is based on three procedures:

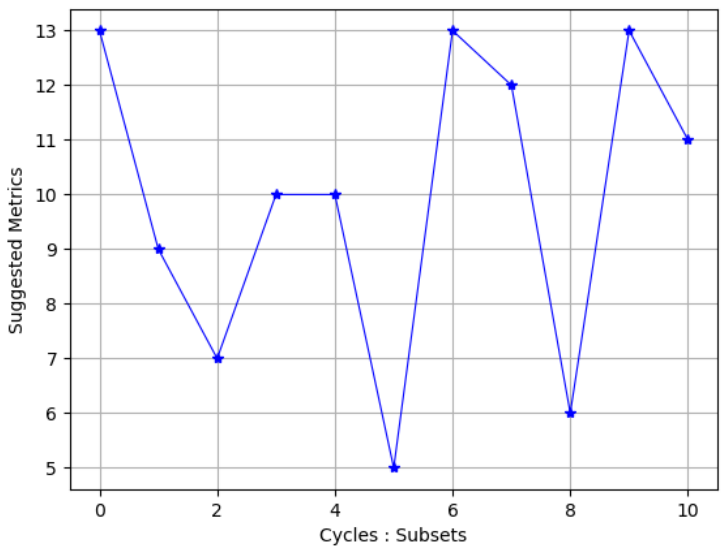
- The elaboration, through software measurements clustering based on unsupervised learning approach X-MEANS, of an analysis model (or mapping system as called in our previous work), herein considered as the initial measurement plan *mp*.
- An analysis procedure, which aims to highlight a software property of interest through a software metrics classification, based on a learning technique, herein SVM.

A suggestion of metrics based on the *mp*, the analysis result and the features selection procedure, which aim to determine the needed metrics, for the conservation of information and based on the learning technique RFE.

Assessment by the Modelio Product Line case study (Softeam)

In context of the final evaluation, Softeam has experimented the usage of the Metric Suggester Tool to define better measurements plan. Collecting a large number of metrics related to the code quality of its Modelio product is rather challenging and resource-intensive.

Softeam sought, to be more precise, the goal was to identify a reduced list of metrics that are the most relevant among the 20 metrics relating to the quality of the code that were collected. With the help of the developer of the tool, Softeam was able to follow the three steps required to set up a new measurement plan: the initialization of the measurement plan, the analysis of measurements data through the supervised classification algorithm SVM and finally the suggestion of novel measurement plans.



Metric suggestions results

The analysed data are data from evaluations history on the software corresponding to specific events on the code. As the above figure is showing, different suggestions were proposed with different number of metrics in the suggested measurement plan.

2.3.2. Future development plans

Although showing very good potential, there are several aspects that need to be improved in the Metric Suggester tool in the following period. We list them below:

- Experiment with several other machine learning algorithms, choosing the best one depending on the context.
- Improve the user interface based on feedback from the end users.
- Investigate the results on several benchmarks of different groups and types of software metrics in order to identify best-performing scenarios (those can be later use to promote the tool in specific domains).
- Allow the user to better integrate the results of the new measurement plan back into the software development process.
- Integrate to the Measure platform the improved approach by collecting directly the data from the platform for the elaboration of the analysis model.
- Generate an activation or deactivation of the execution of measure on the platform according to the suggestion.

2.4. M-ELKI

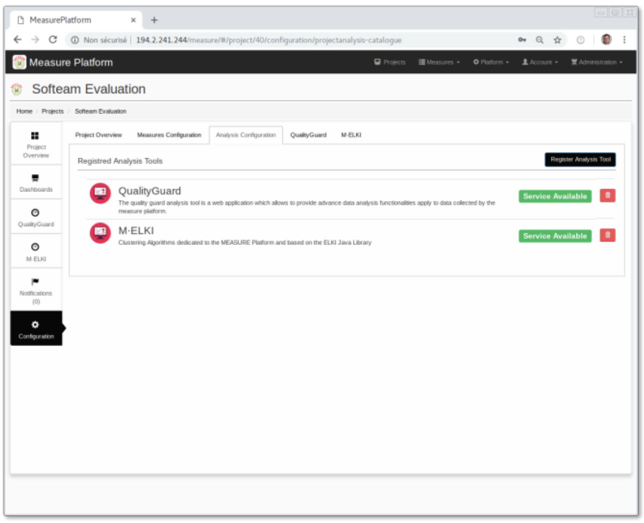
M·ELKI corresponds to an analysis tool that makes possible to perform clustering provided by the algorithms of the ELKI data mining framework in Java.

2.4.1. Current state and areas of improvement

The current development state provides a proof-of-concept about external library embedding into the MEASURE Platform. The source code belongs to the MEASURE project code base ().

- The feature of the M·ELKI tool are:
  - automatic registration on the MEASURE Platform
  - manual association to a project
  - manual selection of a clustering algorithm
  - manual configuration of the algorithm parameters
  - manual selection of the project measures
  - built-in visualization.

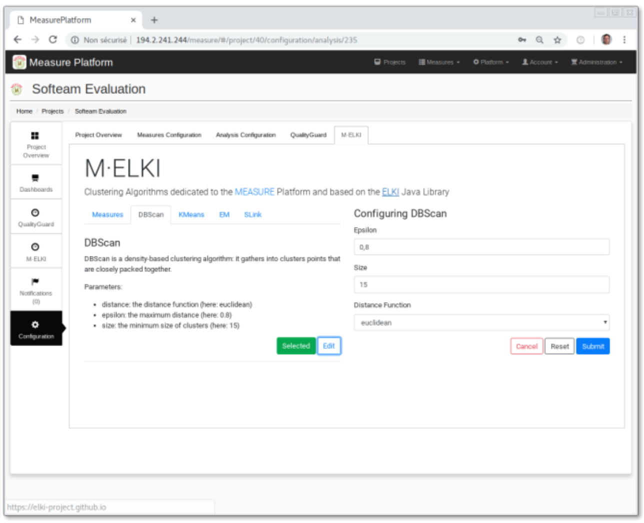
In fact, users have to apply M·ELKI to their projects. The following pictures show how to do such a registration: it is straightforward as users only have to select the M·ELKI tool.



Registred Analysis Tools

QualityGuard
The quality guard analysis tool is a web application which allows to provide advance data analysis functionalities apply to data collected by the measure platform.

M·ELKI
Clustering Algorithms dedicated to the MEASURE Platform and based on the ELKI Java Library



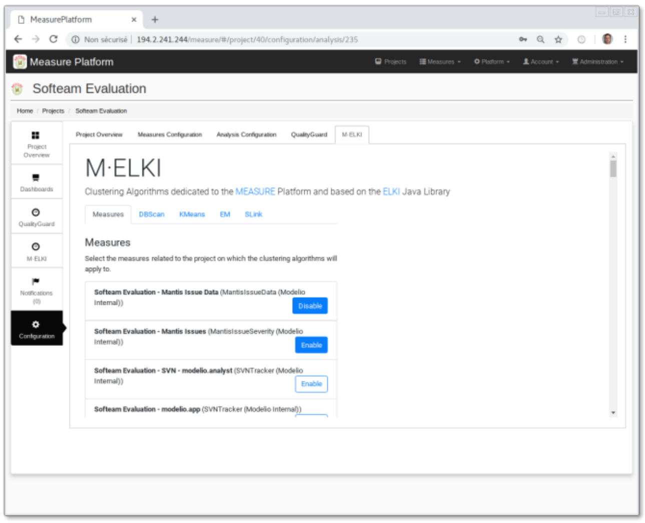
M·ELKI
Clustering Algorithms dedicated to the MEASURE Platform and based on the ELKI Java Library

DBScan
DBScan is a density-based clustering algorithm: it gathers into clusters points that are closely packed together.
Parameters:
- distance: the distance function (here: euclidean)
- epsilon: the maximum distance (here: 0.8)
- size: the minimum size of clusters (here: 15)

Configuring DBScan
Epsilon: 0,8
Size: 15
Distance Function: euclidean

The M·ELKI project instance is also easily configurable. There are two kinds of settings:

1. the selection and parametrization of a clustering algorithm among 4 algorithms (DBSCAN, K-MEANS, EM and SLINK, see picture above);
2. the selection of the project-related measures whose measurements will be processed by the select clustering algorithm (see picture below).



M·ELKI

Clustering Algorithms dedicated to the MEASURE Platform and based on the ELKI Java Library

Measures

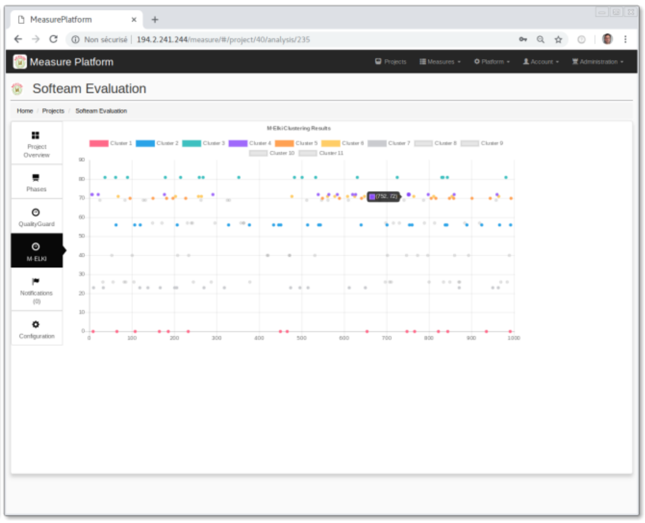
Select the measures related to the project on which the clustering algorithms will apply to.

Softeam Evaluation - Mantis Issue Data (MantisIssueData (Modelio Internal))

Softeam Evaluation - Mantis Issues (MantisIssueSeverity (Modelio Internal))

Softeam Evaluation - SVN - modelio.analyst (SVNTracker (Modelio Internal))

Softeam Evaluation - modelio.app (SVNTracker (Modelio Internal))



The last picture shows how to visualize M·ELKI clustering analysis results.

2.4.2. Future development plans

The future development phases will tackle these 2 main features (their release version number and their planned deadline are mentioned):

- project setting persistence on the side of the analysis tool (version 1.2, April 2019)
- results storage within the MEASURE Platform database (version 1.5, May 2019)

2.5. Stracker

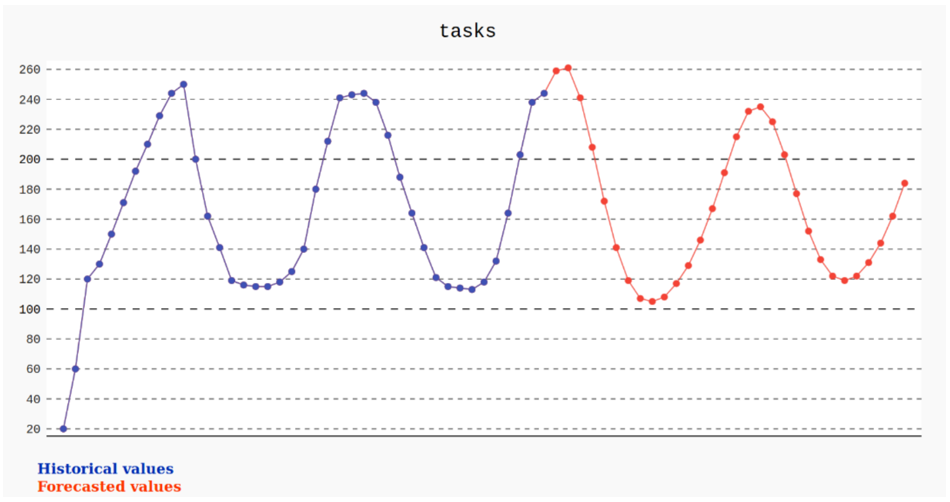
Stracker is an analysis tool that can be used for several purposes: software metric prediction, software metric correlation and software metric forecasting.

2.5.1. Current state and areas of improvement

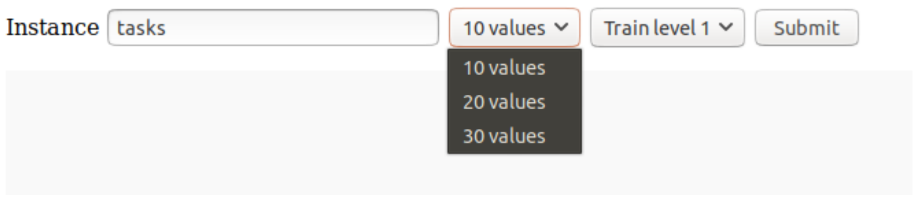
We build several features in the Stracker tool, all of the them using machine learning algorithms:

- **Prediction**: Predicting the value of a metric based on the values of other (correlated) metrics. For instance, we can do "bug prediction", i.e., predict if software bugs still exists in a piece of code based on several values of metrics. (We implemented state of the art algorithms from the literature and experimented with several machine learning algorithm trained on software metric databases from the internet.)
- **Correlation**: We checked the correlation of one or several metrics and compare it with actual values measured on the fly. For instance, we can compare the number of comments with the expected number of comments (obtained through correlation from other metrics).
- **Forecasting**: Based on history values of a chosen metric, we forecast future values of that metrics. For instance, we can forecast the future number of vulnerabilities or needed tests, thus helping the project managers to better provision and estimate the needed human resources.

We give an example of forecasting (blue - historical values" and red - forecasted values):



The above graphic can be configurable in the tool interface:



where the parameters are:
Instance: name of the metric instance for which we generate a forecasting
Values: number of forecasted values for the selected instance
Model Training level: A high value for this feature mean more accurate forecasted values, but also more time to process the data (train the internal forecasting model).

Assessment of the tool:

- *strong points*: Stracker uses the latest available algorithms to do metric correlation and metric forecasting.
- *weak points*: the interface is not very user-friendly and the results may be hard to interpret without explanations
- *integration aspects with MEASURE*: We integrated Stracker in MEASURE platform to use the values of metrics from the platform, although the integration is unstable sometimes
- *integration with other tools or standalone version of the tool*: A standalone version of the tool can import values from metrics tool Sonarqube.
- *assessment by the case study providers:* The tool was evaluated by Bitdefender and Softeam in deliverable D5.5. The tool was found to be useful by the evaluators, especially for its forecasting capabilities. E.g., in Softeam they can forecast the costs associated with certain activities and thus estimate the difference

2.5.2. Future development plans

Here are the future plans for Stracker

- *a list of new relevant features of the tool that will be implemented in the future to solve the weaknesses from the above subsection and improve the adoption, stability and completeness of the tool:*
  - Stracker currently implements the ARIMA forecasting model. In the future, a better model using LSTM (Long short-term memory) neural network will be implemented
  - We will completely redesign the UI of the application
  - The documentation of the tool will be improved, such that the user is guided through the workflow
  - Integration with external sources of software metrics will be enriched
  - The tool may be provided as a add-on or plugin to other software platforms, in order to ease adoption.
- *a release plan:*
  - Q1 2019: a lightweight version of the tool focussing on forecasting will be released
  - Q2 2019: a redesigned UI and better integration with external tool
  - Q3 2019: release of v1.0 of the tool
  - Q4 2019: release of v2.0 of the tool, improving various aspects based in feedback from users
- *any other comments or ideas regarding the tool:*
  - The accuracy of the forecasted values is better on series with many elements, whereas for series with few values the prediction is not very accurate. More historical values imply higher precision for forecasting, but also more time to process them.
  - For the moment, the application uses a single core in order to make the forecasting. Using more CPUs or GPUs in parallel will significantly improve the running time for the training phase.

3. Conclusions

The MEASURE WP4 objective is to design and develop a platform to analyse the large amount of measurement data generated by the use cases carried out in WP5. The data analysis platform will implement analytics algorithms, to correlate the different phases of software development and perform the tracking of metrics and their value. This deliverable D4.3 describes the assessment of improvement areas and countermeasures provided by the data analysis algorithms and supporting tools that are required to identify correlations and in particular the 5 MEASURE analysis tools:

- Quality Guard described by Softeam,
- rule-based correlation used in MINT tool developed by IMT and relying MMT-Correlator developed by MTI described by Montimage.
- The clustering provided by the M-ELKI analysis tool is described by ICAM.
- Machine learning provided by Metrics Suggester tool described by IMT and finally,
- Stracker presented by the University of Bucharest.

The deliverable describes an assessment of each of the tools (self-assessment and assessment by the industrial partners) and future concrete plans to improve them.