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

D5.5: Evaluation and analysis of the global results from the MEASURE case studies

M39 Public deliverable

Table des matières

1. Executive summary

This deliverable describes the experimental studies of four different industrial case studies through some of the analysis tools developed in the context of the MEASURE project. The industrial use cases have been provided by Softeam, Naval Group, Bitdefender and the Turkish partners. The MEASURE analysis tools that have been performed in order to assess the use cases are: Quality Guard, Metrics Suggester, MINT and STracker as illustrated by the Table below.

INDUSTRIAL USE CASES	ANALYSIS TOOLS
Modelio Product Line – SOFTEAM	*Quality Guard / Metric Suggester / STracker*
Large Naval Software System – NAVAL Group	*Quality Guard*
Security Analysis – BITDEFENDER	*MINT / STracker*
Test Case Generation – Turkish Consortium	MEASURE Platform assessment

For each case study analysis, we detail the use case context, the executed analysis tools, the evaluation approach together with the evaluation scenarios. Furthermore, we illustrate the experimentations and the obtained results. Finally, conclusions are drawn.

2. Introduction

2.1. Role of this deliverable

The main objective of this deliverable is to present the evaluation of the MEASURE platform by all the partners according to industrial scenarios and focusing on the MEASURE analysis tools.

2.2. Structure of this document

The document is composed of 5 main sections

- Section 2 presents the Introduction with a detailed reminder of our MEASURE platform
- Section 3 presents the evaluation of Modelio Product Line case study by SOFTEAM
- Section 4 presents the evaluation of a large naval software system by Naval Group
- Section 5 presents the evaluation of security analytics software by Bitdefender case
- Section 6 presents the evaluation of test generation software by the Turkish Consortium.

2.3. Relationship with others MEASURE deliverables

This document is related to:

- D5.1 deliverable "*Consolidated requirements from business use cases and assessment criteria*" that defined the project requirements from different point of views.
- D5.2 deliverable "*Evaluation Plan and Business Cases*" that defined the evaluation plan and use cases for MEASURE industrial partners from the French consortium (since they were the first to start the project).
- D5.3 "*Intermediate evaluation results from the executed case studies*". Last year, the MEASURE platform (release 0.6) was evaluated on 5 general scenarios, which were not specific to the industrial case studies from the members of the consortium and not related to the analysis tools.
- D5.4 "*Final evaluation results from the executed case studies". In December 2018,* we performed a first evaluation on the most recent version of the platform (0.8) including analysis tools within the context of specific industrial scenarios. In this deliverable that is not public, based on the evaluation methodology (D5.3) and requirements (D5.1), KPIs and goals (e.g., business goals) have been achieved.

In this deliverable, we present the final evaluation of all the components developed and smoothly integrated in a common MEASURE platform through well-defined industrial scenarios designing real industrial use cases provided by the companies of our consortium. In particular, we focus on the results obtained by our analysis tools in achieving general business indicators and goals.

2.4. Contributors of this document

For this deliverable, all the MEASURE partners have contributed either in providing the industrial use cases, in developing, integrating components into the MEASURE platform, or in processing the analysis tools and studying the obtained results.

All the work phases started at the very beginning of the project and this document is the final evaluation of our platform, its components and the analysis tools. All partners jointly worked to the design of assessment scenarios.

2.5. Evaluation plan followed by MEASURE

We herein remind that WP5 aims at evaluating the major outcome of MEASURE, that is its methods and tools within the context of real industrial use cases. Working on concrete use cases, it attempts to improve the efficiency and quality of software engineering activities and software products based on the gathered analytics information.

The evaluation of the methods, analysis tools and the MEASURE platform itself is carried out in two ways:

- Evaluation within the case studies directly to validate aspects such as accuracy and reliability of the measures made and the given estimates as well as the applicability of the developed methods and tools in an industrial software development process;
- Evaluation of the performed use cases by investigating if and how the MEASURE methods and analysis tools contribute to the project objectives such as
    - Optimising the productivity and needed resources (e.g., energy consumption) of the software development process;
    - Improving the quality of the software products (e.g., in terms of number of residual software bugs or compliance to non-functional requirements);
    - Improving the estimations and accuracy of data used for the software project planning and management (e.g., efficient executed software metrics).

For both types of evaluations, relevant use cases from the industrial partners are selected; each of them are directed to answer specific evaluation criteria that are of interest to MEASURE and its users. According to Goal Question Metric (GQM) that have been defined in Deliverable D5.2, diverse metrics have been selected and related to analysis tools for our experimentations and evaluations. Each of them has been defined by the industrial experts in charge of the use case accordingly to the tools providers. Indeed, an expertise of the tools, platform and industrial case study was needed to select the relevant goals, metrics and questions to raise. Moreover, for broad analyses, diverse scenarios have been defined in order to cover several metrics and to target different challenges for each of the use cases. These have been crucial requirements for the assessment. Therefore, the following case studies have been performed:

- Modelio Product Line by SOFTEAM analysed by the analysis tools: Metrics Suggester, Mint, Quality Guard and STracker,
- Large Naval Software System by NAVAL GROUP analysed by the analysis tools: Quality Guard,
- Security Analysis by BITDEFENDER analysed by the analysis tools: Mint and STracker,
- Test Generation from the Turkish Consortium.

For this last case study, the evaluation is different. Indeed, the Turkish consortium joined the MEASURE project in February 2018. Their industrial case study is still being defined and their integration with the MEASURE platform in progress. However, the Turkish partners deeply evaluated the MEASURE platform through several indicators and following an internal questionnaire.

2.6. Reminder on the MEASURE platform

The MEASURE platform is a tool dedicated to measure, analyse, and visualise the metrics and to extract and show information of the software engineering processes.

- Implement the tools to automatically measure software engineering processes during the whole software lifecycle by executing measures defined in SMM standard and extracted from a catalogue of formal and platform-independent measurements.
- Provide methodologies and tools which allow measure tools provider to develop a catalogue of formal and platform-independent measure.
- Implement storage solution dedicated to measurements resulting of measure execution in big data context.
- Implement visualization tools to expose the extracted results in an easy-readable fashion, so allowing a quick understanding of the situation and the possible actions that can be taken to improve the diverse stages of the software lifecycle.
- Implement an extension mechanism dedicated to the integration of external analysis tools will provide long terms analysis and predictive evaluations on collected measures.
- Implement an Extended API allowing to facilitate the integration on Measure Platform with external tools and services.

The platform activity is organised around its ability to collect measurement by executing measures defined by the SMM standard. SMM measures are auto-executable component, implemented externally, which can be interrogated by the platform to collect measurements.

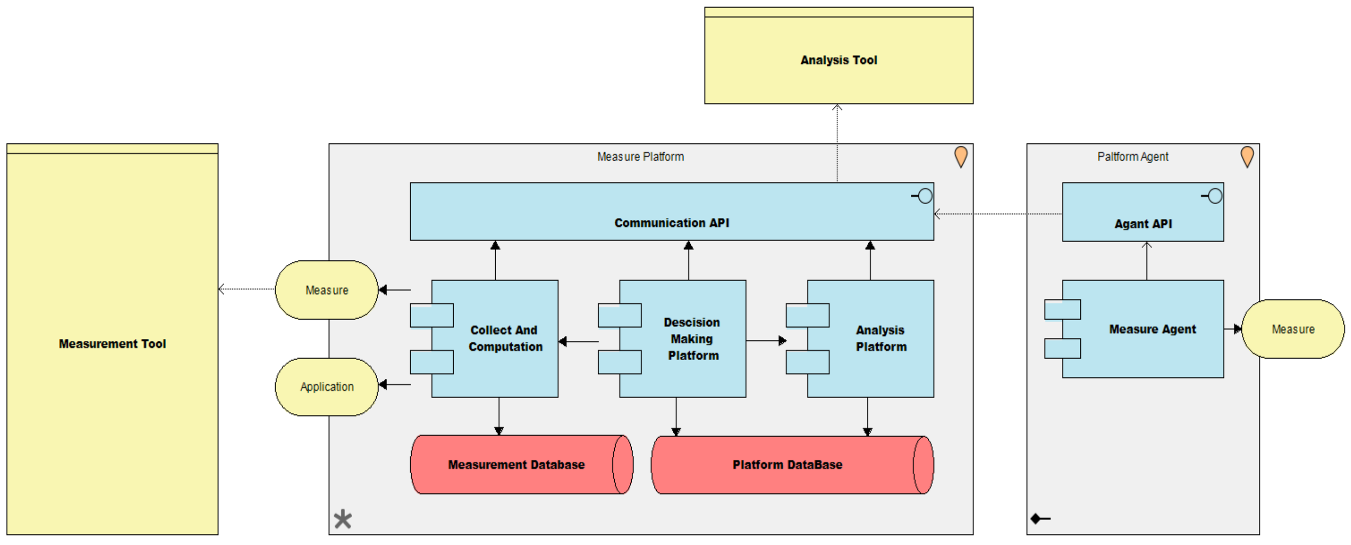


Figure 1. Architecture overview of Measure Platform

Measure Platform: Central component of this deliverable, the Measure platform provides services relate to data collection, analysis and display. It's composed of six sub-components:

Platform Agent: Measure tools on client side which collect data. The executable measure provides a way to collect data in physical world. A measure can be executed on platform side and collect physically this data through an existing measure tool. A Measure can also be directly executed on client side (a computer close to measured element) by the intermediary of a Platform Agent.

Measurement Tools: A Measurement Tool is and external tool which collects or calculates measurement from a specific source. In context of the project, 5 Measurements tools has been developed (see section 3) and a lot of existing tools in the market (such as SonarQube for code quality metrics) can be also considered as Measurement Tools.

Analysis Tools: An Analysis Tool 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. The analysis tool is integrated to the platform using a specific API. This integration includes embedded visualisation provided by the analysis service into the platform.

Measures: A Measure is a small and autonomous Java program based on the SMM specification which can collect measurements. A Measure can be Direct (Collect of measurement in physical world), a Proxy (Ensure communication between a Measurement Tool and the Platform) or Derived Measure (Measure calculated by the aggregation of existing Measures).

Applications: An Application is a set of Measures aggregated together in order to address a functional requirement. The application is associated with a visual dashboard which is directly integrated into the Decision-Making platform when the Application is deployed on a project.

Hardware and Software Requirements

System	**Linux, Windows**		
Installation Scenario	**Minimum Requirement**	**Standard Configuration (500 Metric Collection, Basic Analysis, 50 Users)**	**Advanced Analysis (+2000 Metrics, All Analysis Services, 200 Users)**
RAM	4 Go	8 Go	16 Go
Processor	32-bit, 4 cores	64-bit, 4 cores	64-bit, 8 cores
Hard Disk	80 GB for system drive	80 GB for system drive	200 GB for system drive

Prerequisites

Install MySQL Database

- 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 "measureplatform".

Using MySQL Command Line Client: CREATE DATABASE measureplatform;

Install Elasticsearch

- Download Elasticsearch ver. 5.6 or above (as zip) *https://www.elastic.co/downloads/elasticsearch*

- Unzip the application in your tool directory.

Install Kibana

- Download Kibana ver. 5.4 or above (as zip)

  https://www.elastic.co/downloads/kibana

- Unzip the application in your tool directory

Java 1.8 Installation

- Download and install the jdk8 in your environment:

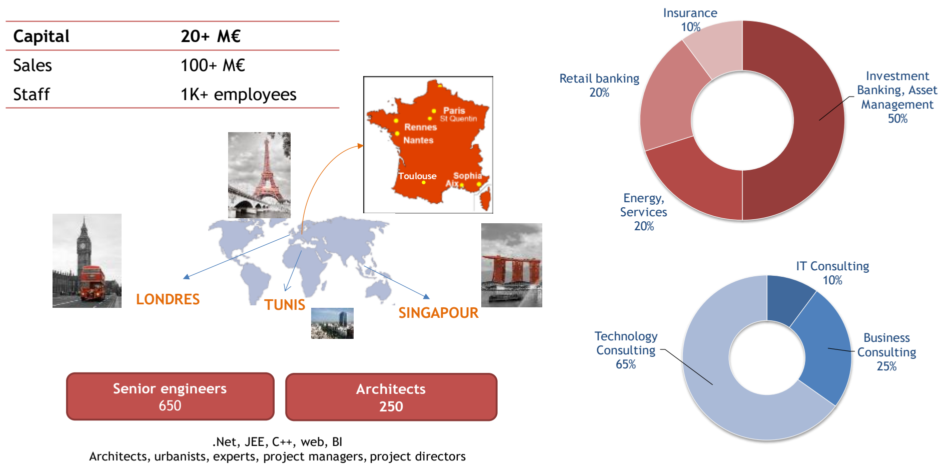
http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

3. Use Case: Modelio Product Line by SOFTEAM

3.1. Use Case Context

3.1.1. Softeam Group

SOFTEAM is a French IT service company of about 920 employees and 82M€ in revenues, based in Paris, and having subsidiaries in Saint Quentin en Yvelines, Rennes, Nantes and Sophia Antipolis. Founded in 1989, SOFTEAM has a long experience in object-oriented methodologies, and has been developing and distributing an object-oriented CASE tool since 1992.



Capital	20+ M€
Sales	100+ M€
Staff	1K+ employees

Figure 2 : Softeam Group

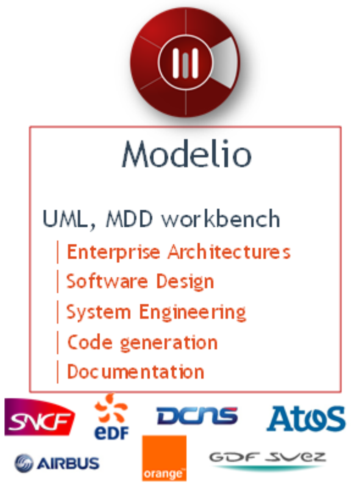
SOFTEAM is involved in three main activities:

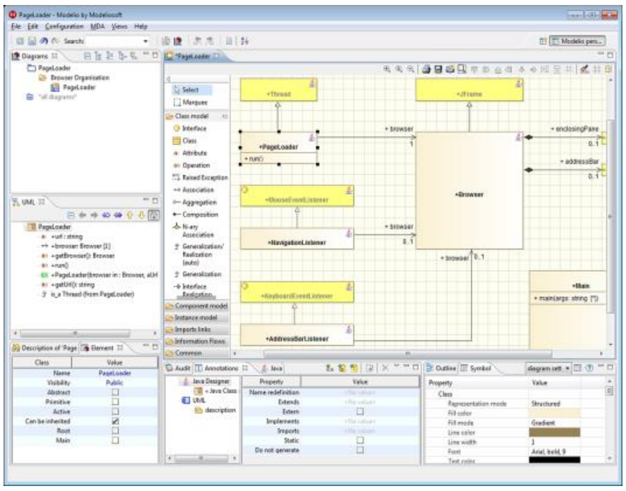
- IT services and consulting: as a provider of object-oriented methodologies, SOFTEAM undertakes a wide range of custom application development projects and is also an active OMG contributor.
- Training: SOFTEAM provides training in related technical areas (languages, methodologies, tools) and in methodological areas.
- CASE tool publishing: SOFTEAM's affiliated company Modeliosoft commercializes solutions based on the open source Modelio UML workbench, providing a supported version of Modelio with extensions dedicated to enterprise systems, and a range of support and maintenance services. Modeliosoft also provides consulting, training and made-to-measure development services.
- Modelio.org: engaging open source community for development of a professional UML workbench.

SOFTEAM solutions and services are used in many industrial domains including: Aerospace & Defence, Automobile, Telecommunications, and Banking & Insurance.

3.1.2. Modelio Toolsuite

Modelio is a Modeling Tool developed by Softeam since January 2009 supporting and integrating all the latest major modelling or methodology standards. Modelio is first and foremost a UML modelling environment, supporting a wide range of models and diagrams, and providing model assistance and consistency checking features. BPMN support is integrated with UML with Modelio combining Business Process Modelling and Notation (BPMN) and UML support in one tool, with dedicated diagrams to support business process modelling.





This software is part of a multi-tool environment combining the modeling tool, several extensions who add new features to Modelio, the Constellation server who allow our customer to organize collaborative works around Modelio and Web Analyst, a cloud services which provide services to model Requirements in relation with a Modelio project. Modelio is declined under various platforms (Windows, several Unix platforms and Mac) and is supplied as several packaging: an open source version and tree different commercials versions.

This Flagship products are completed by a set of extension components called Modules which allow to extend services provided by Modelio in order to adapt the tool to specific requirements of our customers. Each of these modules follows a shorter development cycle than Modelio. Improvements and release are more frequents and are driven by customer requests. The validation process is also simplified.

The Modelio Product Line includes also two web application providing specific challenges and requirement to MEASURE project: Constellation and Web Analyst.

Our first Cloud based product, Constellation, is an advanced repository which stores the models defined using the Modelio case tool. It was published in 2014 and has been evolving constantly. The server provides advanced functionalities, including storing models in remote repositories and managing their versioning, organizing collaborative work around these models and supporting the execution of high resource consuming processing operations on these models. The fragments technology and the Constellation are SOFTEAM's answer to the increase in complexity of models. The Constellation server is used to centralise the storage of all models of an organisation.

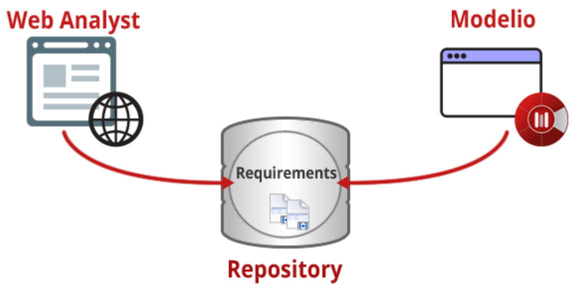


Figure 3 : Web Analyst, a web client of Modelio

Our second Cloud based product, Web Analyst is a Web requirement management tool that works on the Modelio repository sharing the same requirement elements and the same "Constellation" portfolio management tool. Modelio Web Analyst provides new capacities for requirement sharing and management. It also provides a complete traceability amongst every deliverable and component of a system or an application (Application component, process, code, business object, test …).

3.1.3. Reason to use the Measure Platform and specific challenges

SOFTEAM aims at applying MEASURE technologies within the SOFTEAM Modelio tool thus gaining a competitive advantage in time to market in comparison to our competitors.

The Modelio Toolsuite is composed of several components of different natures. Each of these software components follows their own development cycle and the developments processes are adapted to each software components depending on the role and criticality of the tools in Modelio product line. Consequently, it became very difficult for us to accurately track the developments of each of these products, assess quality of this different components in an integrated way and know the advancement of each new release of these software components.

First, we have used tools and techniques developed in MEASURE project will address this problem by providing us a new way to monitor and analyses our software development process. Next, during the last years, in order to complete our product line, we develop new web application which provides new services in relation with Modelio. Performances and efficiency of this application which allows hundreds users to work in collaboration around Modelio, are become one of our first priority. With MEASURE we would like to have green metrics to monitor Constellation and Web Analysts servers to provide input for finding relevant elements during the Deployment phase, to challenge the energy consumption of a given code against computing-equivalent codes.

Challenges for MEASURE

- Define metrics and develop methods and tools for automated, precise, and unbiased measurement of software engineering activities. This metrics should be focused on Specification, Design and Architecture, Implementation and Testing and Deployment developments.
- Deploy and evaluate a set of metrics dedicated to architecture development phases driven by models (UML / BPMN …). Deploy an infrastructure able to processing a large volume of Measures provided by big models repository.
- Integrate the Measure platform with existing quality and project management tool already deploy and used in Softeam development environment.
- Integrating measurement metric tools into Modelio development environments and processes. The measurements tools should be able to collect measurements form product artefacts and tools involve in development process.
- Provide a way to monitor and follow the development status of specifies features related to several small software components in an integrated environment.
- Offer a synthetic view resulting of the analysis several measures allowing a manager to extract key information related to several development process.
- Used advanced analysis techniques like forecasting and automated recommendation system to help us to exploit the monitoring data and define mitigations actions to issues detected with the help of the Measure platform.

3.2. Evaluation Approach

3.2.1. Evaluation Goals

The purpose of this evaluation is to study the usefulness of “Measure Platform” monitoring tool and analysis tool. The academic approach to perform this kind of evaluation is to conduct a comparative study focused on differences between two similar projects with one being monitored by "Measure Platform" and the other not being monitored and evaluate the gains and limitations of using the platform in this similar contexts.

In our specific case, the monitoring of development process of the Modelio Product Line, this approach is difficult to apply due to the fact that each of our developments cycles are different and has its own characteristics (functionalities, complexity, technical difficulties, etc). More, we do not have a sufficient number of monitored projects to use a statistical approach in order to minor the impact of this differences between analyse projects. It would be meaningless to apply this comparison in our current evaluation process. To solve this issue, we decide to conduct the comparative study on the same project by comparing its historical data and the forecasted data.

For this end, two evaluation scenarios have been specified:

- The first scenario consists of analysing "historical data" previously collected by "Measure Platform" throughout the project life cycle history, identify "Key Performance Indicators" related to the software quality in all its phases and estimate the gained cost if the platform monitoring tool had been used.

- The second scenario consists of monitoring the project development processes in real time with "Measure Platform", solve issues highlighted by the monitoring tool and estimate the impact of the integration of analysis tools in our internal development process

During this evaluation, we will measure the impact of the deployment of the Measure platform on Software Development project managed and conduct by the Modeliosoft development team and estimate the potential cost that could have been spared if issues were detected earlier.

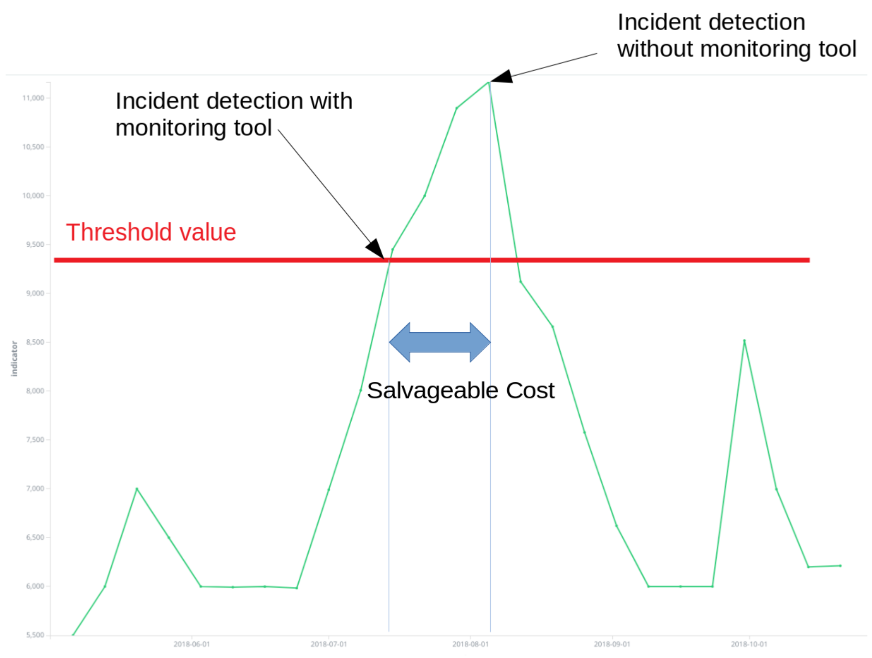
3.2.2. Evaluation Criteria

The evaluation of the current case study has for goal to assess the impact of the Measure Platforms, Monitoring tools and Analysis tools on Softeam development using a qualitative and quantitative approach. The qualitative approach aims to validate that the delivered product rich mains requirements identified at project start-up. The quantitative strategy provides a measured estimation of the financial gain resulting of the deployment of the platform.

The key requirements used to perform the qualitative evaluation are listed below and results will be assessed using survey study completed by the Modelio team members involved in in the platform usage:

- Capability of MEASURE to provide accurate and relevant measures on each phase of Modelio development process and helping users to improve this development process.
- Capability of MEASURE to provide detailed analysis related to the state of development of Modelio low level component, Modules and others artefacts delivered by Modeliosoft based on collected measures.
- In order to address Product Lines issues, capability of MEASURE Provide aggregated measures dedicated to client specifics packaging. These aggregated measures will have to be constructs form unitary measures include in each specific packaging.
- Capability of MEASURE to present these results on an integrated way via a web application to non-specialist end users likes managers. This interface will have for goal to presents unitary measure, aggregated measures and global analysis to end users. More, this interface should allow to detect early the issue on development process of a specific product by trigging alarms if predefine set of constraints defined on global or unitary measures are break.
- Capability of MEASURE to provide new kinds of references measures dedicated to model driven developments approaches and allowing monitoring specification and architecture phase of a software development process driven by Models.

The quantitative evaluation is base of the fact that an incident identified earlier in development process will cost singly lower to solve than an incident detected during a more advanced phases of the development process.



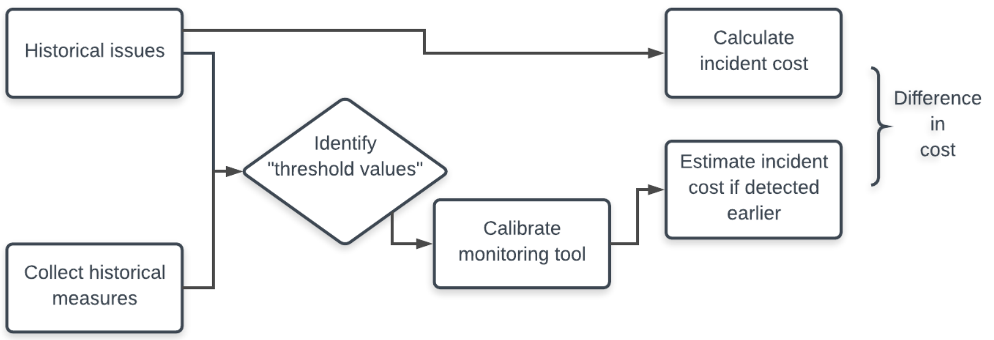
Using the two-scenario presented below, we tried to evaluate the cost saved by the deployment of the Measure platform into Modeliosoft development team by comparing the cost to solve an incident detected using the Measure Platform and the cost to solve the same incident by applying the quality process and tools that existed in our development team before Measure.

3.2.3. Evaluation Scenarios

Scenario 1: Analyse of historical data using Measure Platform and compare it to project backlog

In this scenario, we analyse the evolution in time of the measures collected by “Measure Platform” on a specific project throughout the history of the software development phases. Based on the evolution graphics we look for interesting features and particular moments in history like maximum points or sudden changes in value that could be signalling issues or vulnerabilities. Next, we look for links between these points and the important events or incidents occurred during this project using the project backlog and try to find cause-effect relationships. This would allow us to identify threshold points or limits that should not be exceeded to avoid these types of incidents these threshold points will be used to configure monitoring tool of “Measure Platform”.

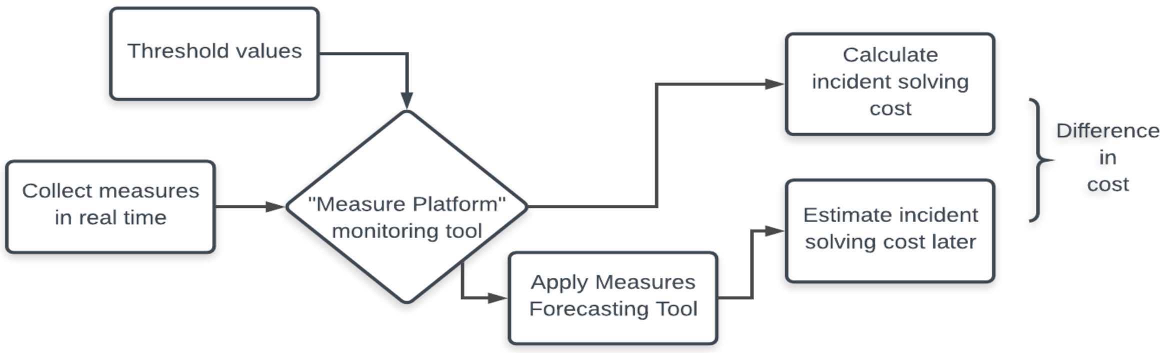
Finally we estimate the gain in cost brought by the monitoring tool by subtracting the cost of solving the detected incident without the monitoring tool and the cost of solving the incident if it had been detected earlier


Historical issues
Collect historical measures
Identify "threshold values"
Calibrate monitoring tool
Calculate incident cost
Estimate incident cost if detected earlier
Difference in cost

with the monitoring tool (salvageable cost, Figure 1). The costs are estimated in duration which refers to the extra amount of work that had to be done to resolve the issue.

Scenario 2: Monitoring Modelio 3.8 developments process with the Measure Platform and Analysis Tools

In this scenario we collect in real time, measures of the same metrics used in first scenario. These measures are being monitored continuously by the incident detection tool configured with the same threshold values specified in the first scenario.


Threshold values
Collect measures in real time
"Measure Platform" monitoring tool
Apply Measures Forecasting Tool
Calculate incident solving cost
Estimate incident solving cost later
Difference in cost

Afterwards, we will include in this scenario the usage of several analysis tool which will help us to detect and address quality issues encountered during the development process of our flagship product:

- **Metric Suggester**: Provides a framework to automatize the suggestion of software metrics based on an initial measurement plan. 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.
- **Quality Guard:** Provide a simple way to monitor in real times the data collected by the platform and to raise alerts and trigger adapted countermeasures if this data drifts beyond a predetermined threshold.
- **STRACKER**: Increase the quality of software development by tracking and suggesting 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.

Finally, we will try estimate the gain in cost brought by the monitoring tool by calculating the difference between the cost of solving the issue at early stage thanks to the monitoring tool and the estimated cost to discover and fix the issue later. This estimation of the cost of fixing the issue later would assume that the issue had not been discovered yet and the evolution of the quality indicators will be predicted using a forecasting tool.

3.3. Experimentation

3.3.1. Deployment of the Measure platform in Modelio development team

In context of the current use case, the Measure platform has been deployed on the Modelio development Team. The team is composed of 20 peoples and includes various profiles like Product Owner (2), Project Managers (4), Developers (10), Quality Engineers (2), Integrators (1) and customer support (1).

An instance of the Measure platform has been set up specifically for this purpose and has been updated on a regular basis to integrate new functionalities developed during the projects. The platform has been configured in order to collect metrics related to developments activities of the team during the evaluation period. In order to evaluate the interest of the Measure platform for several kind of end users, 6 peoples of various profile form this team has been introduce to the usage of the Measure Platform through an internal training organized at the M18 of the project. They were then regularly informed of developments in the platform, including the integration of analysis tools.

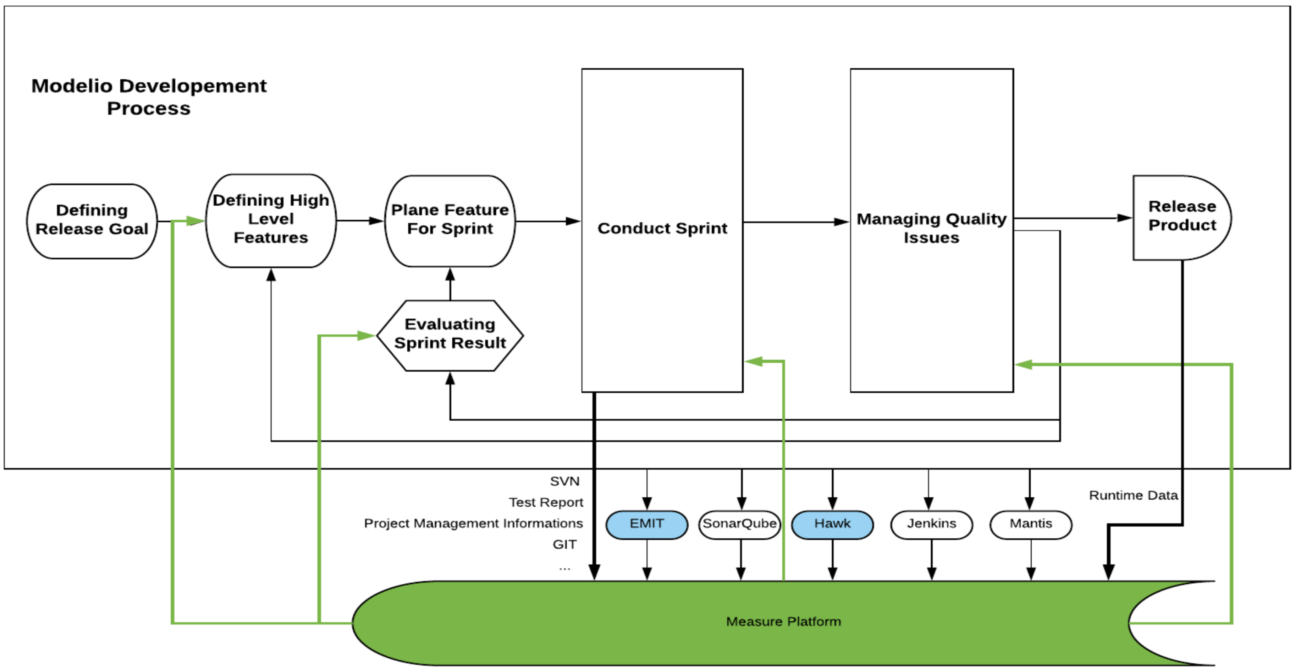


Figure 4 : Integration of Measure into Modelio development process

Integrated to Softrams daily work environment, the Measure Platform provide decision support to decision makers (e.g. managers, POs and developers) in different activities within the development process. For instance, the Measure Platform provide quality related data support decision in definition of features, strategies, sprint implementation, integration and validation. Product owners may use alerts (QualityGuard analysis tool), recommendations (MINT analysis tool) and forecasting services to address quality concern.

The table below provide an overview of data sources repository which has beans analysed using Measure Platform by the intermediary of 30 Measure metrics and the Hawk measurement tools.

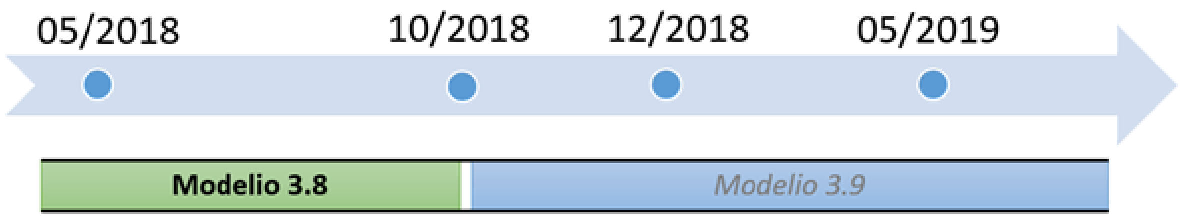
SVN Source Code Repository *Stored as UML Modelio models*	- **8 monitored repository** : modelio.design, modelio.analyst, modelio.app, modelio.archimate, modelio.bpmn, modelio.platform.core , modelio.platform.services modelio.uml - **3 milions lignes of code**, 5000 classes - **4 years history of development activities**
Issues and Bug Traking *Mantis Bug Traker*	- **5000 issues** in modelio toolsuite project - 5 years history
Modelio / Architecture Quality Quality of Architecture Model	- **6 Architecture Model repository analysed** - 10 metrics calculated
Code Quality metrics *SonarCube*	- **7 daily analysed source code repository :** aggregator-modelio.analyst, aggregator-modelio.app , aggregator-modelio.archimate, aggregator-modelio.bpmn, aggregator-modelio.platform.core, aggregator-modelio.platform.services, aggregator-modelio.uml - **60 code quality metrics** calculated daily
Integration Test Data Internal developed testing framework	- **Automated integration tests executed for each builds**
Product Integration Data Jenkins	- **9 Integration process monitored** : Realtime-Modelio.app, Realtime-Modelio.archimate, Realtime-Modelio.app, Realtime-Modelio.bpmn , Realtime-Modelio.app , Realtime-Modelio.platform.core , Realtime-Modelio.platform.services, Realtime-Modelio.uml - **Daily Builds + builds at each commits by development team**
Project Management Data OpenProject	- **Monitor Modelio project development plant,** project tasks identification and progress report
Modelio Runtime Log Modelio Client	- **Collecting Runtime Log data from Modelio 3.8 clients** - Deploy at this time internally on Softeam Team developers : The business utility of this logging tool needs to be demonstrated before considering a larger deployment (Modelio Open Source Version)

3.3.2. Scenario 1: Analyze of historical data using Measure Platform compared to project backlog

As mentioned previously, the historical measurements are analysed in order to identify threshold points that allow us to predict future issues in advance. These measurements are collected weekly by “Measure Platform”. In addition to these metrics, we analysed the product backlog history in order to link the observed events with incidents registered in the product backlog history.

Extract product development incident history form backlog

The historical data analysis has been conducted during a period of 6 months between May and the October 2018. This period corresponds to the end of the development of version 3.8 of Modelio tools originally planned for September but finally released in October.



The main features developed in this release addressed the integration of a workflow system on model elements the development of a trans meta-model tractability management system as presented in the following extract of release notes of the version presented below.

- New 'Workflow' feature allowing to manage state lifecycle for workflow submitted elements.
- New 'Methodology' feature allowing to precisely link elements belonging to different met models.
- The 'Workspace' feature has been fully reworked and now allows projects organisation in Working Sets as well as an easier access to Constellation projects.
- New set of icons (now 24x24 for better visibility).

Based on **Projects Backlogs**, two majors' incidents which has for consequence to postpone the release of the version form a month occurred during the development process of the versions:

- At the end of august we the quality team have detected several functional issues on workflow feature related to the specification of the Workflow feature. A redefinition of the services provided by the function turned out to be necessary and led to new development and to the postponement of the release of 3 weeks (2 people’s works on the subject).
- In early October, a few days before the new release date, a blocking issue (malfunction) had been detected on 'Workspace Creation” feature which has for consequence to postpone the release of one weeks (3 people’s works on the subject).

Collect metricd and Analysis of time-series graph

The collected measurements are mostly numerical values collected periodically at the end of each week and can be represented by time-series graphs. Our approach consists of searching for inflection points or “out of the ordinary” points in these curves such as changes in values, changes in slopes, maximum or minimum points, etc.

First, we explored SonarQube and Hawk metrics and we found remarkable changes in SonarQube violations curve. The violations are issues that have effect on code maintainability, reliability and security. Since Modelio is a big project with many years of code history, there is a large number of residual issues making it difficult to distinguish weekly variation in the number of issues. Hence, we will focus on issues brought by the new added code. Fortunately, SonarQube provide metrics based solely on new code regarding maintainability, reliability and security. Furthermore, the number of issues includes issues with different degrees of severity, that is why we prefer the metrics expressed in duration such as the “technical debt” (maintainability effort), “reliability remediation effort” and “security remediation effort”. The Figure 8 illustrates the measurements collected for these metrics during the period from 15/04/2018 to 21/10/2018. In this figure we find two noticeable peaks shared by these metrics and occurred at the dates 27/07/2018 and 09/09/2018.

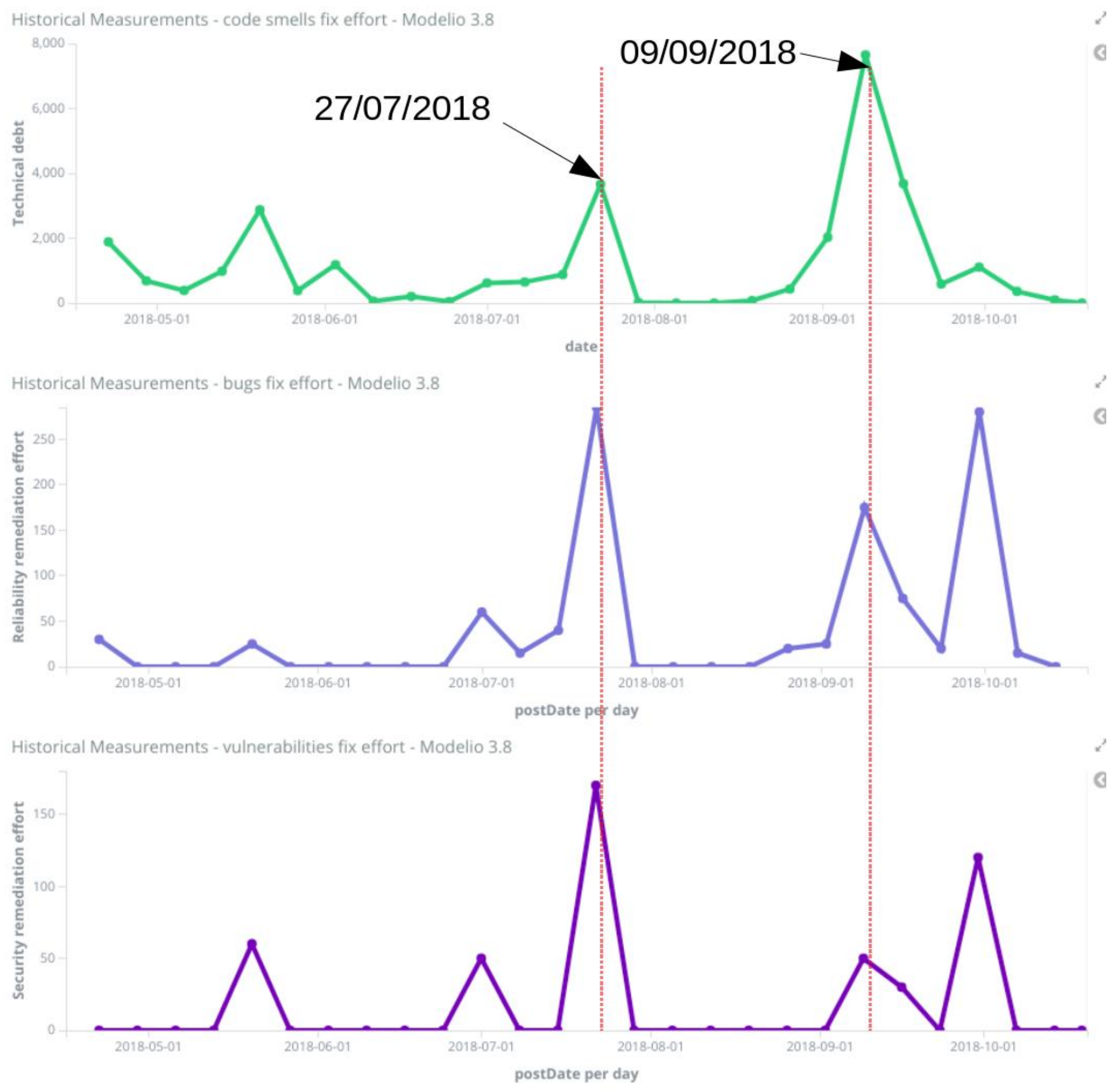


Figure 8: Peaks observed in SonarQube metrics

These peaks indicate a sudden increase of issues at these particular weeks. These events are observed in several metrics and can be explained by important product releases preceding these dates.

The next step is to explore the number of severe incidents (major and block incidents) submitted by developers on Mantis during the same period. The Figure 9 illustrates the number of severe incidents submitted on Mantis by week. We noticed in this figure an irregular increase in the number of the recorded incidents starting from the dates 01/0è/2018 and 08/10/2018.

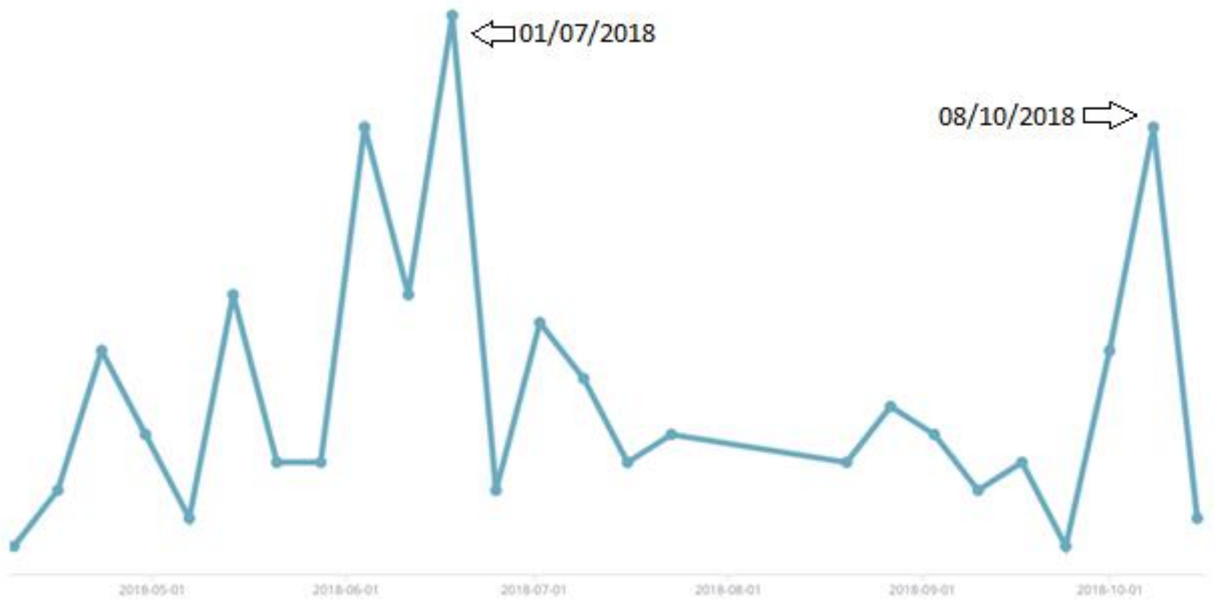


Figure 9: Severe incidents submitted on Mantis

Comparing incident periods with Measures and Identifying threshold values

The link between the collected measures and the product backlog incident history indicates that these measures can help predict incidents if they reach threshold values that trigger incidents. This allow us to estimate the gain in cost of fixing the incidents earlier when the threshold values had been reached. Based on the measures illustrated previously we find the following threshold points and estimated costs. Monitoring of Modelio development process with the Measure Platform

Measure Platform				Project Backlog	
				Development event : Workflow Feature Issue	
Metric	**Inflection point date**	**Threshold value**		**Incident detection date**	**Detection Delay**
New technical debt	27/07/2018	3000		24/08/2018	18 days
Security remediation effort on new code	27/07/2018	200		24/08/2018	18 days
Reliability remediation effort on new code	27/07/2018	100		24/08/2018	18 days

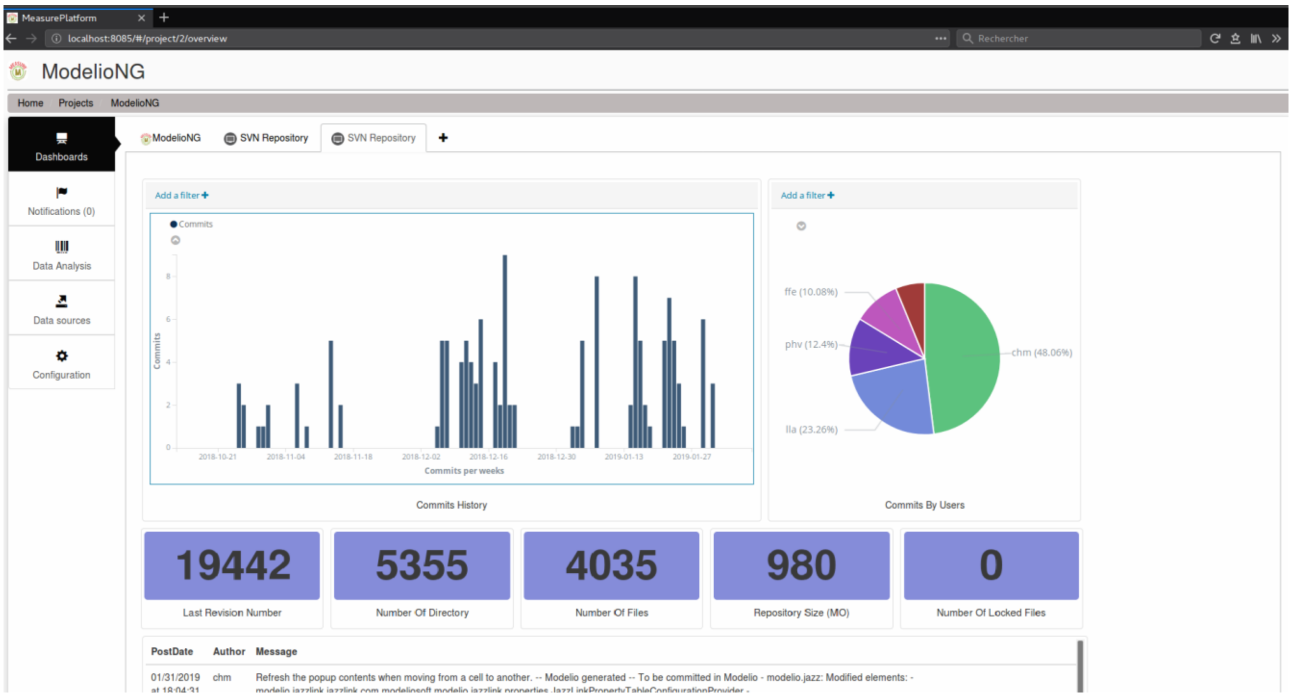
Number of severe Incidents (Mantis)	01/07/2018	10	24/08/2018	24 days
Estimate Remediation Cost with Measure Platform	5 day for 2 peoples = 10 m/d		**Remediation Cost**	15 day for 2 peoples = 30 m/d

Measure Platform			**Project Backlog**	
			Development event : Workspace Creation Issue	
Metric	**Inflection point date**	**Threshold value**	**Incident detection date**	**Detection Delay**
New technical debt	09/09/2018	3000	08/10/2018	19 days
Security remediation effort on new code	30/09/2018	200	08/10/2018	4 days
Reliability remediation effort on new code	08/10/2018	10	15/10/2018	7 days
Number of severe Incidents (Mantis)	08/10/2018	10	15/10/2018	7 days
Estimate Remediation Cost with Measure Platform	1 day for 3 peoples = 3 m/d		**Remediation Cost**	5 day for 3 peoples = 15 m/d

3.3.3. Scenario 2: Monitoring of the development of Modelio 3.8 using Analysis Tools

In this scenario, we integrate the Measure Platform in the development process of Modelio 3.8 and we look for potentially “gained costs” by fixing issues earlier due to the monitoring tool. We then estimate these “gained costs” by calculating the amount of extra work that had been avoided by fixing issues earlier as opposed to fixing them later assuming the development team had not integrated “Measure Platform” and had not been notified yet of the existence of these issues.

Since October 2018, the platform was integrated in the development process of Modelio 3.8. The platform keeps the developers informed about the current quality level by displaying on dashboard customizable views from selected measures that summarize the development progress. The figure below shows the dashboard of “Modelio 3.8” project containing multiple views such as number of violations, number of SVN commits by author, etc.



19442	5355	4035	980	0
Last Revision Number	Number Of Directory	Number Of Files	Repository Size (MO)	Number Of Locked Files

PostDate	Author	Message
01/31/2019 at 18:04:31	chm	Refresh the popup contents when moving from a cell to another. -- Modelio generated -- To be committed in Modelio - modelio.jazz: Modified elements: - modelio.jazzlink.jazzlink.com.modeliosoft.modelio.jazzlink.properties.JazzLinkPropertyTableConfigurationProvider -

To complete our analysis process , we have deployed in the project sevaral analysis tools : the **Metric Suggester** to configure our measurement plan , the **Quality Guard** to monitor in real times the data collected by the platform and to raise alerts and trigger adapted countermeasures if this data drifts beyond a predetermined threshold.

Integration of the Metric Suggester Tool to configure our measurement plan

The Metric Suggester tool provides a framework to automatize the suggestion of software metrics based on an initial measurement plan. 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.

To integrate Metric Suggester in our project analysis, we have followed three steps:

- **The initialization of the measurement plan**: The initial measurement plan is elaborated by the expert and it defines the observed set of metrics, the corresponding software properties and the mandatory metrics, the ones that must always be in the suggested measurement plans. This measurement plan is the definition of the measurement context: what is observed by the software properties, how it is observed by the set of metrics related to the properties and the mandatory ones.

- **The analysis of measurements data through the supervised classification algorithm SVM:** The analysis consists in classifying a set of measurement data, more precisely a set of vectors $\vec{v}$. Each vector is classified in one class which refers to a software property defined in the initial measurement plan and related to ISO/IEC 25010. To classify the data, we use a supervised learning algorithm SVM.

- **The suggestion of novel measurement plans:** The suggestion is based on the classification result and the initial measurement plan. The result of the classification is a set of clusters. The number of clusters is the number of defined class and the data in the clusters is the set of vectors gathering during the measurement process. From this set of clusters, we choose the one with the largest number of vectors classified as the class of interest. Then, we add in the new measurement plan the set of metrics corresponding to the class of interest defined in the initial measurement plan. If all the vectors are classified in the same class, we thus suggest all the metrics. Next, in the first case, we determine the metrics which were necessary for the classification by using the RFE algorithm. The RFE algorithm used the classifier and the used data to select the features which allowed the classification result. Once the selection is done, we add the selected features in the new measurement plan. Finally, we add the mandatory metrics defined in the initial measurement if these latter were not selected by the previous steps. Then according to these steps, we generate the new measurement plan as a suggestion.

In the context of our case study, the analysis of this tool focuses on the developed code and we have used the Metric Suggester tool to help us to select the most relevant Code Quality metrics for a large set of metrics provide by a Sonar Cube Measure. The considered set of metrics for the measurement process includes 13 metrics giving information on 3 software properties: the maintainability, the reliability and the security of the code. The metrics related to the maintainability property give information on the quality of the code. The ones related to reliability give information on the reliability of the services and the security ones are about the vulnerabilities in the code. The table described this initial MP.

Table 1: Initial Measurement Plan

Metrics	Classes	Mandatory
Code smells	Maintainability	X
New Code smells	Maintainability	
Technical debt	Maintainability	
New Technical debt	Maintainability	
Technical debt ratio	Maintainability	
Bugs	Reliability	X
New Bugs	Reliability	
Reliability remediation effort	Reliability	
New Reliability remediation effort	Reliability	
Vulnerabilities	Security	X
New Vulnerabilities	Security	
Security remediation effort	Security	
New Security remediation effort	Security	

Once the initial measurement plan defined, we have train the classifier with a training file of 100 manually classified data. Then, as suggestion experiment, we use as input files to analyse, a dataset of 50000 unclassified vectors divided in 10 subsets of 5000 unclassified vectors.

The Figure 2. shows the results of suggestions based on the previous analysis model. The results show a dynamic suggestion of measurement plan. Each measurement plan is between 5 and 13 metrics. There aren't convergences and the suggested measurement plan evolves continuously according to the dataset values.

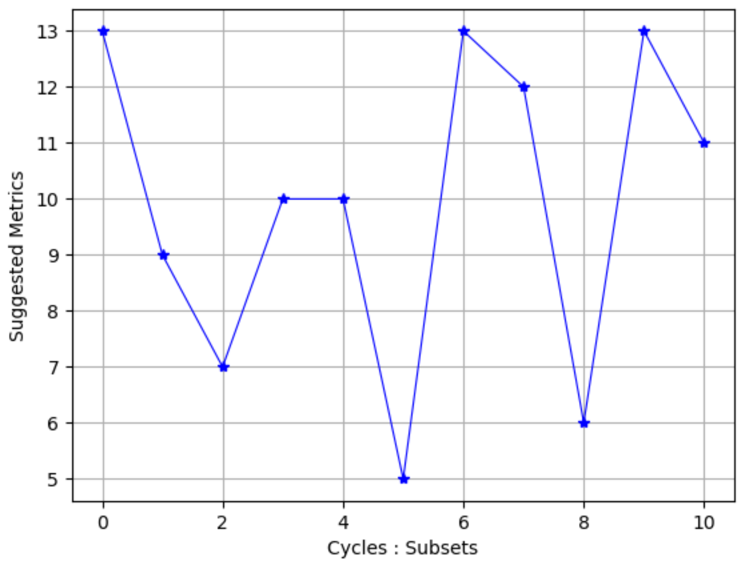


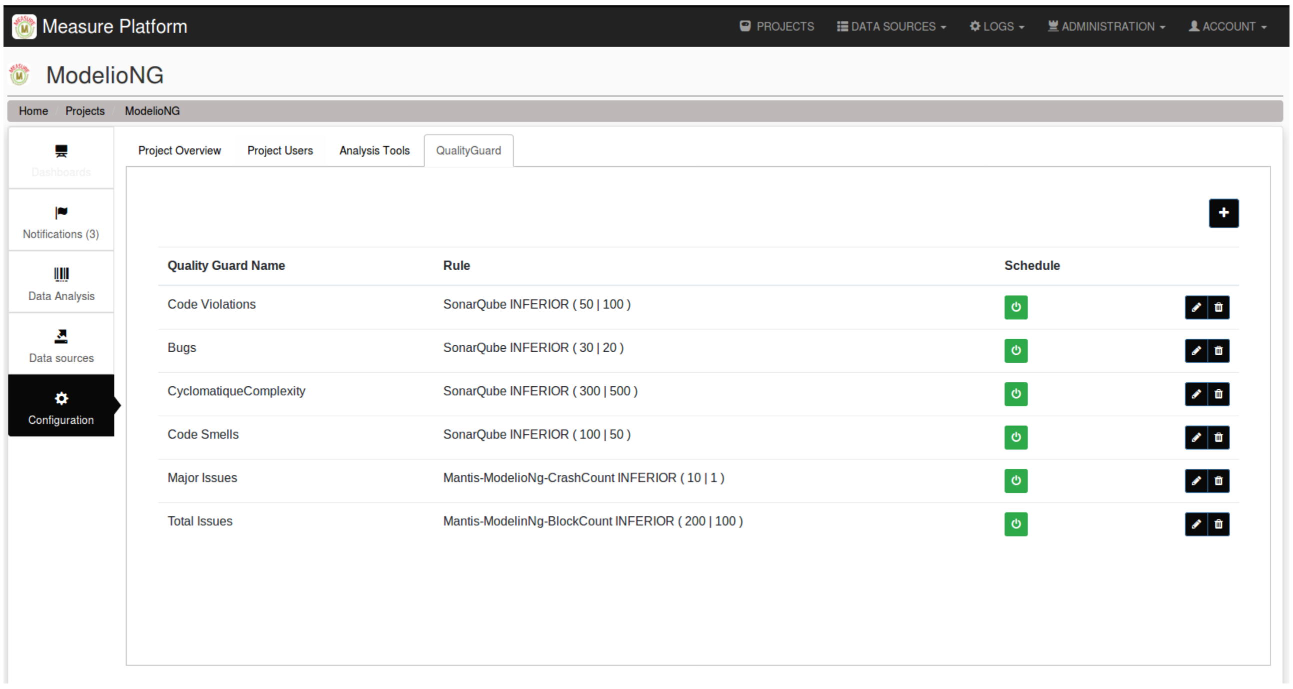
Figure 5. Suggestion results

The analysed data are data from evaluations history on the software corresponding to specific events on the code. As the Figure2. is showing, different suggestions are proposed with different number of metrics in the suggested measurement plan.

Integration of the Quality Guard Analysis Tool to detect defects during the development process

The Quality Guard contains rules specifying threshold values that specific measures should not reach. These threshold values were deduced form historical measurements conducted earlier in the first scenario but were occasionally adjusted later according to the development team priority. When the development team is notified by the existing of such issues, they will act upon these signals by trying to make corrections in specific development phases to keep the measures within the acceptable range in order to avoid development incidents.

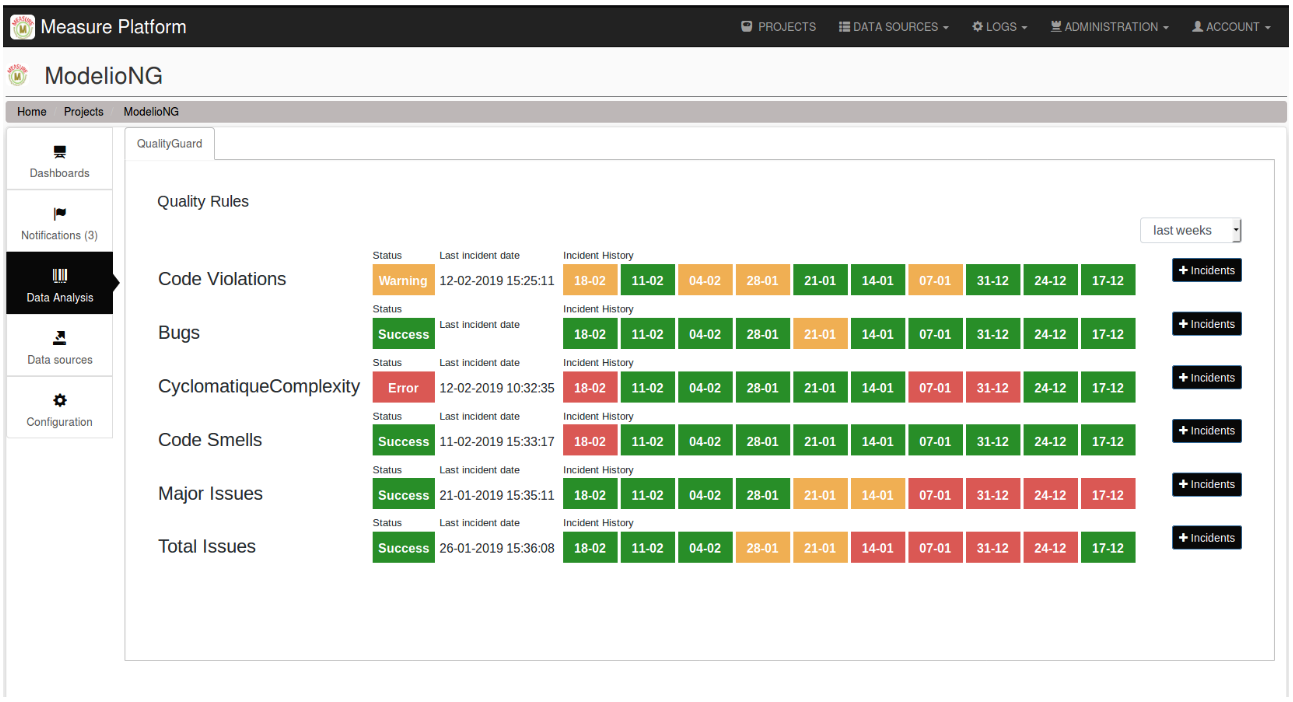
The figure below shows a list of rules defined in quality guard. The first rule is related to “Mantis registered issues” specifically blocking issues and issues that lead to crashes. This rule indicates that the platform will send a warning notification if the number of crashes exceeds 1 and will send an error if the number of crashes exceeds 5. It will also send a warning if the number of major issues exceeds 10 and an error if the total number of issues, regardless of their severity level, exceeds 100.



Quality Guard Name	Rule	Schedule
Code Violations	SonarQube INFERIOR (50 | 100)	
Bugs	SonarQube INFERIOR (30 | 20)	
CyclomatiqueComplexity	SonarQube INFERIOR (300 | 500)	
Code Smells	SonarQube INFERIOR (100 | 50)	
Major Issues	Mantis-ModelioNg-CrashCount INFERIOR (10 | 1)	
Total Issues	Mantis-ModelinNg-BlockCount INFERIOR (200 | 100)	

Figure 6 : Configuration of the Quality Guard Analysis tool for Modelio 3.8 development process

As expected, we 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 corresponding to a validation phase of the components delivered by the development team, 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 are 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

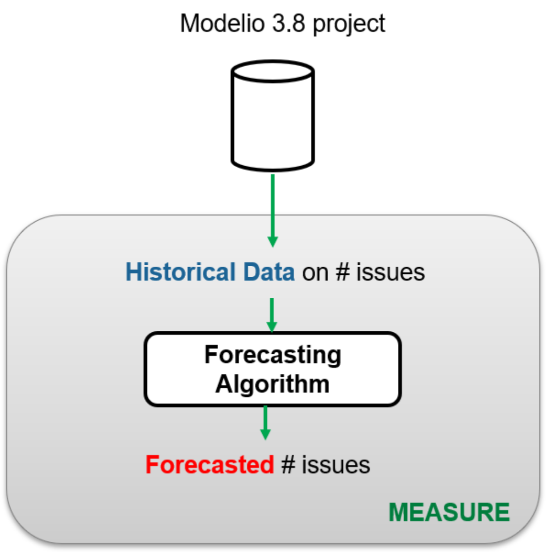
Figure 7 : Quality Guard Analysis from December 2018 to January 2019

The Measure Platform keeps a history of the registered incidents containing the violated rules and the time at which the violations occurred as it is illustrated in the figure below. To estimate the gained cost by using "Measure Platform" we identify registered incidents, examine the current values of these measures from the past weeks to forecast the probable evolution of these measures if we have not been notified by the platform and then we deduce the amount of extra work saved by using the platform.

Stracker is an analysis tool whose aim to increase the quality of software development by tracking and suggesting values for numerous software metrics during the development process and includes a module that predicts future metric values based on values recorded so far.

In context of the evaluation of the MeasurePlatform and with the help of the University of Bucharest, Softeam used the prediction module provided by Stracker to forecast metrics evolutions based on the historical values with the aim to try to evaluate the financial gain enabled by the early discovery of anomalies in Modelio 3.8 development process. This estimation of the cost of fixing the issue later would assume that the issue had not been discovered yet and the evolution of the quality indicators will be predicted using a forecasting tool.

Once a quality alert about our development process were triggered by the Quality Guard tool, we have initiate immediately mitigation action to solve the issues at the origin of this alert. The effort required to conduct this mitigations actions has bean recorded in project development plan. To estimate the gained cost by using "Measure Platform" we identify registered incidents, examine the current values of these measures from the past weeks to forecast the probable evolution of these measures if we have not been notified by the platform and then we deduce the amount of extra work saved by using the platform.



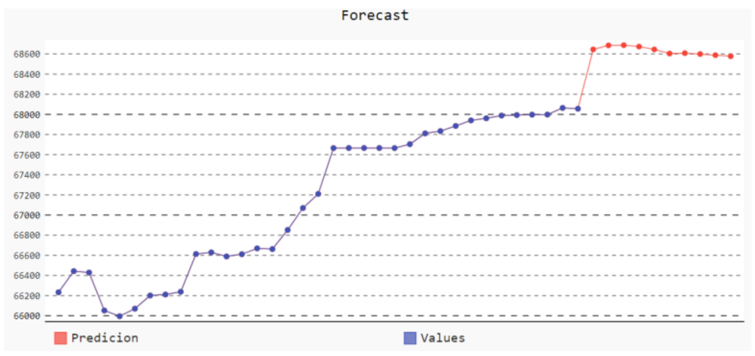


Figure 8 : Forecasting service apply to Modelio 3.8 issue count measure

At this time, the evaluation of the financial gain has been performed by the project manage based on his expertise and so accuracy of the estimation can be questioned. In a future release, Stracker plan also to implement a service to estimate the gain in cost brought by the monitoring tool by calculating the difference between the cost of solving the issue at early stage thanks to the monitoring tool and the estimated cost to discover and fix the issue later.

3.4. Results Analysis

Firstly, we used the Measure Platform and Analysis tools to analyse a large set of historical data by deploying measures addressing a large set of project management tools involved in Modelio development process. We have collected successfully data from commercial tools like SonarQube, OpenProject, Jenkins or Mantis and from tools more specific to our Case Study like our internal testing framework or Hawk. Next, by comparing data collected by the Measure platform and the backlog of the Modelio 3.8 development process, we found that the deployment of the Platform Measure on this period would have enabled us to detect much earlier two incidents of development that occurred during the project.

By comparing the remediation cost of this incidents effectively measured to an estimation of the cost to solve the same incident if it has been detected earlier with the help of the Measure Platform, we obtain the fooling result:

(Remediation Cost - Estimate Remediation Cost with Measure Platform) * Daily Rate

	Remediation Cost	**Estimate Remediation Cost with Measure Platform**	**Total**
Workflow Feature Issue	30 m/d	10 m/d	20 m/d
Workspace Creation Issue	15 m/d	3 m/d	12 m/d
Daily Rate	800 €	800 €	25600 €

We can conclude the deployment and exploitation of the Measure Platform on Modelio 3.8 development process during the period between the 05/2018 an the 10/2018 would have allow us to save nearly 25 600 €

Regarding the second evaluation scenario which has been conduct during the period between the 11/2018 an the 01/2019, we see that the usage of the Measure Platform, in combination with some of the analysis tools help us to detect early incidents that occurred during the development process of our product: Modelio 3.8. Even if the exact evaluation of the financial gains is difficult to realize because relying on the use of a prediction tools, we estimate that these are of the same order of magnitude to those estimated during the course of the first evaluation scenario.

A Qualitative Analysis of impact and usefulness of the Measure Platform deployment and integration in Modeliosoft development team complements the results presented above. These qualitative results, aggregated with those of the other case studies, are presented in the general conclusion of this documents.

4. Use Case: Large Naval Software System environment by Naval

4.1. Use Case Context

Naval Group (formerly known as DCNS) is a French industrial group specialized in naval defence and marine renewable energy. The group employs more than 13,000 people in 18 countries. Naval Group, a private law company in which the French state holds a 62.25% stake, Thales 35%, the personnel a 1.80% stake and the company itself 0.95%, is the heir to the French naval dockyards and the Direction des Constructions et Armes Navales (DCAN), which became the DCN (Direction des Constructions Navales) in 1991, DCNS in 2007 and Naval Group since 2017.

Naval Group’s activities can be broken down into two main sectors: naval defense, the group’s historical core business (ships, submarines, operational readiness management of the forces), and energy and marine infrastructures (renewable marine energies, civilian nuclear energy, construction of naval bases and electric power plants).

Naval defense remains the main activity of the group, in which it designs, develops and manages the operational readiness of surface and underwater naval systems and of their associated systems and infrastructures. As a project manager and integrator of armed vessels, Naval Group intervenes all along the value chain, from strategic program planning, to design, construction and the management of operational readiness.

The group works with the French navy and other navies, for conventional products, and with the authorization of the French government. It also offers its military expertise to the French Air Force to design automated navigation and combat systems, and to renovate aircraft.

<u>Specific issues of the CMS software</u>

The software called Combat Management System (CMS) is the heart of the Combat System (CS). The goal of the CMS is to enable the Command Team to manage the ship's CS resources in order to achieve the military objectives in an effective way. It is composed from redundant calculators, simulators, recording units and multi-function consoles.

The CMS interfaces the Combat System sensors, weapons and subsystems in order to provide facilities to assess the tactical situations effectively and assists in the planning and the execution of activities in support of the overall operational tasks of the ship. It collects, correlates, associates, evaluates and displays information coming from the ship's own sensors as well as from others ship’s sensors through the tactical data links. These facilities are used by Operators in the Operations Room.

CMS are part of the family of complex software systems, which require specific tools, solutions and methodologies to help mastering this complexity.

A CMS is software requiring tenths or even hundredths of man-year development time. The software is usually made of many millions of lines of code. This often leads to long or very long elongation of the development lifecycle (up to 10 years) and so to increased risks of planning drifts.

Because of such atypical elongation, NAVAL GROUP is faced to an unusual issue that is, to develop systems on a time scale that is far superior to the technology time-scale. This problem is even worth when taking in account the maintenance time which can be as long as thirty or forty years.

To help mastering such complexity, tools and solutions from software engineering are applied, such as:

- Component model;

- Iterative approach;
- Continuous integration;
- MDA (Model-Driven Architecture);
- Middleware agnostic application.

The following elements can benefit from the MEASURE project :

Metrics over the model

- Naval Group concentrates great development efforts over the modelling stage rather than over the code. Projects architecture, global conception, detailed conception and code are stored into the model.
- Naval Group has created its own components model called “JACOMO”.

Thus the need is to have metrics to characterize this important modelling stage. On the other hand, being able to create our own metrics is essential to evaluate the JACOMO component model as it is specific to Naval Group.

Deployment in the CI (Continuous Integration) process

Due to the complexity of developed software, the production process is massively automated. Thus it is required for the MEASURE platform to be compatible with our CI process. The aim is to be able to automatically and regularly fetch metrics to ensure a tight monitoring of the project’s activity.

In Naval Group’s context, cost for defect investigation and correction grows exponentially the more we go further in the V-model : A defect detected while peer reviewing an architecture isn’t expensive, while a defect detected during sea trials is extremely expensive (hard to schedule, constraints while working on the embedded system,…)

Automated alert tools

Due to the potentially great amount of measurements which will permanently be executed, we need to automate metrics analysis and alerts raise, and also have a synthetical presentation of results.

Metrics and indicators are of great interest, as they can greatly help measuring, assessing and improving the software quality and so avoid non-quality defaults that could compromise the respect of cost and delay constraints. Management targeted metrics in complement allow detecting as soon as possible drift on the planning; allowing to take measures ensuring the situation is kept under control.

MEASURE platform’s environment

Naval Group’s software development environment contains servers which provide a set of services for several projects. Available services list includesautomated build, continuous integration, code analysis, “wiki-like” platforms, test campaigns management systems, requirement management software, issue tracker, etc.

The MEASURE platform has been interfaced with a Java project developed using the “model-driven engineering” methodology. The content of this project is modeled using the Modelio tool, and Java code is automatically generated from the model. The software thus produced is then tested by the developer on his workstation, fixes are made on the source code and reinjected into the model. Once the expected result has been obtained, the model is “pushed” on the project’s SVN server. This “push” triggers the execution of several actions on the Continuous Integration server (SonarQube analysis, automated tests …).

The Modelio project is versioned using a SVN server. Softeam provided us the “Hawk” tool and several elementary metrics to fetch information from the project’s model.

The project's architecture is expected to follow identified patterns that can be checked using derived metrics based on elementary metrics provided by Softeam.

These derived metrics provide ratios which shall tend to specific values regarding the architecture's design. If a derived measure moves away from the target value, it helps us identify a drift from the architecture design in the model's implementation.

To this day, the supervision of the derived measures is manual. We work with a dashboard on the MEASURE platform which shows us our metrics values; periodically, an operator shall verify that all derived measures outputs are still in an acceptable range. Due to the heterogeneity of the target values and tolerances, this approach is prone to human errors. Plus, the necessity for an operator to manually process all the verifications each time a model change is made implies that these verifications aren't done in real-time but on a frequency which depends on the operator's availability.

To improve our workflow and shrink delays between the commit of an erroneous model and the corrective action on it, we must automate the surveillance task.

This automation is made though the "Quality Guard" tool.

4.2. Evaluation Approach

4.2.1. Analysis tools

Naval Group aimed at deploying the "Quality Guard Analysis Tool" provided by Softeam on its platform.

The Quality Guard Analysis Tool is an application which allows to provide analysis functionalities over data collected by the MEASURE platform. These analysis services are external application integrated to the platform as components.

The integration is based on 2 principles: the platform provides the REST API which allow the analysis tool to access to the platform data, and the analysis tool provides specific web pages which are directly embedded to the Measure Platform web application.

On the "project configuration" page of the MEASURE platform, the operator has the ability to bind the analysis tool to the project and configure "quality guards", which are a set of rules to apply on selected project's metrics.

Each rule of a quality guard is defined by :

- The measurement on which it will apply ;
- The comparison operator ("INFERIOR" and "SUPERIOR");
- The "Warning" trigger value;
- The "Error" trigger value;
- The time range on which to apply the rule (either on every measurement or on an average value over a minute/hour/day/week/month).

When declaring more than one rule in a Quality Guard, the tool also asks us a combination operator for the rules ("AND", "OR").

4.2.2. Evaluation Scenarios

At Naval Group, we expect to deploy the Quality Guard analysis tool to supervise the derived metrics we developed to monitor our projects design and warn us when a drift is happening.

To validate this requirement, we will deploy the Quality Guard Analysis Tool on an already running instance of the MEASURE platform of our production network, and then add Quality Guards with rules on fetched metrics.

To test alert raises an erroneous value will be simulated through a manual variation of warning/error floor values in Quality Guard rules.

Tests will be executed in the following order;

1. Scenario 1: Deployment: This scenario will focus on the tool deployment in our C.I. production environment:
    a. Deployment must be possible in Naval Group's production environment:
    As our environment is very constrained, the tool must be able to easily deal with those constraints and integrate fluently in our workflow.
2. Scenario 2: Integration: This scenario will focus on interfacing the tool with metrics used in our CI production environment:
    a. Integration with metrics provided by the MEASURE project:
    It is mandatory for the tool to work with "standard" metrics provided by the MEASURE project. This step will allow to check whether the tool is working or not in a default environment.
    b. Integration with metrics developed by Naval Group:
    As explained above, we developed derived metrics to cover our specific needs. The improvement provided by the Quality Guard Analysis Tool mostly comes from the monitoring of these metrics.
3. Scenario 3: Exploitation: This scenario will focus on the presentation and capitalization on data provided by the tool.
    a. Synthesis of Quality Guards current state and history:
    Results of the analysis shall be easily accessible : they shall provide the current state of quality guards applied on a project and a mean to easily synthetize alerts history (either over the whole project or on specific quality guards only).
    b. C.I. environment users must not be impacted by the usage of the tool:
    We must minimize the impact of the tool over the production workflow (no additional actions to be manually done by environment users).
    c. External trigger on alert raise:
    The tool shall provide a mean to transmit information to other components of the CI toolchain to be able to forward messages to operators, managers, log systems,…

These scenarios will be executed on our production environment's MEASURE platform. They follow a logical order regarding the deployment of the tool on a standard project.

4.3. Experimentation

4.3.1 Scenario 1: Deployment

Deploying the Quality Guard Analysis tool is simple and straightforward. We only have to unzip the release archive and edit the "application.properties" file it contains to match the MEASURE platform's configuration.

The only minor problem we had when deploying the tool was due to one of our auto-start scripts (to automatically launch the MEASURE platform and its dependencies on server startup). This script was launching the application from the wrong working directory, thus resulting in an invalid configuration as the configuration file wasn't found on the current working directory.

4.3.2 Scenario 2: Integration

Once started, the tool interacts with the platform and we can see it on the platform's "Analysis services configuration" page.

To be able to use the tool in a project, we must bind the tool to it through the "Project Configuration" page. This action results in the addition of a "Quality Guard" tab on the project's page.

When we get in the project's "Quality Guard" tab, we are then able to declare "quality guards" and the rules they shall monitor.

In this use-case, we defined that our metrics can vary in a range of +/- 2 % around the target value without triggering an alert, but a variation of more than 20 % shall trigger an error.

To this point, we came across an important limitation of Quality Guard Analysis Tool's version 1.0.0: Rules triggers were using only integer values, which are not flexible enough to work with our target trigger values. Indeed, our target values can be in a range between 1 and 3; this implies that warning value can be of 0.98 for a target value of 1 for example. In this context, the tool was truncating the warning value to 0 when creating a new rule.

We declared this problem to Softeam which quickly released a 1.0.1 version of the Quality Guard Analysis tool to fix this limitation and allow us to use floating point values when declaring new rules.

This led us to declare quality guards over our four derived metrics:

- Quality Guard 1 :
    - Base metric : Component Services over Component Events
    - Target value : 2
    - Acceptable variation (warning floor) : ± 2 % → [1.96 : 2.04]
    - Erroneous variation (error floor) : ± 20 % → [1.6 : 2.4]
- Quality Guard 2 :
    - Base metric : Number of Component Types over Number of Entities
    - Target value : 3
    - Acceptable variation (warning floor) : ± 2 % → [2.94 : 3.06]
    - Erroneous variation (error floor) : ± 20 % → [2.4 : 3.6]
- Quality Guard 3 :
    - Base metric : Threading ratio

- Target value : 1
- Acceptable variation (warning floor) : ± 2 % → [0.98 : 1.02]
- Erroneous variation (error floor) : ± 20 % → [0.8 : 1.2]

- Quality Guard 4 :
  - Base metric : Configurability ratio
  - Target value : 3
  - Acceptable variation (warning floor) : ± 2 % → [2.94 : 3.06]
  - Erroneous variation (error floor) : ± 20 % → [2.4 : 3.6]

After creating those quality guards, we realized alerts weren't triggered : whereas floor values were exceeded by far, Quality Guards still were showing a "Success" status message. Softeam identified an error in our Quality Guard tool configuration which caused it to be unable to find ElasticSearch.

Once this configuration problem solved, we've been able to monitor both "standard" and Naval Group specific metrics and see variations in Quality Guards states.

4.3.3 Scenario 3: Exploitation

To be able to monitor project's current state, we used "Analysis Card" visualisations provided by the Quality Guard Analysis Tool.

We deployed both "Quality Status" and "Quality Issues" visualisations and verified the coherence of retrieved data.

We found out that when a quality guard was triggered, alerts were raised for each of its rules, even though they weren't violated (for example when having two rules to monitor a measurement which shall be "greater than" and "lower than" two specific values, both "greater than" and "lower than" rules are displayed on the Quality Issues visualization. We transmitted this issue to Softeam.

These visualisations were quite handy to have on the "Project Overview" page, but find their limitations through the usage of scroll bars whereas there's only a few Quality Guards declared (< 10) ; this presentation let us prefer to use the "QualityGuard" tab instead. It would have been more convenient in overview dashboards to have a more concise/dense presentation of results to apprehend the global state of the project at a glance.

Regarding actions needed by the operator to fetch information from the Quality Guard Analysis Tool, automation of its processing makes it "non-invasive" to use: It is a "configure and forget" approach, which doesn't require manual operation to generate reports. The only limitation we found to the tested version is related to data *access* which requires the operator to go on the platform to visualize information. Indeed, in CI environment like ours, it would have been a good feature to have a mean to trigger external actions when an alert is raised, at least to instantly inform concerned operators about the alert, instead of waiting for them to go on the platform.

4.4. Results Analysis

4.3.4 Scenarios validation synthesis

Scenario 1: Deployment

Deployment must be possible in Naval Group's production environment	✓

Scenario 2: Integration

Integration with metrics provided by the MEASURE project	✓
Integration with metrics developed by Naval Group	✓

Scenario 3: Exploitation

Synthesis of Quality Guards current state and history	⍻
C.I. environment users must not be impacted by the usage of the tool	✓
External trigger on alert raise	✗

4.3.5 Naval Group use case synthesis

This analysis tool can be of great interest but might need more maturity to provide its full potential. Its exploitation is still not sufficiently ergonomic 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.

4.3.6 Encountered difficulties

<u>Invalid Quality Guard configuration without any warning/notification on the platform</u>

The most critical difficulty we encountered was related to the configuration of the tool; we did some mistakes over its initial configuration, which resulted in its incapacity of interacting with the ElasticSearch database. This could have been of a minor gravity, but QualityGuard was not showing anything but "valid" status over the MEASURE platform interface or on its tool webpage, letting us think of problems other than then base configuration, as the tool seemed to interact correctly with the platform, which led to too much waste of time for both Naval Group and Softeam which helped us investigate the problem.

It would be interesting to have a basic tool status (and maybe a few log lines) to be displayed over the tool webpage which is currently unused, and also trace back a "degraded" status up to the MEASURE platform to quickly identify an issue with analysis tools.

<u>Limitations through alert floor values type</u>

Alert floors were declared as integer values, which was very limiting as most of our metrics are using float values. This problem has been solved by Softeam through the v1.0.1 release of the QualityGuard tool.

<u>Missing operators of comparison</u>

The Quality Guard rules are based on usage of "comparison operators". Currently two comparison operators are available : "INFERIOR" and "SUPERIOR". Unfortunately, it is not possible to identify whether it is a

"STRICTLY INFERIOR/SUPERIOR" or "INFERIOR/SUPERIOR OR EQUAL" comparison, which can be meaningful depending on the metric it applies on.

We suggest replacing the two current operators by the four operators following:

- STRICTLY INFERIOR;
- INFERIOR OR EQUAL;
- SUPERIOR OR EQUAL;
- STRICTLY SUPERIOR.

Display integration bugs

Several pages contain display bugs. For example, on MEASURE platform's page to configure QualityGuard for a project, when declaring several rules on the same QualityGuard leads to have longer description lines which overflows the width of the screen and makes "schedule", "edit" and "delete" buttons disappear on the right side. As a result, an horizontal scrollbar appears but it is only visible when the vertical one is completely scrolled down. Thus when we have loads of QualityGards on the project, it is really harsh to access "edit" buttons for example, as we have to find the Quality Guard name on the left side of the table, then scroll down to the bottom of the table to access the horizontal slider, scroll completely to the right –which makes the Quality Guard names disappear-, then scroll up and try to find out where the Quality Guard we wanted to edit is, which tend to be a guess as we can't have both edit buttons and names at the same time.

It would be way more convenient either to avoid having an horizontal slider by using multiple lines for rules, or have the horizontal slider accessible at any time.

Limitations of currently available visualizations

Currently available visualizations find their limitations through the usage of scroll bars whereas there is only a few Quality Guards declared (< 10); as a consequence, operators cannot have an instant overview of the project state when logging on a dashboard (they have to scroll pages).

It would have been more convenient in overview dashboards to have a more concise/dense presentation of results to apprehend the global state of the project at a glance. For example, a "GitHub-like" configurable heat map might have been more convenient. We could think about several kinds of visualisation based on heat maps: For instance, we could have a heat map which shows all current Quality Guards states as little coloured squares, and putting the mouse on one of the squares would show us the name of the Quality Guard. We could also think of a heat map to show violations history, either for one Quality Guard or for the whole list (see figure below for example).

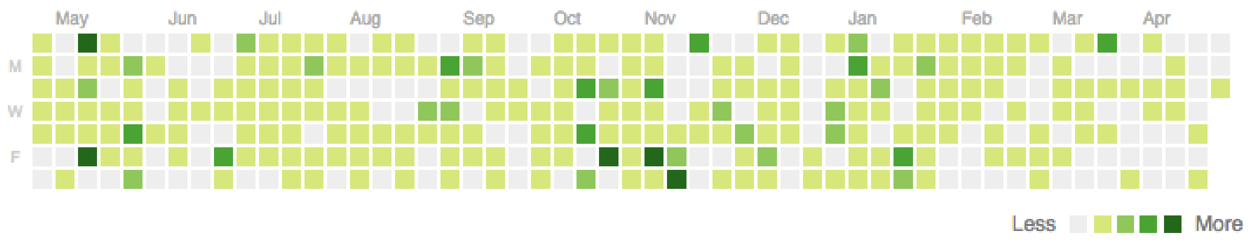


Figure 9 – History heat map example from Github

Improvement: Tool statistics

As CI environment servers often host multiple applications, it is interesting to have one page on the application dedicated to provide information and statistics about the tool.

For example, the Quality Guard Analysis tool's main page could show: name and version of the tool (the latter being very important to us), state of links between the tool and other component (MEASURE platform, ElasticSearch …), JVM statistics, tool statistics: how many rules are available/activated/periodically checked, how many checks are performed per minute/second/other and/or last minute, a graph showing checks count for the last period (minute/hour/day/week/month…) would also be handy, last detected errors, etc.

5. Use Case: Security analysis case study by BITDEFENDER

5.1. Use Case Context

5.1.1 Bitdefender

Bitdefender is a Romanian large company and the creator of one of the world's fastest and most effective lines of internationally certified internet security software. The company is an industry pioneer, introducing and developing award-winning protection since 2001. Today, Bitdefender technologies secure the digital experience of over 500 million home and corporate users across the globe. Top international testing organisations and world-renowned software reviewers acknowledge Bitdefender as the world's most effective anti-malware solutions. Following the world recognition gained in January 2014, when Bitdefender has won both awards from AV-Test, becoming the first solution to be named "Best Protection" and "Best Performance" at the same time, in 2015 Bitdefender won again "Product of the Year" awarded by AV Comparatives. During recent years, Bitdefender's latest line of products also won the title of "Best Antivirus" and two Editor's Choice awards from PC MAG, and CNET has also named Bitdefender its Editor's Choice, that confirmed its leadership status among security products. Through R&D, alliances and partnerships, Bitdefender is trusted to be ahead and deliver robust security you can rely on.

5.1.2 Security Analytics Technology

To uncover and stop elusive threats Bitdefender developed Security Analytics technology, one of the core technologies that stands behind GravityZone Business Security. Key advantages of the GravityZone solution are:

- protection that do not trade performance, consistently ranked in top of independent tests by AV-Test and AV Comparatives
- easiness of installation and usage, users can focus on their day to day business and let the solution to be installed in minutes and managed by users with no extensive IT background
- cloud based solution that is saving money, no additional hardware is needed, business users need to move to cloud administration of current endpoints

Gravity Zone Ultra, the maximum protection package from Bitdefender, integrates Security Analytics technology to deliver next-gen endpoint protection and easy to use EDR platform. EDR technologies complement much of the existing security solutions (AVs, SIEMs, NFT-Network Forensics Tools, ATD-Advanced Threat Defense), being able to provide a richer depth of behavior-based anomaly detection and visibility into endpoint specific information, relevant for detecting and mitigating advanced threats. The EDR tools are able to record many detailed end-point and network events and store this information in a centralized database for deep detection, analysis, investigation, reporting and alerting.

5.1.3 Security Analytics Team

For the use-cases presented by Bitdefender MEASURE Platform has been deployed in Bitdefender Security Analytics development team. The team is composed of 15 people and includes various profiles like Product Manager (1), Software Engineering Manager (1), Technical Leads (2), Software Developers (6), Security Researchers (6) and Quality Engineers (1).

Special instances of MEASURE platform, GravityZone have been deployed and used for the use-cases. Metrics from internal tools (like Sonar) were collected and integrated into local MEASURE Platform.

In order to integrate Security Analytics measure and evaluate the platform Software Engineering Manager, Technical Leads, Software engineers and Quality Engineers performed various tasks like:

- defining use-cases
- developing measure specific integration software development activities
- evaluating developed and platform available metrics (like SonarQube metrics)
- developing tools to showcase certain software development security-sensitive scenarios
- developing automated scenarios to simulate in-production cyber-attacks

5.1.4 Reason to integrate Security Analytics in MEASURE

Being a global technology security, Bitdefender is interested in protecting any customer not only in production but also when developing a product or a service. Thus, MEASURE Platform is a great opportunity to provide our expertise and our services. Nowadays there are a lot of security incidents happening around the world and with them there are also a lot of security challenges for many companies. Bitdefender Security Analytics solution is integrated in MEASURE Platform in two ways

- Testing/Development mode - where Security Analytics technology can help developers or managers to identify security issues before releasing technical solution
- Production mode - where Security Analytics technology is monitoring Windows based devices, where the solution is installed, to uncover critical security issues related to exploits, data exfiltration and command and control communications

MEASURE Platform provided the opportunity to Bitdefender to add the security perspective on software development. Security Analytics metrics are adding another security perspective focused on testing and production monitoring activities and RIVER, that was integrated into the platform last year using concolic execution.

Challenges for MEASURE:

- testing tools are platform specific - for example Security Analytics technology is available only on Windows OS, but Bitdefender is making consistent efforts to come up with solutions also on Linux and MacOSX
- metrics provided by Security Analytics are dependent on Gravity Zone solution installation available in client’s environmental
- Security Analytics metrics development and integration was not very easy due to lack of up to date documentation, but it was finalized successfully due to collaboration with other partners and existing examples (Softeam and Montimage) code adaptation in order to test concolic execution with RIVER.

5.2. Evaluation Approach

The purpose of this evaluation is to provide insightful information about security measures usefulness in MEASURE monitoring platform. The approach selected for evaluation was to make tools and experiments in order to see the gains and limitations of the integration in the platform.

Looking at the platform and Bitdefender tools, three evaluation scenarios were selected

- First scenario consists in analyzing an attack result metrics

- Second scenario consists in evaluating metrics resulted in testing a specially crafted executable with some potential security issues
- Third scenario consists in evaluating the platform and a software development project using its source code

In this evaluation we will measure the impact of MEASURE Platform, Bitdefender Security Analytics, RIVER and other tools available in the platform.

5.2.1. Analysis tools

Montimage

MMT tool is integrated in Measure Platform and can provide security metrics for production deployed solutions. It works at the network level and combined with Bitdefender's Security Analytics solution a customer will have fewer security gaps in security monitoring. MMT can also extract metrics regarding network usage and status because it also monitors this area. Bitdefender's test-case for this evaluation is targeting endpoint level attacks but without doubt a network level solution will give relevant information. For example, a brute force attack vector to a public server, lateral movement and command and control connections would have been detected by MMT technology. In the use-case presented below a spear-phishing email was used for initial foothold but immediately after the user clicked on the link victim machine was connecting to the command and control server. If the attacker is very lucky the victim would have compromised the most valuable security asset from start, but usually this is not the case and the attacker must try to discover new victim machines from the local network and move laterally. MMT technology is capable of recording information related to LAN and WAN networks.

In terms of detection an attack collaboration using Measure Platform is very useful but if a user has or needs forensics capabilities there should be another level of integration between these two technologies.

Stracker

Stracker module is a very good idea of how metrics can be analyzed with minimum user intervention and it provides great value to a Measure Platform user. Being able to predict near future metrics value it helps the user to confront with the reality and make immediate adjustments if needed. In Bitdefender Stracker can be used mostly augmenting software development metrics with future predictions. A new level of metrics should be developed in order to summarize an overall trend for software development aspects that are currently being monitored. (Stracker was evaluated using toy data)

Mint

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 Mint can be used to monitor 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.

5.2.2. Evaluation Scenarios

The following scenarios were executed for evaluation purposes.

Scenario 1 - Performing an attack on a Windows device with Security Analytics installed
Actors

- Bitdefender Software Engineers

- Bitdefender Technical Leads
- Bitdefender Project Manager (Software Engineering Manager)

Current Workflow

- Security Analytics technology is currently integrated in Gravity Zone for installation, monitoring devices and investigating incidents

Measure Workflow

- integration in MEASURE Platform is used mainly for counting the number of incidents overall in the company
- we hope to add value by showing the number of incidents in monitoring the post production phase

Scenario 2 - Running a specially crafted executable in order to uncover security related information for software developers

Actors

- Bitdefender Software Engineers
- Bitdefender Technical Leads
- Bitdefender Project Manager (Software Engineering Manager)

Current Workflow

- no visibility on security related issues on running applications in test environmental

MEASURE Workflow

- visibility of security development related issues and possibility to investigate them in Bitdefender Security Analytics platform

Scenario 3 - evaluating software development metrics in MEASURE Platform

Actors

- Bitdefender Technical Leads
- Bitdefender Project Manager (Software Engineering Manager)

Current Workflow

- using internally deployed tools from Atlassian suite (Jira, Confluence, Bamboo, Bitbucket) and SonarQube

MEASURE Workflow

- viewing information related to software development project in MEASURE Platform

Below we provide the details of executing the scenarios.

5.3. Experimentation

5.3.1. Scenario 1 - Performing an attack on a Windows device with Security Analytics installed

Targeted cyber security attacks are sometimes rare events, especially for Bitdefender, but there are some tools to simulate some. In this evaluation scenario we have selected an attack that uses an Adobe Reader exploit (CoolType SING - CVE 2010-2883) and some techniques to exfiltrate password hashes and to migrate on a second machine using those password hashes. In this scenario a user must open a specially crafted document from a spear phishing attack and then the attacker will perform the rest of the steps. Of course, this is not a typical attack scenario for a solution deployed in production but spear phishing attack is only one of many possible attack vectors.

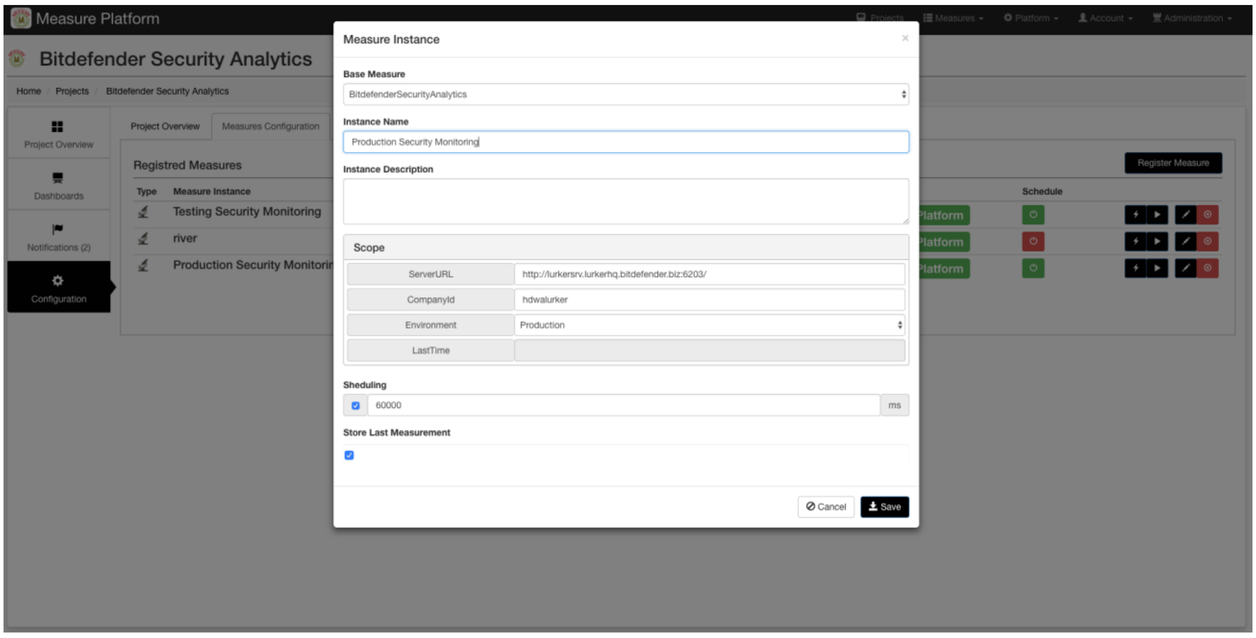
Tools and devices used

- attack script that will perform the attack steps and an Auto Exploiter tool. Auto Exploiter tool is an internal tool that is able to run steps from the attack.

- Metasploit user penetration test framework - a large part of the attack script are commands to be run in this framework. This tool is developed by Rapid7 and shared to the open-source community. Metasploit framework is a framework that is often used by attackers, there are also other frameworks available like Cobalt Strike, PowerSploit, etc
- 5 virtual machines
    - 1 Ubuntu 16.04 machine with MEASURE Platform installed
    - 1 Ubuntu 16.04 Attacker machine with Metasploit and Auto Exploiter tool
    - 1 Ubuntu 16.04 Security Analytics
    - 2 Windows 7 Victim machines with Adobe 9.x which is vulnerable to Adobe CoolType SING attack

How to configure

In order to receive metrics in MEASURE Platform there are some steps to be done prior running the attack Bitdefender Security Analytics metric must be configured accordingly. Using the IP or the domain name of Bitdefender Analytics and the port 6203, MEASURE Platform will gain access to incident data available, but because Bitdefender technology is multi-tenant a companyId is also needed. CompanyId is a special key generated for any company active in Bitdefender Gravity Zone solution.



Registred Measures

Type	Measure Instance
	Testing Security Monitoring
	river
	Production Security Monitorin

Measure Instance

Base Measure: BitdefenderSecurityAnalytics

Instance Name: Production Security Monitoring

Instance Description:

Scope

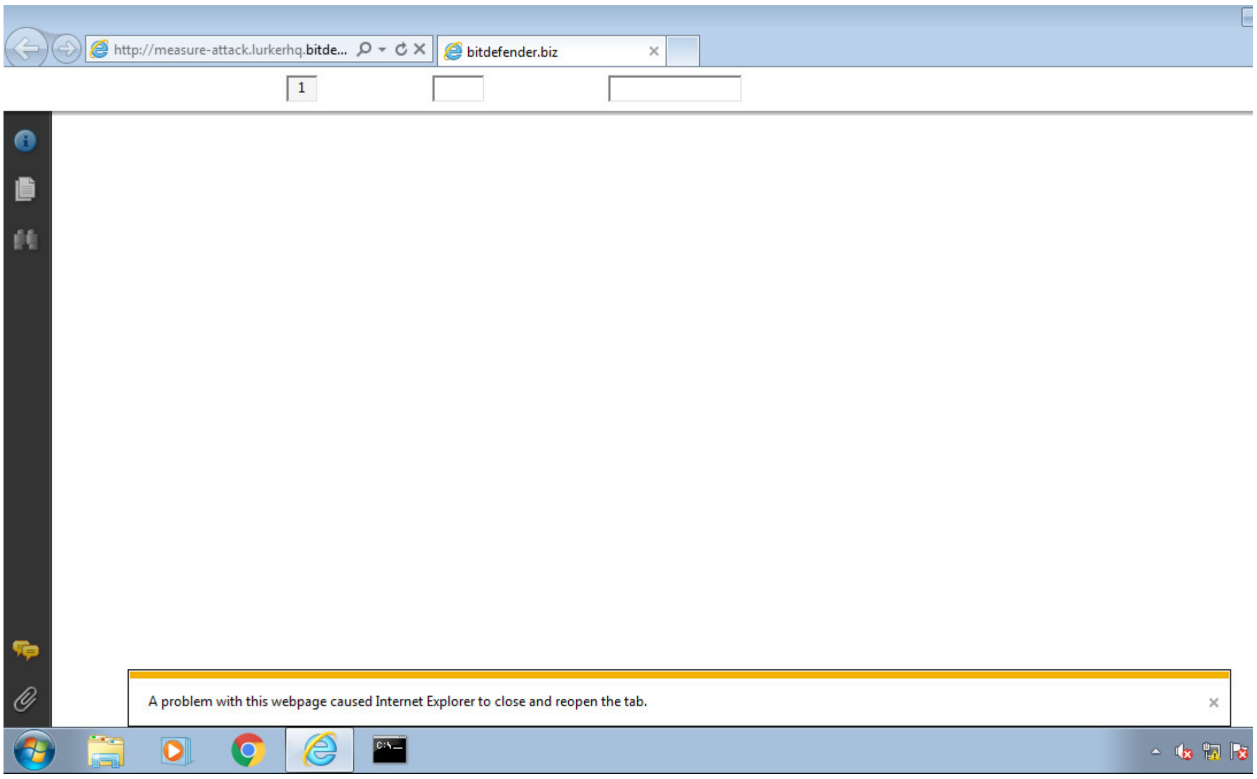
ServerURL	http://lurkersrv.lurkerhq.bitdefender.biz:6203/
CompanyId	hdwalurker
Environment	Production
LastTime	

Sheduling: ✓ 60000 ms

Store Last Measurement: ✓

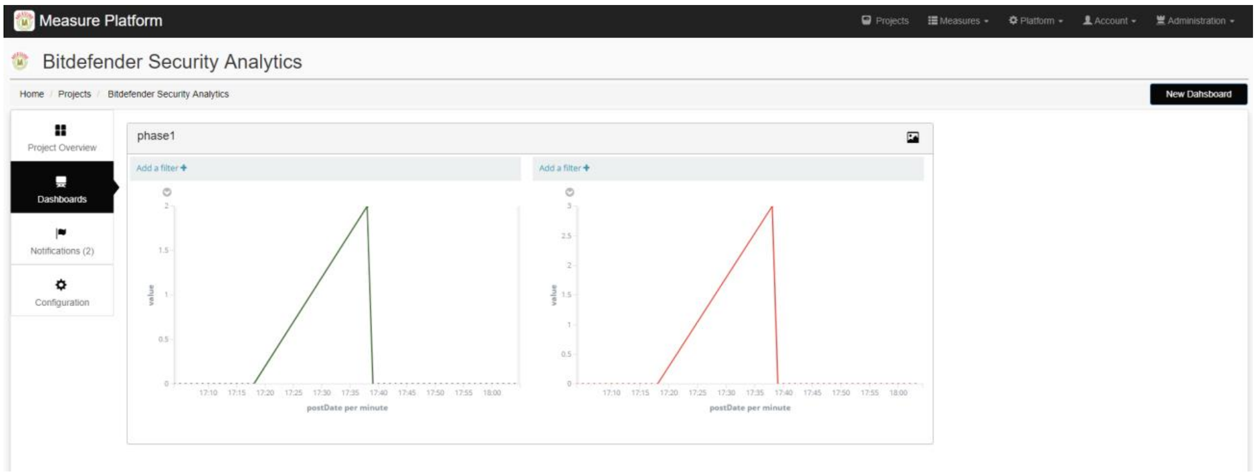
Run scenario

The attack is started by opening an adobe reader document stored on the attacker server and the page will cause a crash in user's browser.



A problem with this webpage caused Internet Explorer to close and reopen the tab.

The number of incidents can be seen in the platform and this attack generated a number of two incidents (left graph) and three suspicious events (right graph).



MEASURE Platform can be used in this case as a monitoring tool and it is useful for high level reports, but it is more meaningful to integrate multiple technologies that can add different perspectives. For example, Montimage tools, that are already integrated into the platform can provide information at the network level monitoring internal and external traffic. Bitdefender Security Analytics is a host-oriented detection and monitoring solution. In the security domain companies are collaborating by sharing information and using the same paradigm Bitdefender and Montimage can collaborate very easily using MEASURE Platform and helping a Security Operation Center to have enough high-level visibility. Of course, for deeper understanding of the attack and the damage caused the user must use actual tools that are generating data (or the platform must open an API to load content also from the tools not only from its Kibana instance)

5.3.2. Scenario 2 - Running a specially crafted executable in order to uncover security related information for software developers

Testing solutions before going to into production is mandatory and in many companies the focus of testing is largely distributed to testing user functionality and features and less focused on security. Bitdefender Security Analytics Technology can provide a way to test for potential security issues. For this use-case we have developed a proof-of-concept executable to show its potential. The executable is performing a few steps like: adding itself to startup registry key and in startup folder to be run using an encoded Powershell command.

Examples of security development information Security Analytics can provide

Security Analytics technology can provide information that is relevant to any software developer just looking at the executable behavior. Here are some examples:

1. Saving information on what command to execute to start a component of your application in a registry entry (file_drop + registry_exec)

If your application has an update component, this could function by downloading a new version of the application executable, closing the already running one, saving information on how to run it (command line arguments or the full command-line) in a registry entry, then restarting the system for some changes to take effect only to read the command-line from the registry after reboot and start the new version of the application. Such an approach is quite unsafe, since the registry entry might be modified by an attacker such that, besides the command-line of the updated application a malicious script might be executed. As such, dropping a binary file which was downloaded via HTTP, creating a registry entry that contains the path to the new dropped file and then, at a future point in time, creating a new process with that command-line is considered a security issue which should be looked into.

2. Ensuring you application always runs at system-boot by using a run registry (file_drop + added_to_reg_run)

If you have an application which needs to be started every time the OS boots, one simple way of achieving this on Windows is to have your installer process create a run registry with the path to your application and let Windows do the rest of the work. Such an approach is dangerous for a number of reasons: first of all the registry key might be removed (by executing a system clean tool or from direct user interaction) and second of all since this a preferred method abused by many malicious applications to ensure persistence - the original sample dropped on disk is packed and, when ran, it unpacks its payload somewhere in the file-system and adds that path to a run registry to ensure the actual malicious code gets executed at every boot.

3. Updating an application by using powershell scripts

If you want to create a simple update process for you application one means of doing this (on Windows) is to have a powershell script connect to the update location, download the new file, stop the existing application, replace it with the new one and start the updated version. In order to obfuscate your update script you might decide to encode it using base64 and then have powershell.exe decode it at runtime and execute thus obtained code. Such a method should be avoided since the script could be easily altered and you might end up executing a malicious powershell payload. Moreover, utilizing powershell or other already-available system tools is a method used by malicious apps to try and avoid being detected, since they do not drop any files on local disk, they simply utilize tools readily available on the system to achieve specific goals (for e.g. they use powershell to for C&C communication, reg.exe to add the C&C script at startup or net.exe to stop a security service or other legitimate applications).

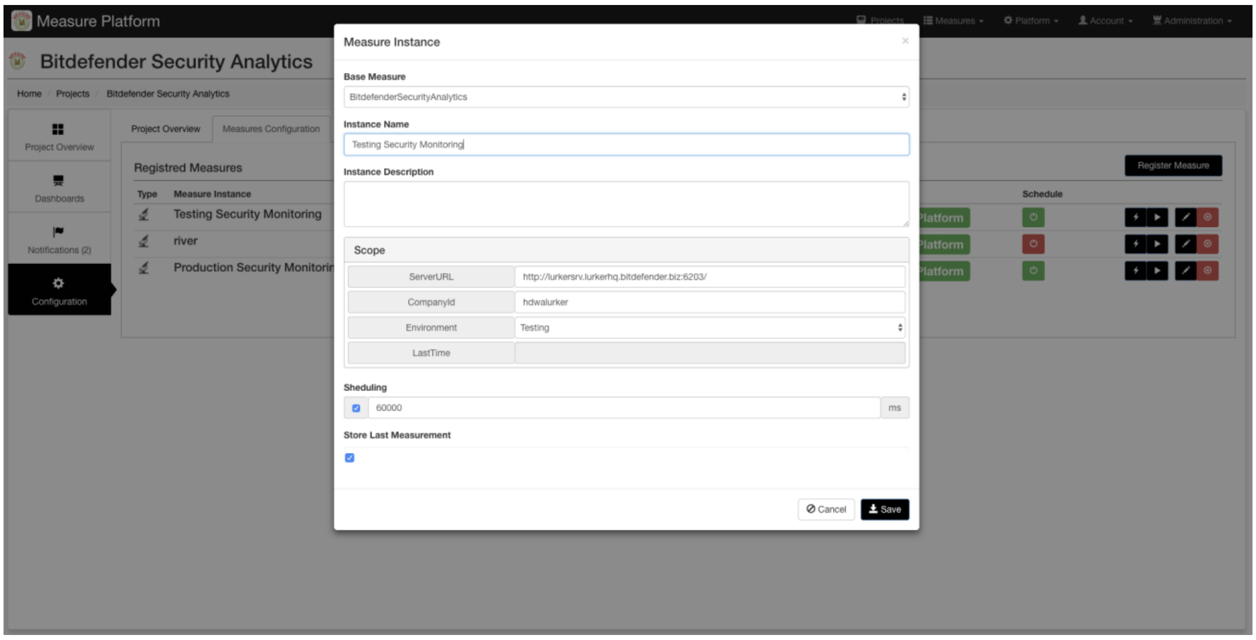
Tools and devices used

- “grayware” executable - proof-of-concept binary file
- 5 virtual machines
    - 1 Ubuntu 16.04 machine with MEASURE Platform installed

- 1 Ubuntu 16.04 Attacker machine with Metasploit and Auto Exploiter tool
- 1 Ubuntu 16.04 Security Analytics
- 1 Windows 7 machines

How to configure

In order to receive metrics in MEASURE Platform the configuration steps are very similar with the previous scenario, except that Environment field must be “Testing”.



Registred Measures

Type	Measure Instance	Schedule
	Testing Security Monitoring	
	river	
	Production Security Monitorin	

Measure Instance

Base Measure: BitdefenderSecurityAnalytics

Instance Name: Testing Security Monitoring

Instance Description:

Scope

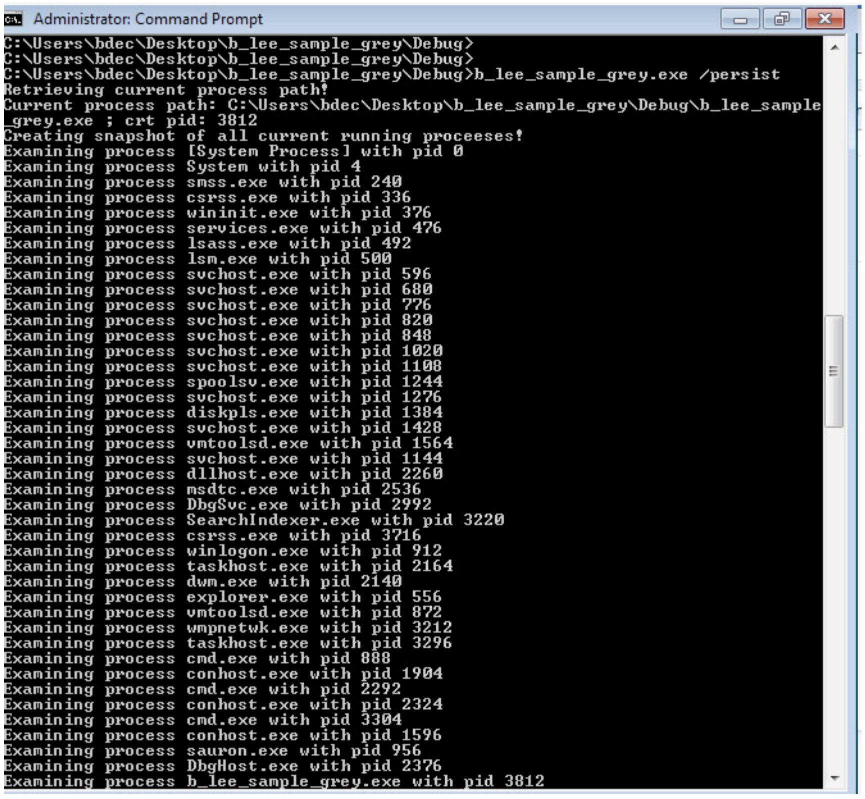
ServerURL	http://lurkersrv.lurkerhq.bitdefender.biz:6203/
CompanyId	hdwalurker
Environment	Testing
LastTime	

Sheduling: ✓ 60000 ms

Store Last Measurement: ✓

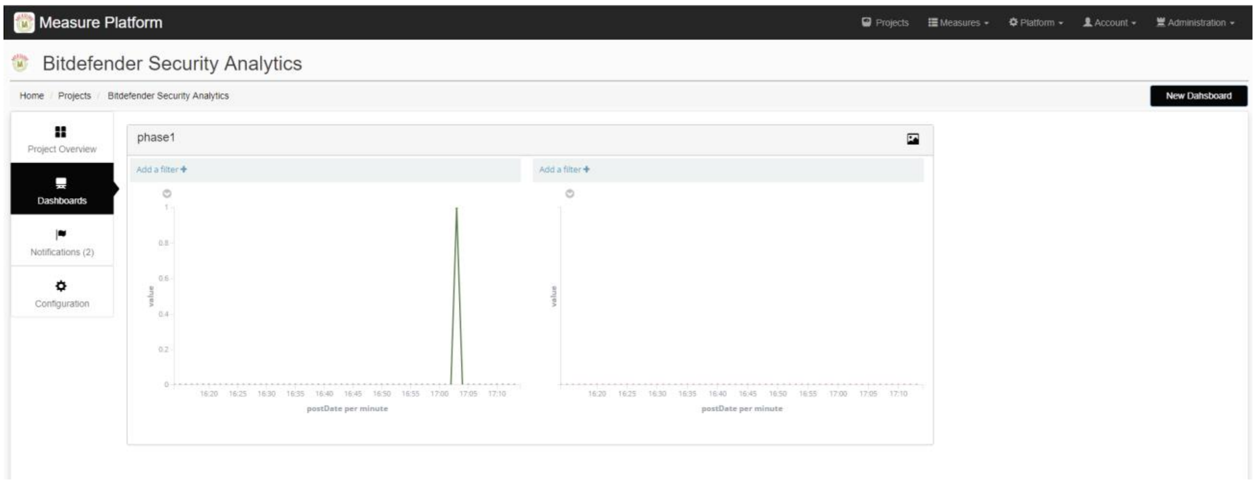
Run scenario

The executable must be run in a command prompt using administrative privileges.



```
C:\Users\bdec\Desktop\b_lee_sample_grey\Debug>
C:\Users\bdec\Desktop\b_lee_sample_grey\Debug>
C:\Users\bdec\Desktop\b_lee_sample_grey\Debug>b_lee_sample_grey.exe /persist
Retrieving current process path!
Current process path: C:\Users\bdec\Desktop\b_lee_sample_grey\Debug\b_lee_sample
_grey.exe ; crt pid: 3812
Creating snapshot of all current running proceses!
Examining process [System Process] with pid 0
Examining process System with pid 4
Examining process smss.exe with pid 240
Examining process csrss.exe with pid 336
Examining process wininit.exe with pid 376
Examining process services.exe with pid 476
Examining process lsass.exe with pid 492
Examining process lsm.exe with pid 500
Examining process svchost.exe with pid 596
Examining process svchost.exe with pid 680
Examining process svchost.exe with pid 776
Examining process svchost.exe with pid 820
Examining process svchost.exe with pid 848
Examining process svchost.exe with pid 1020
Examining process svchost.exe with pid 1108
Examining process spoolsv.exe with pid 1244
Examining process svchost.exe with pid 1276
Examining process diskpls.exe with pid 1384
Examining process svchost.exe with pid 1428
Examining process vmtoolsd.exe with pid 1564
Examining process svchost.exe with pid 1144
Examining process dllhost.exe with pid 2260
Examining process msdtc.exe with pid 2536
Examining process DbgSvc.exe with pid 2992
Examining process SearchIndexer.exe with pid 3220
Examining process csrss.exe with pid 3716
Examining process winlogon.exe with pid 912
Examining process taskhost.exe with pid 2164
Examining process dwm.exe with pid 2140
Examining process explorer.exe with pid 556
Examining process vmtoolsd.exe with pid 872
Examining process wmpnetwk.exe with pid 3212
Examining process taskhost.exe with pid 3296
Examining process cmd.exe with pid 888
Examining process conhost.exe with pid 1904
Examining process cmd.exe with pid 2292
Examining process conhost.exe with pid 2324
Examining process cmd.exe with pid 3304
Examining process conhost.exe with pid 1596
Examining process sauron.exe with pid 956
Examining process DbgHost.exe with pid 2376
Examining process b_lee_sample_grey.exe with pid 3812
```

Grayware executable has generated no incident (left graph) and one suspicious event (right graph). The suspicious event must be investigated in Security Analytics platform through Kibana UI.



Conclusion

Although limited, Bitdefender Analytics can provide information regarding security issues before releasing and having vulnerabilities in the market. Next and useful step will be to document different types of vulnerabilities that can be detected.

5.3.3. Scenario 3 - evaluating software development metrics in MEASURE Platform

The ultimate goal of the MEASURE Platform is to help software development companies to improve their capabilities. Peter Drucker, an American management consultant, educator and author, once said “*If you can’t measure it, you can’t improve it*”. Although many companies are using agile methodologies at the core most of them are using Software Development Life Cycle (SDLC) processes. In SDLC there are multiple stages like:

Planning, Defining, Designing, Building, Testing and Deployment. In order to remain competitive a software development company must improve its capabilities on each of these phases. MEASURE platform can help Software Engineering Managers or Project Managers to actively monitor and improve their current SDLC process or a sub-part of it.

In Bitdefender, the activity of evaluating MEASURE platform was performed in the organisation that delivers Security Analytics technology. This organisation is focused on delivering this technology but also creating experiments to ensure the future of current and next technologies. Thus, the activities performed are covering all phases in the SDLC process. In this scenario we will be focusing on using metrics and tools already available in MEASURE platform and also focus on security metrics.

Tools used

Software development processes are formalized and tracked using tools from Atlassian Suite (Jira, Bamboo, Bitbucket, Confluence), TestRail and SonarQube. For example:

1. Jira tool is used for monitoring bugs and tasks. Jira embraces the Agile software development methodologies and the team is using it when performing a sprint. This tool can generate metrics regarding the number of issues (bugs, tasks) and sprint performance (sprint burndowns, control charts and cumulative flow diagram)
2. Bamboo is used for continuous integration and release management. This tool can generate metrics for the build duration, number of failures, number of unit-tests runs
3. Bitbucket is our Git internal repository and hosts the code but it is the place where software developers are submitting their source-code and waiting for their pull requests to be approved. This tool can generate metrics regarding the number of pull-requests, number of approvers and time to approve.
4. Confluence is a document management tool and it is used as a knowledge base repository for the activities outcome performed during the projects.
5. Testrail is a test-case management tool and it is used for monitoring test runs. It can expose metrics related to test-cases coverage and failed test runs, most failed test-cases, etc.
6. SonarQube is a tool for monitoring software quality and it is extensively used in MEASURE Platform.

Run scenario

A number of three repositories were used with in this scenario (out of 12 possible repositories) using SonarQube:

- First repo is used for committing code of the tool for collecting events from Bitdefender and Windows OS technologies. It has 6.8k lines, 16 bugs, 1d of Technical Debt and 0 Vulnerabilities
- Second repo is used for anomaly detection code written using Python. It has 3K lines, 18 bugs, 3h of Technical Debt and 1 Vulnerability
- Third repo was used to integrate Security Analytics in GravityZone platform using NodeJS. It has 7.5K lines, 13 bugs, 6h of Technical Debt, 0 Vulnerabilities and 5.6% duplicated code

These three repositories were selected because of the variety of languages, size and activity. For example, second and third repositories are in maintenance mode, code is deployed in the market. The first repository is active and people are committing almost daily.

In SonarQube, there are added more than 50 repositories in Bitdefender and it is very hard for a team to have an overall status. If we go further to other development processes like continuous integration and release management, bug and task software development that are available in other platforms the task for a manager or a technical lead becomes more and more challenging.

In this scenario, Software Engineering manager created a dashboard using data from SonarQube, Bitdefender Security Analytics and RIVER and presented to Technical Leads in order to evaluate MEASURE Platform capabilities. The feedback was very good for metrics visualization and there were questions about metrics integration from Atlassian suite tools and TestRail. Another point was that presenting only a status/snapshot of current metrics is good but not enough. For example, the above metrics presented for evaluated repositories have low numbers on Technical debt and it is interesting to see the trends and correlate with the number of bugs from Jira, number of failed builds and failed test-cases in TestRail. Here is where MEASURE analysis tools like Quality Guard, Metrics Suggester and Stracker have a lot of potential to improve managerial activities.

5.4. Results Analysis

MEASURE platform is a tool with a lot of potential and it can be improved through community contributions and the challenges are that there are a lot of commercial and non-commercial tools that are collecting data on software engineering processes. A new layer of presenting and analysing data, like MEASURE Platform, will provide a lot of value to managers and their teams.

In terms of business perspective, integration of Bitdefender Security Analytics in Measure platform has both short-term and long-term benefits for the technology we work on and for the people that collaborated. Being integrated in a platform can create business opportunities if the platform is marketed and gains enough customer traction and Bitdefender is already present in Kaseya MSP solution (Managed Service Providers). Another facet is that being part of the Measure Project it also helped evaluators from management and software development to see how many opportunities of measuring there are for software development initiatives. Mint and Stracker Measure Platform plugins are great examples of how monitoring quality of work can be improved.

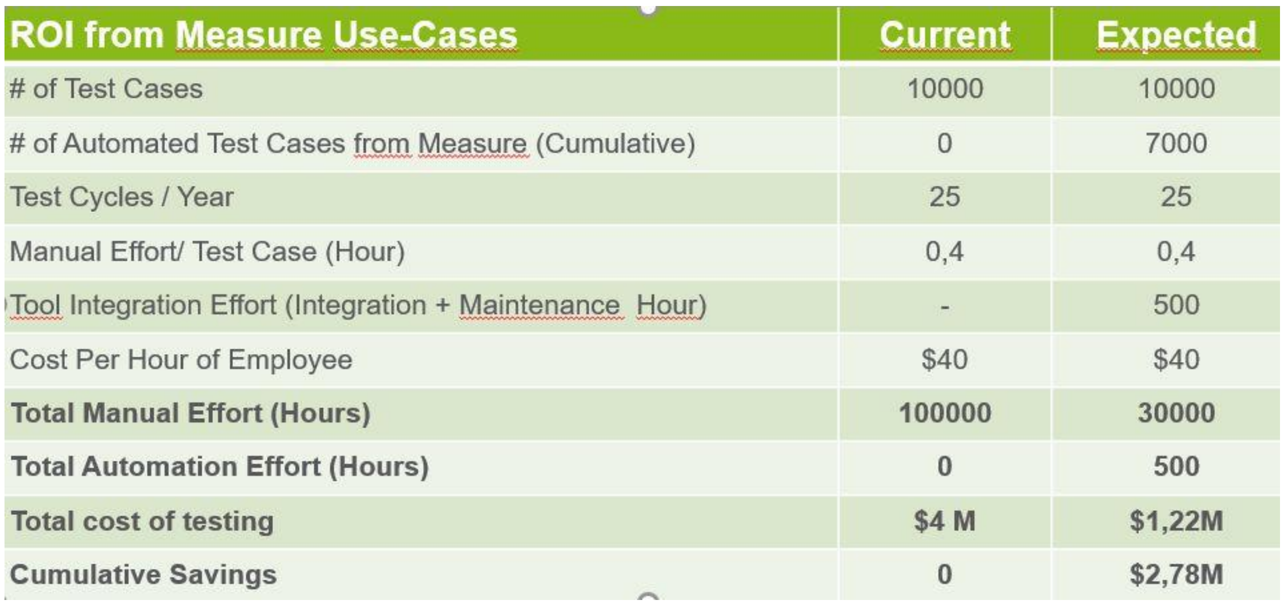
6. Use Case: Test generation case study by Turkish consortium

While the Turkish team joined the project in February 2018, the results of the case study are currently a work in progress. We however develop in this section the main evaluations of the MEASURE platform performed by the Turkish consortium.

6.1. Evaluation Approach

The purpose of this evaluation is to study the usefulness and practicality of the MEASURE platform monitoring tool and test generation use cases of the Turkish consortium. We want to provide insightful information about software quality and test usefulness in MEASURE monitoring platform. Since our use cases are still in progress, we use mocking services and outputs to figure out how test case generation use cases will work within the MEASURE Platform.

In our specific cases, we will assume that our case studies are completed and our test metrics/ KPI's are integrated within the MEASURE platform properly.



ROI from Measure Use-Cases	Current	Expected
# of Test Cases	10000	10000
# of Automated Test Cases from Measure (Cumulative)	0	7000
Test Cycles / Year	25	25
Manual Effort/ Test Case (Hour)	0,4	0,4
Tool Integration Effort (Integration + Maintenance Hour)	-	500
Cost Per Hour of Employee	$40	$40
Total Manual Effort (Hours)	**100000**	**30000**
Total Automation Effort (Hours)	**0**	**500**
Total cost of testing	**$4 M**	**$1,22M**
Cumulative Savings	**0**	**$2,78M**

Figure: ROI from MEASURE use cases

From this point of view, three evaluation scenarios have been specified:

- 1st scenario consists of analysing test case requirements and after creation of all test cases regarding those requirements and identify "Key Performance Indicators" related to the software quality in requirement phase.
- 2nd scenario consists of monitoring test case execution time and compare the results of duration with baseline duration and identify "Key Performance Indicators" related to the software quality in performance.
- 3rd scenario consists of analysing test case results and after finishing all test cases of application under test identify and "Key Performance Indicators" related software quality in functionality.

6.2 Evaluation scenarios

6.2.1 Analyse Test Case Requirement coverage using Measure Platform

In this scenario, we analyse test case requirements and test cases that derived by test case generator. Measure Platform will analyse the results from test case generator; those results are requirements number, derived test case number, requirements risk etc. and Measure Platform will identify coverage metrics for Application under test.

This will allow us to understand whether efficient test cases are derived from requirements or not. Depends on the analyses results

We will continue to test processes or create more test scenario for the requirements.

6.2.2 Monitoring Execution time duration of test cases using Measure Platform

In this scenario, we monitor test case execution duration and note the duration of all test scenario execution time totally and individually.

We will have already predefined baseline execution times of all scenarios, so automatically executed automation scenarios will let us to understand any latency or performance bottlenecks happened during last execution. Measure Platform will interpret those numbers and give feedback about software quality in performance. In the light of this information we will decide to take actions regarding software performance.

6.2.3 Analysing test case results using Measure Platform

In this scenario, test case generator automatically will execute all the test cases that derived from requirements with the help of automation framework build inside the test case generator. We will have a clear sight of which test cases are passed and which ones are failed.

This will allow us to identify key performance indicators related to whether software functionality is working, partially working or not working.

Measure Platform will analyse the test results and rate target application has enough maturity in software functionality aspect.

6.3 Overall evaluation of the MEASURE platform in the Turkish case study

Overall evaluation of the MEASURE platform in the Turkish case study

After running the first experiments, MEASURE platform seems valuable to Turkey use case. In particular, the Turkish use case is used for processes that has an ambition to develop innovative products for telecom companies.

Specifically, software development projects that require high experience on Requirements Model, Functional Size Measurement and Test Automation are the main beneficiaries of this use case.

MEASURE platform will help our use case to exploit accumulated Big Data in our software development engineering cycle.

MEASURE platform will help Turkish use case by improving the software lifecycle performance and in return shortened the time to market duration. Besides, it improved the quality of the outcomes which also decreased the unnecessary costs.

MEASURE project brought a standardization approach at existing requirement model and test model.

Moreover, defining test metric suite on Measure Platform, created a possibility of knowledge discovery from big data that is accumulated in the process of software engineering activities by using machine learning algorithms.

7. Conclusions

This document presented the final evaluation of the MEASURE platform and its tools through industrial case studies and diverse scenarios. Metrics, goals and questions defined in D5.3 have been used to tackle the feasibility and usability of our platform in industrial contexts. Three main use cases well-defined by SOFTEAM, NAVAL Group and BITDEFENDER have been analysed and a general valuability have been performed by the whole Turkish consortium (that lately joined the MEASURE project in Feb. 2018). We illustrate in the following Table the general satisfactions/success of the processes of the analysis tools on the MEASURE use cases.

INDUSTRIAL USE CASES	ANALYSIS TOOLS	SATISFACTION / SUCCESS
Modelio Product Line **SOFTEAM**	Quality Guard	✓ / ✓
	Metric Suggester	✓
	STracker	✓ / ✓̸
Large Naval Software System **NAVAL Group**	Quality Guard	✓ / ✓̸
Security Analysis **BITDEFENDER**	MINT	✓ / ✓̸
	STracker	✓ / ✓
Test Case Generation **Turkish Consortium**	MEASURE Platform assessment	✓ / ✓

Table 2 - General satisfaction/success of the analysis tools performed on the industrial use cases (✓: fully satisfied / ✓̸: difficulties encountered)

Since there are today a lot of commercial and non-commercial tools that are collecting data on software engineering processes, the MEASURE tools for data analysis have been plebiscited as a major contribution. A new layer of presenting and analysing data, like MEASURE Platform, will provide a lot of value to managers and their teams.

It has been concluded that being integrated in the MEASURE platform provides relevant business perspectives and opportunities, in particular if the platform is marketed and gains enough customer traction. Furthermore, business advantages have been raised illustrating the possible transfer of our technologies in other domain. It has been demonstrated that the deployment of the MEASURE platform allows to detect potential faults prior in the phases during a project (e.g., in the development phase). An estimation of the gain has been calculated by SOFTEAM concluding that the deployment and exploitation of the Measure Platform on the Modelio 3.8 development process saved nearly 25 600 € (within the period May-Oct 2018). A same study on the use of the analysis tools (in particular in processing and exploiting our prediction tool) led approximately on the same financial gains (within the period Nov 2018-Jan 2019).

Furthermore, it is interesting and relevant to note that some of the MEASURE analysis tools have been executed on different scenarios of different use cases, such as Quality Guard. It provided a broad view of the usability of the platform and its tools within diverse contexts. Basically, three indicators have been extracted to express the facility in terms of Deployment, Integration and Exploitation of the platform in the current performed industrial use cases. The two first have been highly achieved (into the two first use cases) while the third one revealed issues in terms of lack of logs to debug in some project phases and the reactivity to raise alerts or misfunctioning. However, due to the effectiveness of the MEASURE teams in charge of the platform and the analysis tools, some of these lacks were fixed while some others are in progress.

Based on the Turkish evaluation while partners started assessing its usefulness, the consortium concluded that the MEASURE platform seems valuable to their use case and is very promising. Through an internal questionnaire and its results, it has been noted that software development projects that require high experience on Requirements Model, Functional Size Measurement and Test Automation are high beneficiaries with regards to exploiting accumulated data within the engineering cycle (and analysis through learning techniques), improving the software processes performance and then shortening the time to market duration with improved outcomes quality and decreased costs.

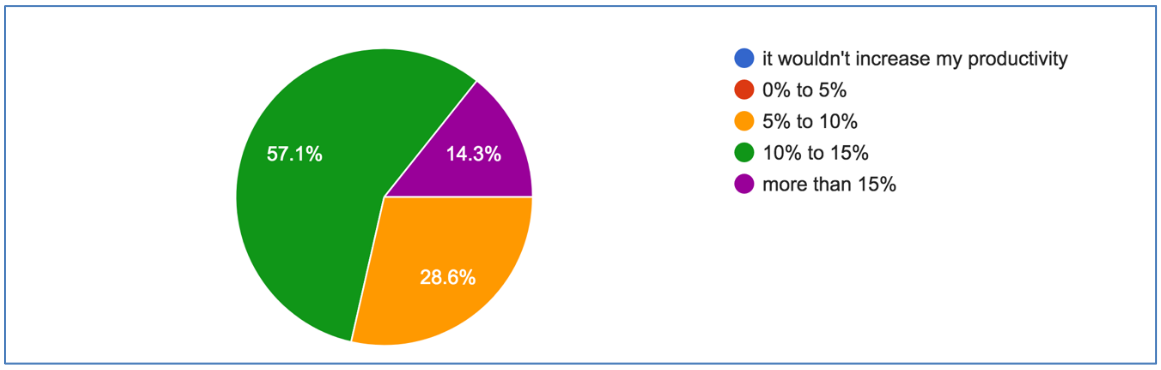
Finally, based on an internal consortium-based evaluation, we also may provide the following qualitative analysis of the impact and usefulness of the MEASURE Platform deployment and integration. They describe the general requirements of our platform based on the deliverable D5.1 and the level of fulfilment. Some requirements were evaluated by experimented users and responses by an internal questionnaire. We depict in the following tables and figures, two types of requirements: the subjective usefulness and the business advantages of our MEASURE platform.

General Subjective Requirements about the MEASURE platform	Fulfilment
Its usefulness in a common job.	Good 57% – Very Good 43%
Its contribution to provide valuable information to a job.	Good 43% – Very Good 57%
The platform provides the functionalities the manager/engineer/user need.	Good 85,7%
Its results can be integrated in your development process.	Good 85%
It can enhance the quality of your work.	Good 60% - Very Good 40%

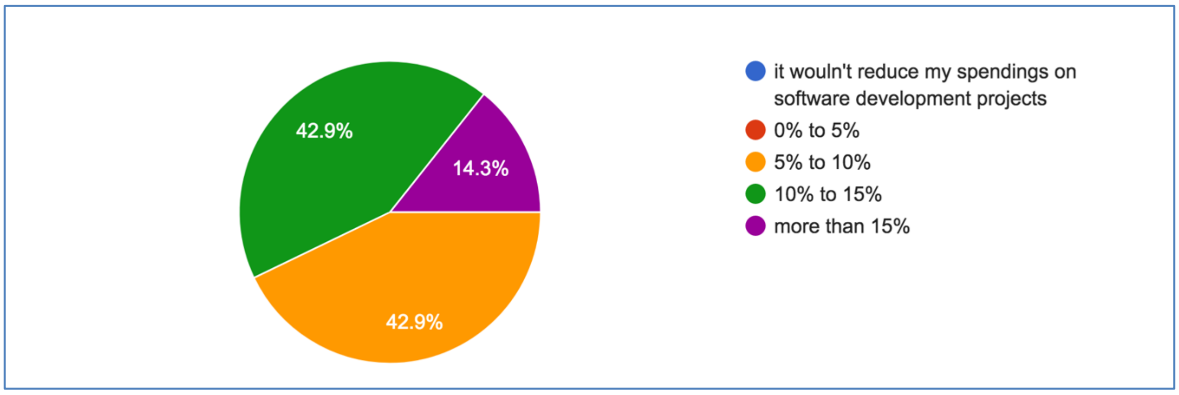
Table 3 - General Requirements Fulfilments about the MEASURE platform.

Through these industrial use cases, we also evaluated a very important aspect which was the perceived business advantages of using the MEASURE platform. We asked the evaluators to estimate how much would MEASURE platform help them, by providing a percentage of improvement. Three dimensions were successfully assessed:

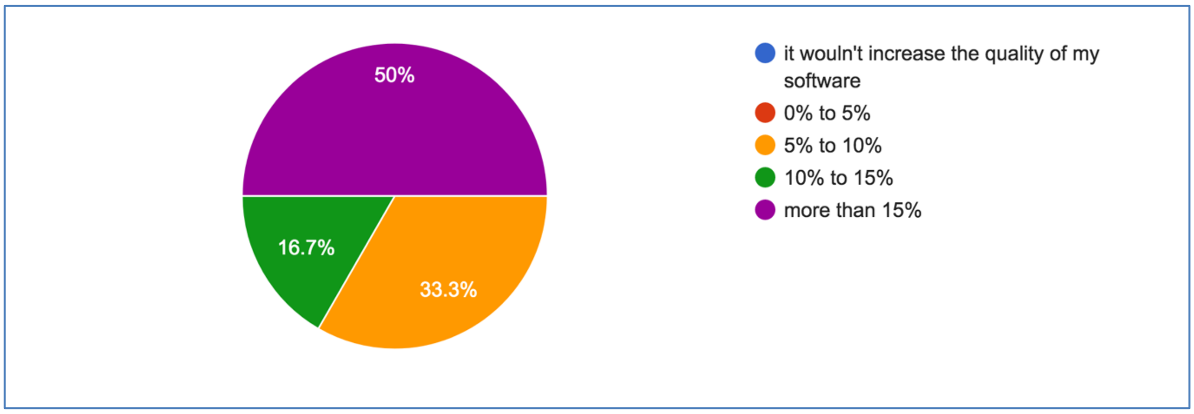
Software development improvement: By what percentage do you think that MEASURE platform could increase your software development productivity?



Software project costs reductions: By what percentage do you think that MEASURE platform could reduce the spending on your software development projects? Please answer based on your specific scenario



Software quality increase: By what percentage do you think that MEASURE platform could increase the quality of your software? Please answer based on your specific scenario


● it wouln't increase the quality of my software
● 0% to 5%
● 5% to 10%
● 10% to 15%
● more than 15%