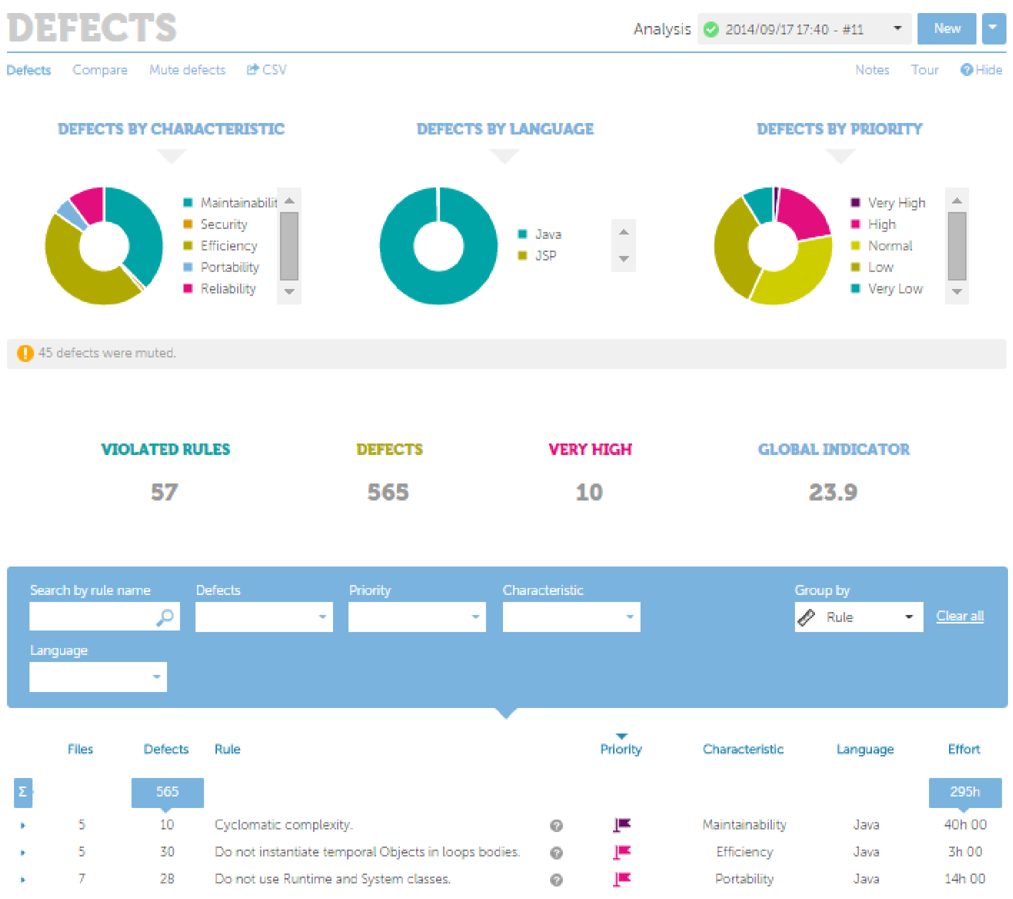
the full features list.

kiuwan
DIVE IN YOUR CODE

Index of features

Measure & analyze your software



45 defects were muted.

VIOLATED RULES	DEFECTS	VERY HIGH	GLOBAL INDICATOR
57	565	10	23.9

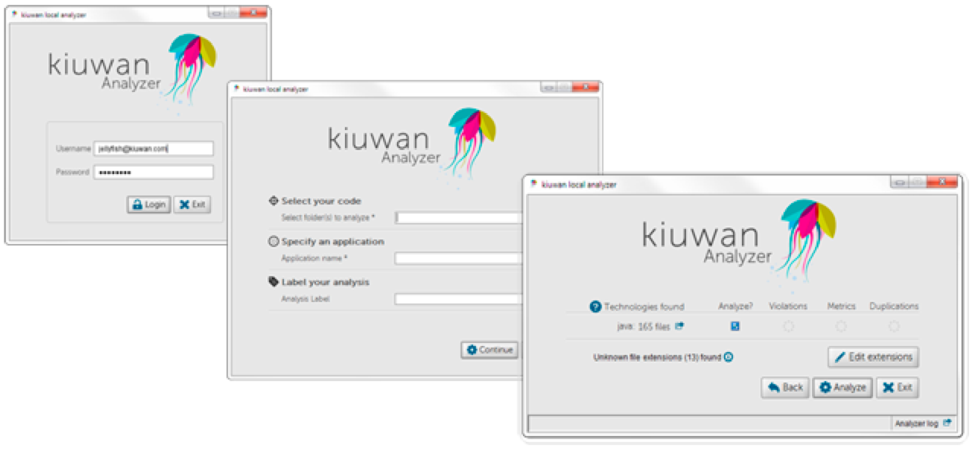
	Files	Defects	Rule		Priority	Characteristic	Language	Effort
Σ		565						295h
▸	5	10	Cyclomatic complexity.	?	⚑	Maintainability	Java	40h 00
▸	5	30	Do not instantiate temporal Objects in loops bodies.	?	⚑	Efficiency	Java	3h 00
▸	7	28	Do not use Runtime and System classes.	?	⚑	Portability	Java	14h 00

Measure, analyze and verify code quality of your application portfolio, that is the goal when using Kiuwan. It immediately provides the essential results and reports on the analyzed code including the following:

- Defect detection. Based on the compliance to quality rules and best programming and design practices, defined by the industry.
- Software quality metrics.
- Duplicated code detection.
- Kiuwan supports different technologies such as: Objective-C, Java, JSP, Javascript, PHP, C/C++, ABAP IV, Cobol, JCL, C#, PL/SQL, Transact-SQL, SQL, VB 6, VB.Net, Android and Hibernate and you will benefit from the above and everything you require for the assurance, control and certification of your developed software, essential to:
- Manage development and maintenance cost. Either in-house or outsourced.
- Align business and software development that supports it.
- Increase development project productivity.

Most organizations don't have or can't afford the resources necessary to automate quality certification. Kiuwan is the answer for them.

Analyze your software locally



Username: jellyfish@kiuwan.com

java: 165 files

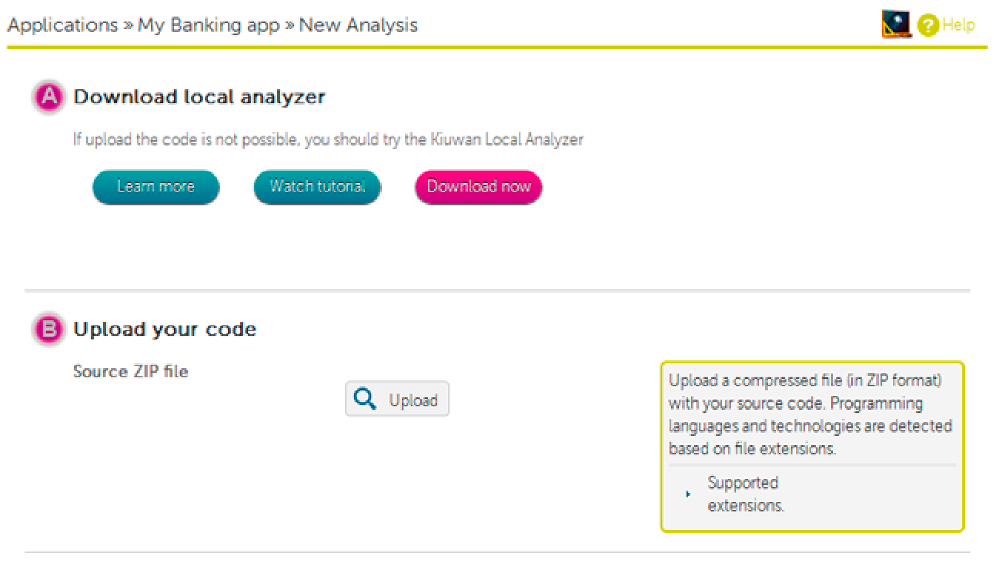
If you don't want to upload your code, that's fine. You have the option to download the Kiuwan analyzers and run the source code analysis on your local computer.

Just go to the new analysis screen and you will have the option to upload your code or download the analyzer right on the spot.

Downloading the analyzers has other benefits, like integrating the measurement and analysis of your applications in your continuous deployment process. Or make it part of your nightly builds.

All resulting data is securely uploaded to the Kiuwan platform where you can see the results in the comprehensive dashboards, generate reports, run what if analysis, see the metrics and the defects lists. Just the same as if you had uploaded and run the analysis on the platform.

Analyze your software securely in the cloud


Applications » My Banking app » New Analysis

A Download local analyzer

If upload the code is not possible, you should try the Kiuwan Local Analyzer

B Upload your code

Source ZIP file

Upload a compressed file (in ZIP format) with your source code. Programming languages and technologies are detected based on file extensions.

Supported extensions.

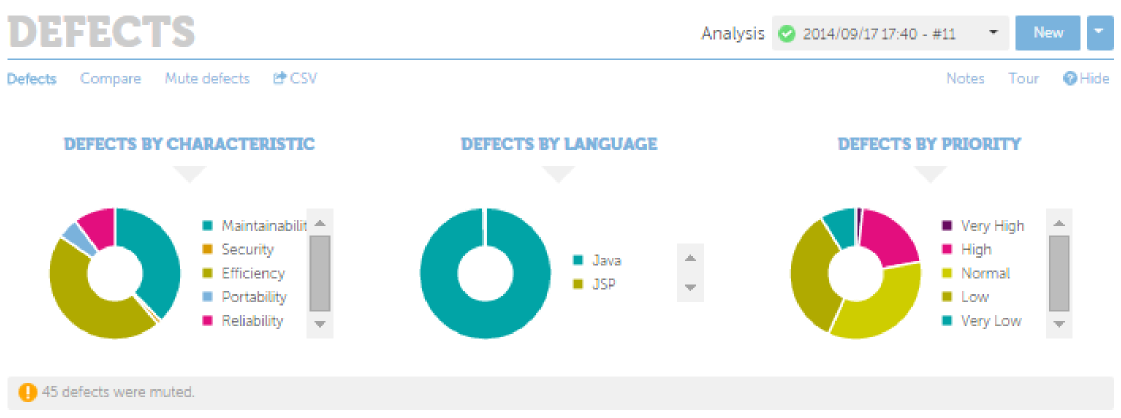
Simple and secure. When you want to run a code analysis with Kiuwan you always have two options:

A. Run your analysis locally on your own machine with our downloadable analyzer. Just click the download button in the New Analysis page to install the analyzer on your machine, we have a nice tutorial to help you on this, but don't worry it's a piece of cake! Once the analyzer is installed you don't even have to go to the New Analysis page any more, just open the analyzer GUI and off you go.

B. Stuff all your application's source code in a zip file, and upload it using the Upload button in the New Analysis page. Don't worry about the directory structure, we'll find the source code files and tell you which ones will be analyzed.
Label your analysis, decide what quality model to use, tell us the encoding used in your files and that's it!

Just one more thing, don't worry about security, your code is always uploaded over a secure connection and you can rest assure we never keep any of the code we analyze, it is immediately removed from our servers once the analysis is complete.

Multi language/technology application support



These are the technologies currently supported. We are continuously working in supporting more, come back often to check them out!

Languages	Recognized extensions
Objective-C	**.m**
JCL	**.jcl, .prc**
Transact-SQL	**.tsql**
JavaScript	**.js**
PHP	**.php .php3 .php4 .php5 .php6 .phps. phtml**
C/C++	**.c .h .hh .cpp .hpp .cc .pc**
Java	**.java**
JSP	**.jsp .xhtml .jspx**
Cobol	**.cbl .cpy .cob**
Abap IV	**.abap**
C#	**.cs**

Oracle PL/SQL	**.st .sp .sps .trg .plsql .spp .sf .pkb .pks .fnc .spb**
SQL-92 Standard	**.xml .sql**
VB6	**.cls .bas .frm**
VB.net	**.vb**
Powerbuilder	**.sru .sra .srw .srf .srs,srm .srx**

And more!
(Contact us to learn about other supported languages)

Frameworks	Language
Hibernate	**Java**
Android	**Java**

Nowadays it is difficult to define what an application is. Organizations have lots of code, where are the borders to know if some piece of code belongs to this or that application? It is difficult to say. Only you know that.

Let's get a little academic and try to define application: "It is a set of software components organized in a predefined architecture that work together to provide the necessary functionality to accomplish one or more business goals".

Fair enough, but let's be more specific. What are these software components and what is a predefined architecture?

A software component is for example, a java, .net or C++ class, a COBOL copy or a COBOL program that implements CICS transaction, a configuration file, an HTML page, a JavaScript, a database table, a shell script, a JCL, a web service or other kind of software service or even a message sent to a middleware messaging bus. Anything running on a computer that cannot provide business value by itself. Okay, put a bunch of components together following certain relationships and dependencies -that is your predefined architecture-, and you have an application implementing a set of functionalities with a clear business goal.

You can see that an application can have components written in different languages, and it all depends where you draw the lines.

The point is that, with Kiuwan, no matter where you put the border of an application. We'll treat all the code equal. You don't have to worry about different languages in the same application (i.e. java & JavaScript & html). Kiuwan will analyze them all, give you relevant info for each language and provide language/technology independent indicators for what

Jenkins
continuum
Atlassian Bamboo
cruisecontrol.
Hudson

you consider your application.

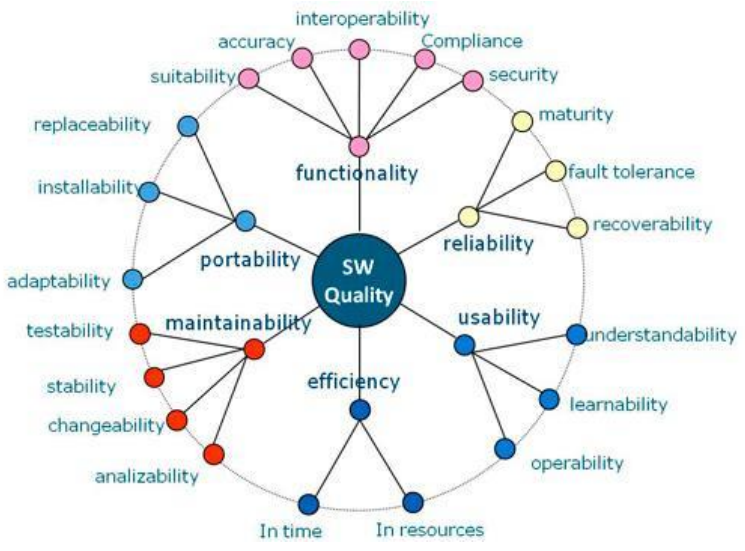
Integrate with your continuous deployment process

Once you have the capability of measuring and analyzing your code locally, why not integrate the quality control with your continuous integration process?

We make it simple for you, follow the instruction in the documentation to make your orchestrator invoke the Kiuwan analyzers.

The results are automatically uploaded to your kiuwan account, so you are one click away to see the evolution of your applications quality for every deployment you do.

Ready to use quality model



If you don't have a quality model, you can't measure and analyze your software. Period. That's why we have developed the Kiuwan quality model. We call it CQM and is based on standard recommendations to evaluate software products like the ISO/IEC 9126 "Software Product Quality Requirements" and the ISO/IEC 14598 "Software Product Evaluation".

Why CQM? Well, we have a broad experience introducing quality plans to all kinds of organizations. We noticed that organizations have common issues and interests. There was a need to define a quality model and an associated methodology, supported by tools, products and solutions. Therefore CQM, its adoption will provide the target benefits and will resolve the most common issues.

Using CQM saves time and money in the quality assurance process.

You will get short term benefits by using CQM:

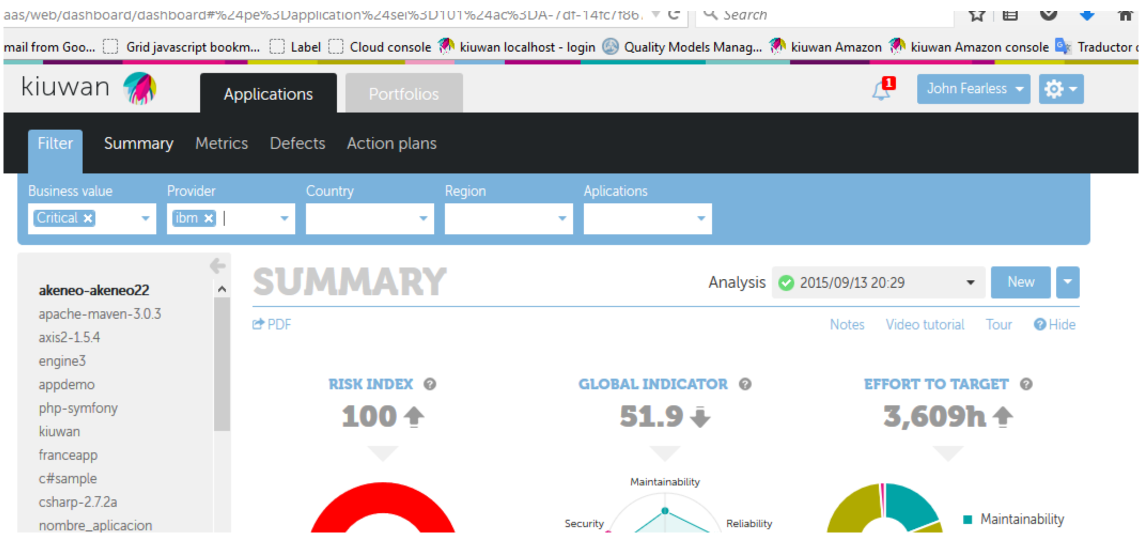
- Abstract from technical layer. Your quality information will be independent from programming languages and platforms.
- Compare different versions of the same software. This answers the most important question: has my software improved?
- Compare the quality of different applications. It does not matter if they are different kinds of applications or that they are developed in different technologies.
- Evaluate the technical requisites in order to accept the software from a third party provider.

But also, other benefits:

- Aggregations of data. You can aggregate the quality information from different applications in order to get an evaluation of the software produced by a provider, a country or IT area compared to others.

- Continuous improvement process. You can apply a quality control methodology in your software lifecycle process.

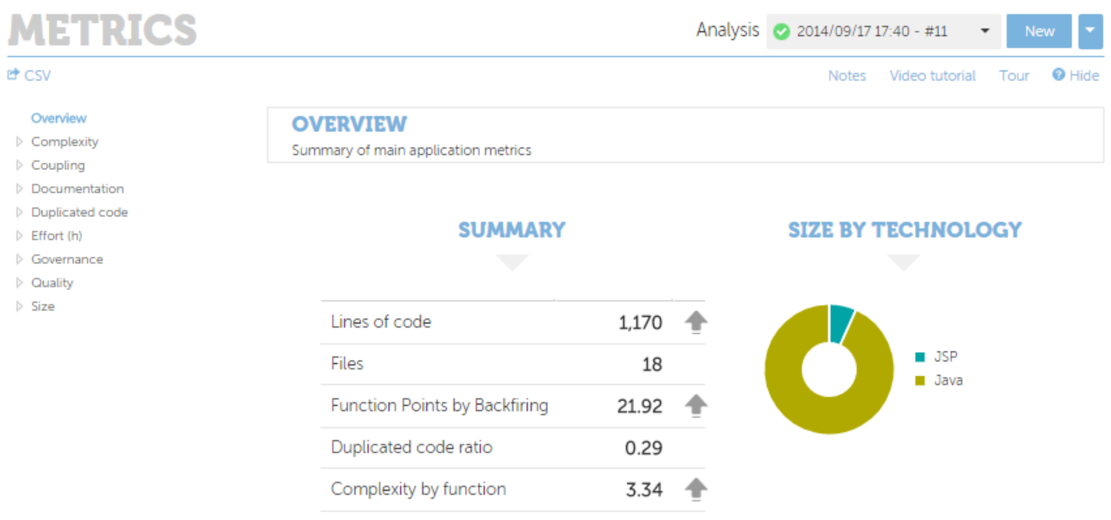
Filters: a powerful way to see your results



Kiuwan results like you've never seen them before: narrow the applications you want to view based on your defined portfolios, decide how you want to aggregate your Kiuwan results by any combination of portfolios to sharpen your conclusions by using filters.

Filters will apply throughout all Kiuwan screens. You can modify them anytime, anywhere in the application, and your selection will be applied on the fly.

Code metrics



Lines of code	1,170
Files	18
Function Points by Backfiring	21.92
Duplicated code ratio	0.29
Complexity by function	3.34

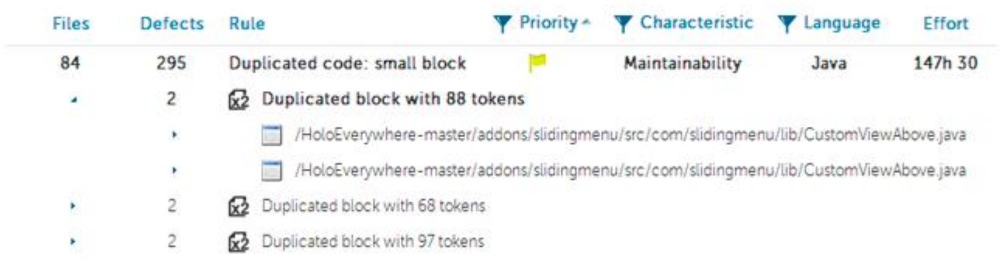
Kiuwan measures several intrinsic code metrics in each analysis. We organize them in different categories depending on their nature:

- Size. That includes physical size and functional size metrics.
- Complexity related metrics. Including CCN and fan-out.
- Documentation. Measuring how good, or bad, is your code documented.
- Quality. Compliance to rules and best practices defined in the quality model metrics. Number of defects, rules violations.
- Governance. In terms of the exposure to risk of your development efforts.

You can access all the information you can imagine about your code metrics in the application metrics dashboard, including the evolution of the most relevant metrics so you can assess the improvement in your development teams.

Get also a Metrics Report (in a convenient XLS format), containing all the calculated metrics, both globally and at file level.

Code duplication analysis



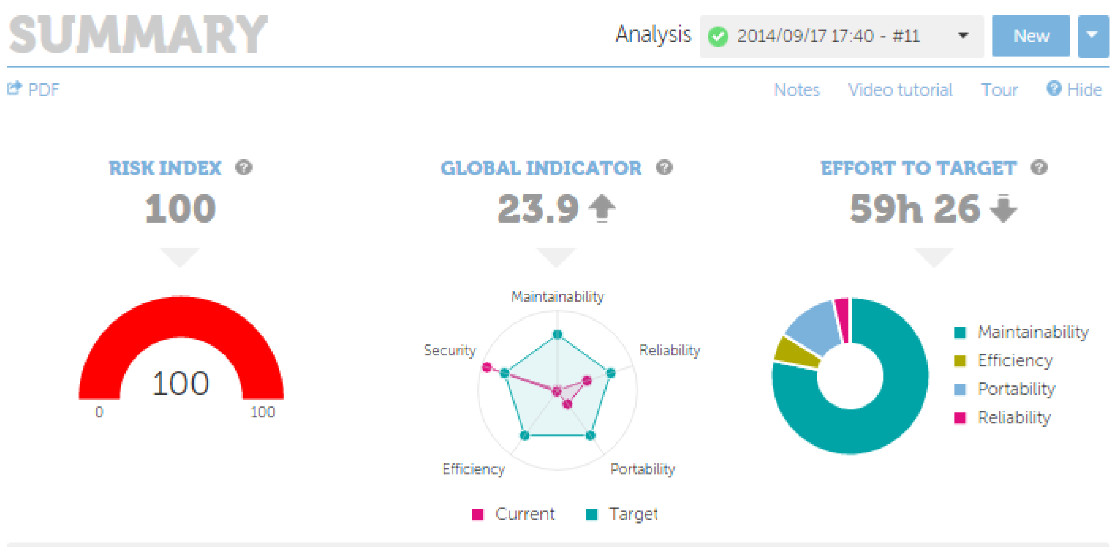
Files	Defects	Rule	Priority	Characteristic	Language	Effort
84	295	Duplicated code: small block	🟨	Maintainability	Java	147h 30
	2	Duplicated block with 88 tokens				
		/HoloEverywhere-master/addons/slidingmenu/src/com/slidingmenu/lib/CustomViewAbove.java				
		/HoloEverywhere-master/addons/slidingmenu/src/com/slidingmenu/lib/CustomViewAbove.java				
	2	Duplicated block with 68 tokens				
	2	Duplicated block with 97 tokens				

Code duplication is one of the most frequent development bad practices. Ah! The temptation of cutting & pasting code, show me a developer who doesn't do it. I have to admit that it is a useful practice if you don't want to start a program from scratch, but it can get dangerous if you abuse its use.

Code duplication is a nightmare for code maintenance. Large software systems typically contain 10-25% of such redundant code, just imagine you have a bug in a piece of code you have duplicated in several modules. The right thing to do is to refactor the code to properly reuse functionality and avoid cut & paste.

Kiuwan helps you to track the rate of the duplications found in your code and even tell you where to look in your code to fix the problem. Kiuwan duplication analysis is based on tokens, so we can consider a piece of code duplicated even when it is not exactly the same. This is your typical situation when code is cut & pasted.

Kiuwan Software indicators



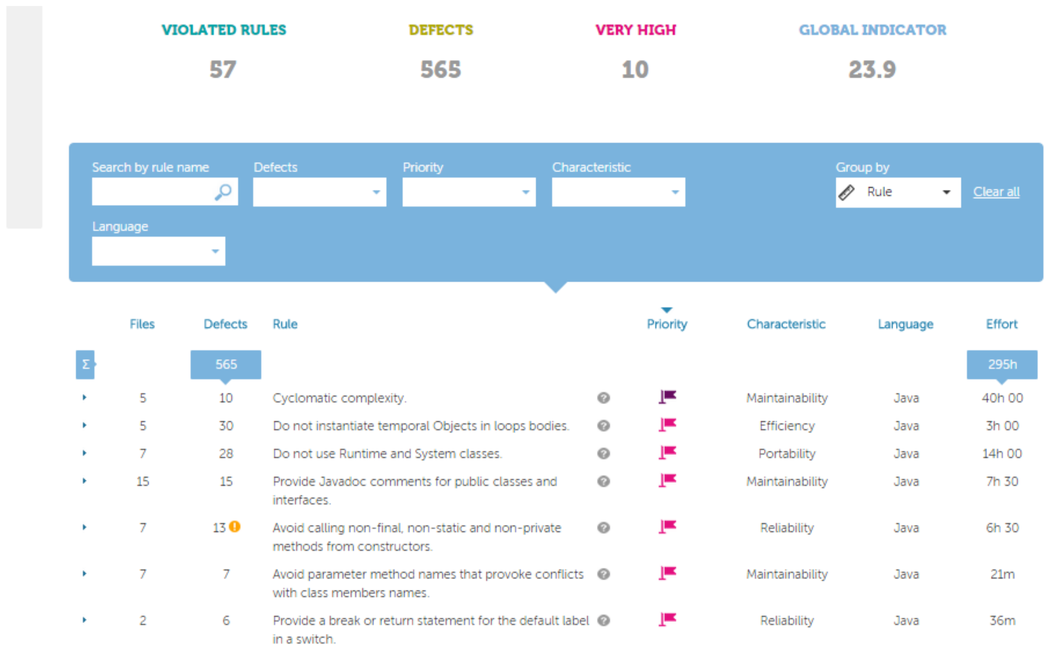
We calculate the Kiuwan indicators based on evidence (defects and metrics) taken from the source code analysis. Now, you can see the trend of all quality indicators with colored arrows. Based on the values of the previous analysis, we give you an immediate representation of the effectiveness of your efforts to improve the quality of your software. Right. What does Kiuwan consider a defect?

A defect is a violation to a rule defined in the quality model for a specific language. Each rule in the model is categorized in one of the 5 software characteristics defined in CQM. No violations to the rules in a category means no defects. Hence a 100 value to the corresponding software characteristic indicator, the best score you can get. When we find a violation to a rule in a category, the corresponding software characteristic indicator will decrease. How much? It will depend on the number of computed violations to that rule in all the source code and the importance that rule has in CQM. The bigger the number of violations and the bigger the importance of the rule, the bigger the contribution to decrease the software characteristic indicator.

Kiuwan provides indicators for:

- Software characteristics defined in CQM: Efficiency, maintainability, reliability, security and portability
- Global quality indicators. Calculated as weighted average of the above software characteristics. The weight for each characteristic can be customized in CQM. There are organizations that want to give maintainability more weight than reliability, for example
- Risk index. Associated to the structural quality of the software.
- Effort to target. The amount of work needed to reach the quality goal.

Defects list



VIOLATED RULES	DEFECTS	VERY HIGH	GLOBAL INDICATOR
57	565	10	23.9

Search by rule name · Defects · Priority · Characteristic · Group by: Rule · Clear all

Language

	Files	Defects	Rule	Priority	Characteristic	Language	Effort
Σ		565					295h
▸	5	10	Cyclomatic complexity.	⚑	Maintainability	Java	40h 00
▸	5	30	Do not instantiate temporal Objects in loops bodies.	⚑	Efficiency	Java	3h 00
▸	7	28	Do not use Runtime and System classes.	⚑	Portability	Java	14h 00
▸	15	15	Provide Javadoc comments for public classes and interfaces.	⚑	Maintainability	Java	7h 30
▸	7	13 !	Avoid calling non-final, non-static and non-private methods from constructors.	⚑	Reliability	Java	6h 30
▸	7	7	Avoid parameter method names that provoke conflicts with class members names.	⚑	Maintainability	Java	21m
▸	2	6	Provide a break or return statement for the default label in a switch.	⚑	Reliability	Java	36m

Kiuwan indicators are based on evidence. Part of that evidence —along with intrinsic code metrics— are the defects found in the source code analysis.

Right. What does Kiuwan consider a defect? A defect is a violation to a rule defined in the quality model for a specific language and a software characteristic.

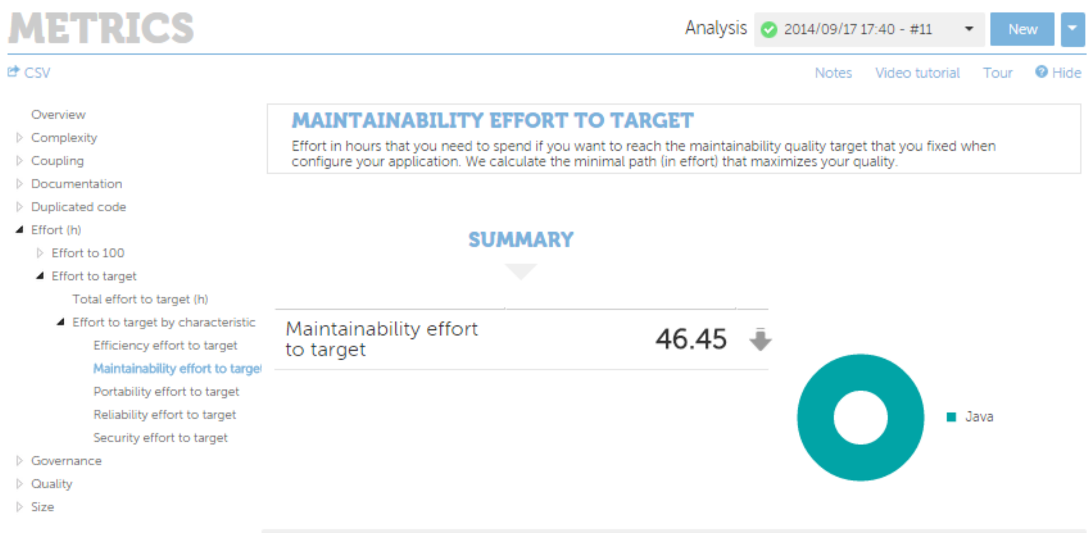
Kiuwan provides a full list of all detected defects found in the source code.
The first thing you see is a summary with total number of violated rules, the total number of defects found and the number of critical defects.

Each row in the list will give you the defect related to one violated rule. You have the number of files affected, the number of defects for the rule, the rule itself, the affected software characteristic, the programming language of the code and the total estimated effort to fix all those defects. You can sort the list by any of the above mentioned columns.

This list is the bread and butter for software developers. They will find all the information they need to fix the code. Each row expands to show the specific files where the defects were found. For each file, you can expand it to see the specific line of the defect and you can even see an excerpt of the code with the defect.

Of course, you can export the list with all detailed information to an Excel sheet.

Technical debt or effort to target


METRICS
Analysis
2014/09/17 17:40 - #11
New
CSV
Notes
Video tutorial
Tour
Hide
Overview
Complexity
Coupling
Documentation
Duplicated code
Effort (h)
Effort to 100
Effort to target
Total effort to target (h)
Effort to target by characteristic
Efficiency effort to target
Maintainability effort to target
Portability effort to target
Reliability effort to target
Security effort to target
Governance
Quality
Size
SUMMARY
Java

MAINTAINABILITY EFFORT TO TARGET

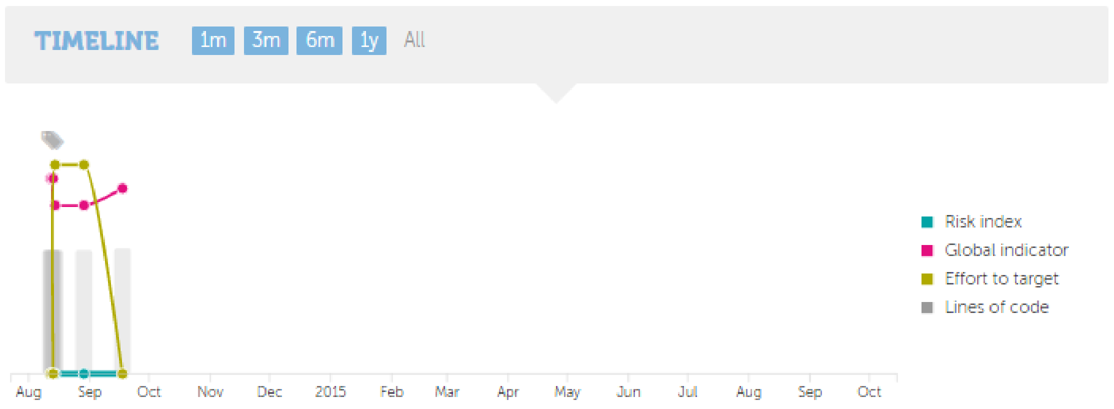
Effort in hours that you need to spend if you want to reach the maintainability quality target that you fixed when configure your application. We calculate the minimal path (in effort) that maximizes your quality.

Maintainability effort to target 46.45

The term "technical debt" was coined by Ward Cunningham, to describe the obligation that a software organization incurs when it chooses a design or construction approach that is expedient in the short term but that increases complexity and is more costly in the long term.

To explode the financial metaphor, technical debt is often expressed in dollars, euros, rupees... that's right good ol' money. Kiuwan provides functionality to discover debt incurred both intentionally and unintentionally. Kiuwan technical debt algorithms based on factual evidence: the code defects found in the source code analysis. Instead of giving you the debt in terms of money, we use the **effort to target** and the **effort to 100 quality indicator** as proxies for the technical debt. Translate it into money is up to you, multiply by your hourly development cost to get a number.

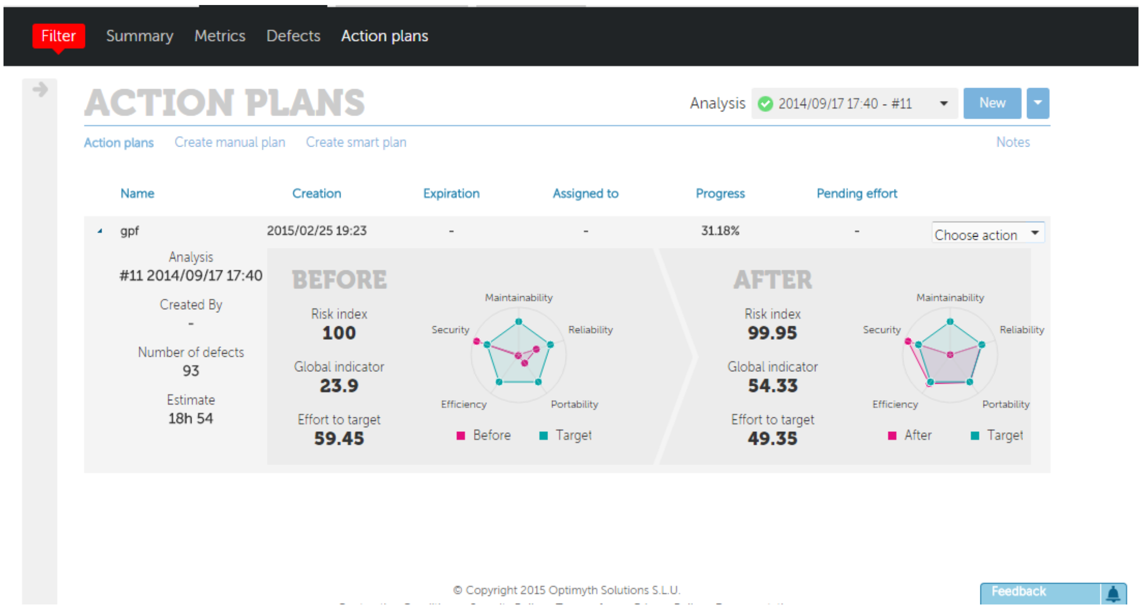
Historical evolution



When it comes to software quality, you want to track the evolution of all software quality indicators over time.

Kiuwan does not throw away any data. In fact, every time you analyze your applications with Kiuwan, we store all the data and show it in the overview dashboard in a time graph. You can see at a glance the tendency of your application's quality, but not only for a single application: Kiuwan provides the same functionality for application portfolios and for the global view that summarizes the quality of all your applications with one single set of indicators.

Action plans



Filter Summary Metrics Defects Action plans

ACTION PLANS

Analysis 2014/09/17 17:40 - #11 New

Action plans Create manual plan Create smart plan Notes

Name	Creation	Expiration	Assigned to	Progress	Pending effort	
gpf	2015/02/25 19:23	-	-	31.18%	-	Choose action

Analysis #11 2014/09/17 17:40

Created By -

Number of defects 93

Estimate 18h 54

BEFORE: Risk index 100; Global indicator 23.9; Effort to target 59.45

AFTER: Risk index 99.95; Global indicator 54.33; Effort to target 49.35

© Copyright 2015 Optimyth Solutions S.L.U.

Feedback

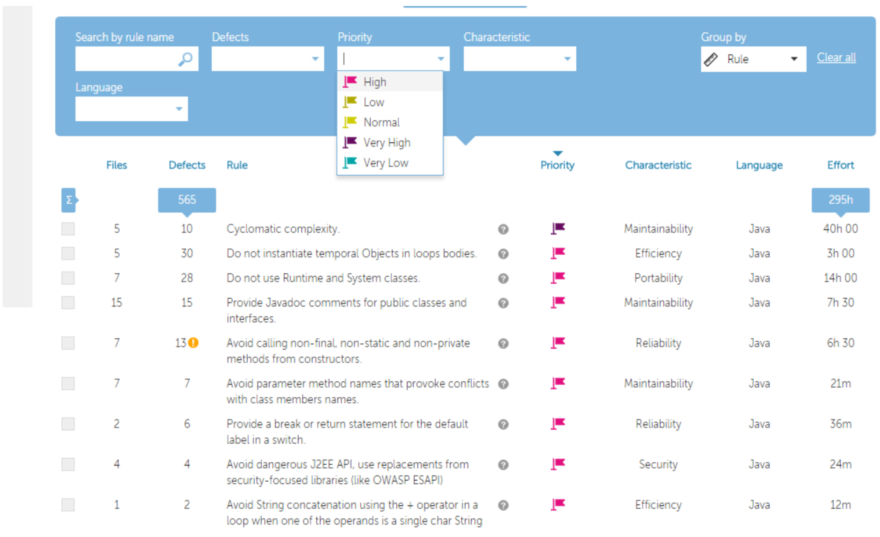
Action Plans have rightfully claimed their place as a managable entity in the **Applications** view.

All your action plans listed in a newly designed page. Click on them to expand a summary view with your application's indicators **before** implementing the action plan and **after** your development finishes with it.

Create new action plans with our good ol' "what if" simulations. With the **selector** on the right you can see its details, edit them and get your reports.

Create manual plan

You can create a manual action plan and pick which defects to work on.



Search by rule name · Defects · Priority · Characteristic · Group by: Rule · Clear all · Language

Priority options: High, Low, Normal, Very High, Very Low

Files	Defects	Rule	Priority	Characteristic	Language	Effort
Σ	565					295h
5	10	Cyclomatic complexity.		Maintainability	Java	40h 00
5	30	Do not instantiate temporal Objects in loops bodies.		Efficiency	Java	3h 00
7	28	Do not use Runtime and System classes.		Portability	Java	14h 00
15	15	Provide Javadoc comments for public classes and interfaces.		Maintainability	Java	7h 30
7	13	Avoid calling non-final, non-static and non-private methods from constructors.		Reliability	Java	6h 30
7	7	Avoid parameter method names that provoke conflicts with class members names.		Maintainability	Java	21m
2	6	Provide a break or return statement for the default label in a switch.		Reliability	Java	36m
4	4	Avoid dangerous J2EE API, use replacements from security-focused libraries (like OWASP ESAPI)		Security	Java	24m
1	2	Avoid String concatenation using the + operator in a loop when one of the operands is a single char String		Efficiency	Java	12m

What if analysis

This is probably one of the most interesting features of Kiuwan.

Wouldn't it be nice to know the effort it will take to reach the quality level you want? Or the other way around, what would be the quality of your application if you decide to *invest* a given effort to improve it? Go ahead, try Kiuwan's What If analysis!

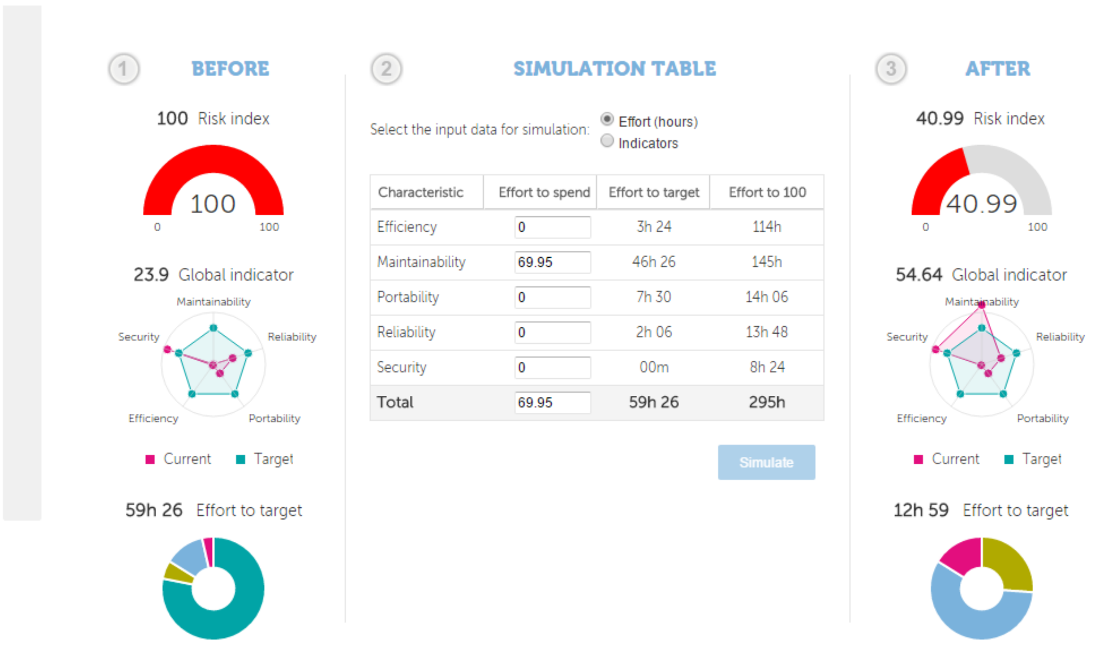
Just decide what you want to simulate:

- Effort. You can estimate the resulting quality indicator and risk index, if you invest the specified number of hours in one or more software quality characteristics.
- Quality. You can estimate the effort you need to invest to reach the quality indicators you specify for each software quality characteristics.

In one single view, Kiuwan will show you the current situation, the simulation parameters and the estimated results.

That's not all, folks! Kiuwan generates an automatic action plan for you to implement the simulation.

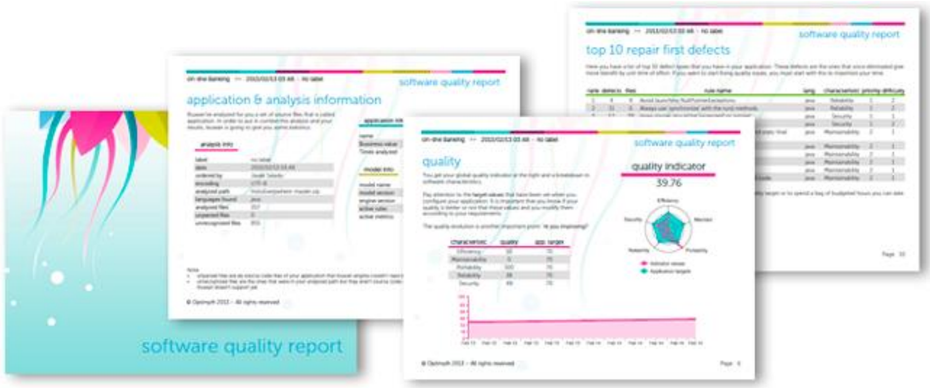
How cool is that?



Select the input data for simulation: [✓ Effort (hours)] Indicators

Characteristic	Effort to spend	Effort to target	Effort to 100
Efficiency	0	3h 24	114h
Maintainability	69.95	46h 26	145h
Portability	0	7h 30	14h 06
Reliability	0	2h 06	13h 48
Security	0	00m	8h 24
Total	69.95	59h 26	295h

Reporting



Developers, QA managers, development managers, QA engineers, even the CIO will need information regarding the health of your organization development efforts at different levels. That's right: you have all information you need for everybody in the Kiuwan itself, but sometimes is nice to have reports you can distribute and show around.

We have a report suited for any roles in your organization:

- Application overview report. A high level application's health overview in PDF format. It includes an introduction to Kiuwan's measurement methodology, a summary with the application analysis information, the Kiuwan indicators —risk index, quality indicator and reparation effort—, main code metrics values and top 10 *repair first* defects.
- Defects list. Developers need all the details of the defects found. This Excel format report lists all the violations to the rules with the file they were found, the violated rule and even the affected line in the code.
- Metrics values lists. More details for the developers. This Excel format report shows all the metrics values for each analyzed file.
- Action plan. This is the final —and most useful— result of the 'what if' analysis feature. With this PDF format report you will get a summary of the What If analysis with the final quality values and efforts needed to execute the plan, and the detailed list of the defects to repair to reach the goal.
- Organization overview report. This report includes the same information as the application overview report, but referred to the whole organization based in the latest analysis of all the applications. The Kiuwan indicators here are an aggregation of the individual application's indicators. Yes, it is in PDF format, as well.
- Portfolios overview report. Right, you guessed it: this is a PDF format summary of all the portfolios defined in your account with details of the applications that each portfolio includes and the criteria used to create them. Of course, for each portfolio you will have the Kiuwan indicators and main metrics values.

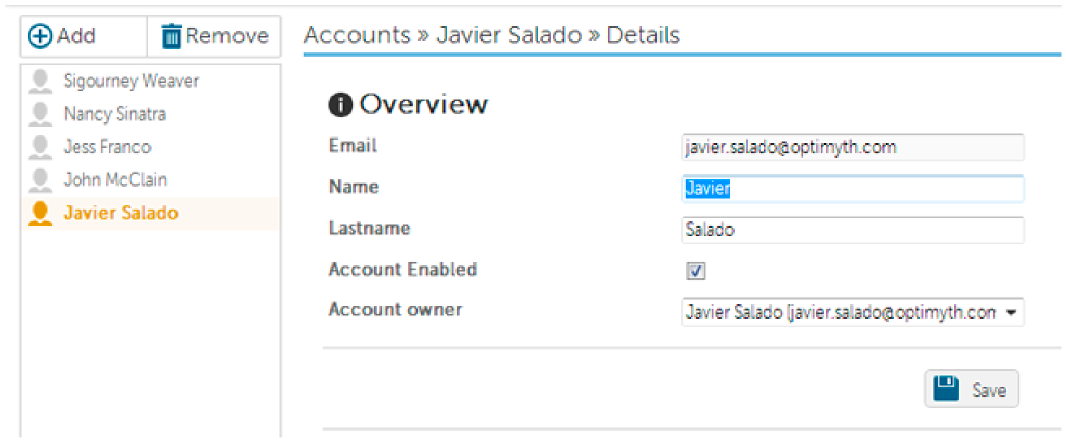
- **Quality Model reports**: for our paying customers, with annual or pay-as-you-go subscriptions, the quality model manager brings two brand new reports:

  1) A PDF executive overview of a specific quality model, with all the high level detail you need to understand its scope.

  2) An Excel spreadsheet with the detail all the rules and metrics documentation of a specific quality model. All the rules in one place fully documented. You can use it to play and decide your next quality model will look like.

We think we are not missing anything to report.

Multi user environment



Sigourney Weaver
Nancy Sinatra
Jess Franco
John McClain
Javier Salado

Accounts » Javier Salado » Details

Overview

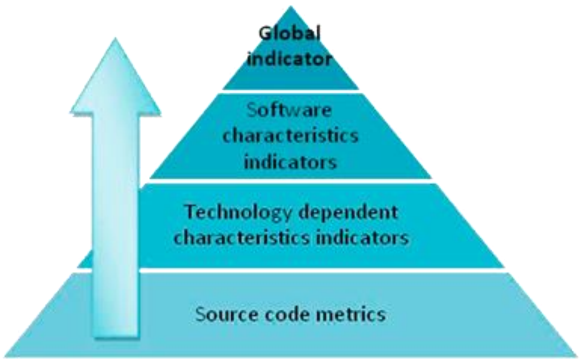
Email	javier.salado@optimyth.com
Name	Javier
Lastname	Salado
Account Enabled	✓
Account owner	Javier Salado (javier.salado@optimyth.con

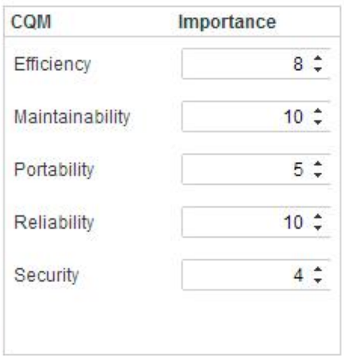
From small shops with only a couple of applications to enterprises with hundreds of applications and several development teams, Kiuwan can suit the needs of any organization size.

The broad range of Kiuwan's features makes it the perfect tool for everybody in the development team and even in other departments. From developers to QA engineers and managers of all types, they all are going to use Kiuwan in different ways for different purposes.

That's why, for any given account, you can create all the kiuwan users that you want, so everybody in the team has her own login and password and don't depend on shared credentials.

Customize your quality model


Global indicator
Software characteristics indicators
Technology dependent characteristics indicators
Source code metrics

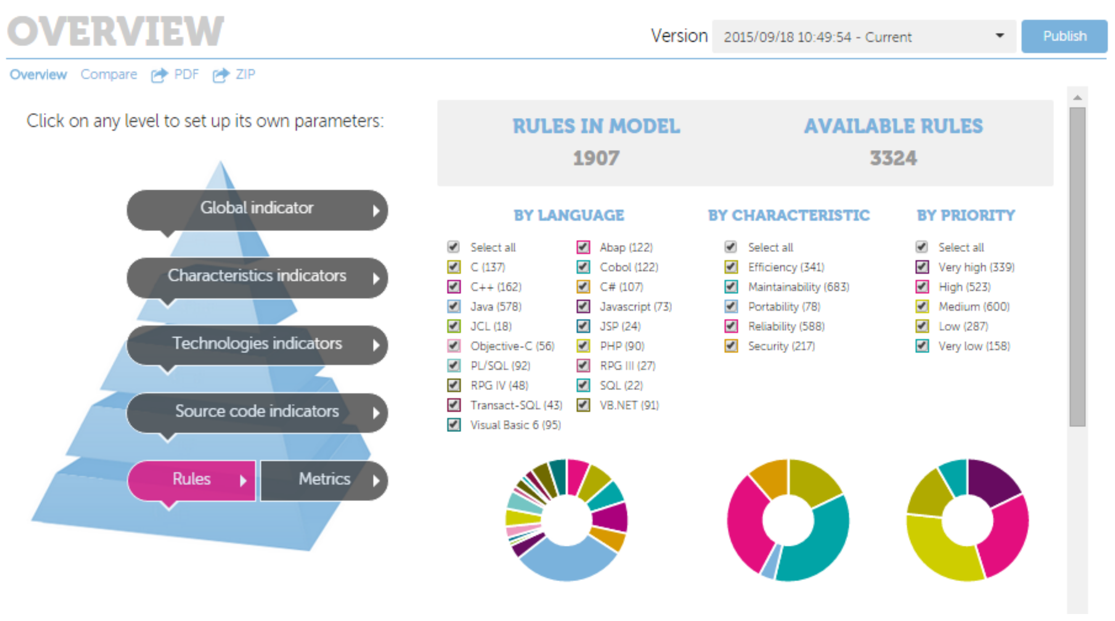


CQM	Importance
Efficiency	8
Maintainability	10
Portability	5
Reliability	10
Security	4

We have tried our best to provide you with the best quality model we could think of, the Kiuwan CQM model.
However, we understand that one quality model may not fit all. Don't worry, we have thought of that, too.
With Kiuwan, you can create all the quality models you may need. You can use CQM as a template or build one from scratch.
So, what can you configure?

- Model thoroughness. Or how strict we are to calculate the quality indicators. This is like at school, having an A in a test depended on the thoroughness of the teacher. There were some with whom getting an A was much easier than with others. With this parameter, you can choose the teacher.
- Importance of software characteristics indicators. You can set the weight of each software characteristic and how they contribute to the global indicator. For example, you may want to give more importance to the maintainability of your applications than to the portability.
- Importance of technologies. When applications are muti-technology, you want to set the weight of each technology and how they contribute to the quality indicators. For example, you may want to give more importance to the quality of the Java code than to the quality of the JSP pages.
- Importance of rule priority. You can set how important are the violations to rules of the different priorities, meaning that you can decide how bad it is to violate a very high priority rule compared to a violation of a rule with medium priority. This is all reflected in the quality indicators.
- Repair difficulty. It is the time a developer has to expend to fix a defect for the diffrent priorities. It will usualy take longer to fix a very high priority violation than a low priority violation.
- Metrics. The metrics that kiuwan calculates when analyzing your code with your quality model.
- Rules. You can enable or disable rules, change their priority and even change the software characteristic they measure. Kiuwan has broad library of over 1,250 rules for you to choose. All of these enriched with a timeline to track all the changes you make in your quality model.

Pretty cool, huh?



Click on any level to set up its own parameters:

RULES IN MODEL 1907

AVAILABLE RULES 3324

BY LANGUAGE

- ✓ Select all
- ✓ C (137)
- ✓ C++ (162)
- ✓ Java (578)
- ✓ JCL (18)
- ✓ Objective-C (56)
- ✓ PL/SQL (92)
- ✓ RPG IV (48)
- ✓ Transact-SQL (43)
- ✓ Visual Basic 6 (95)
- ✓ Abap (122)
- ✓ Cobol (122)
- ✓ C# (107)
- ✓ Javascript (73)
- ✓ JSP (24)
- ✓ PHP (90)
- ✓ RPG III (27)
- ✓ SQL (22)
- ✓ VB.NET (91)

BY CHARACTERISTIC

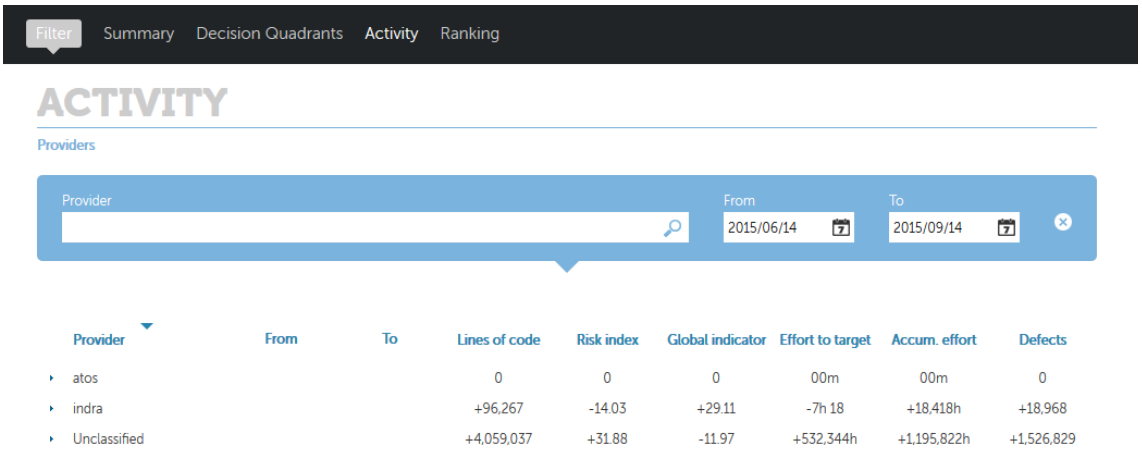
- ✓ Select all
- ✓ Efficiency (341)
- ✓ Maintainability (683)
- ✓ Portability (78)
- ✓ Reliability (588)
- ✓ Security (217)

BY PRIORITY

- ✓ Select all
- ✓ Very high (339)
- ✓ High (523)
- ✓ Medium (600)
- ✓ Low (287)
- ✓ Very low (158)

The QMM dashboard (detail)

Software providers management



ACTIVITY

Providers

Provider

From 2015/06/14

To 2015/09/14

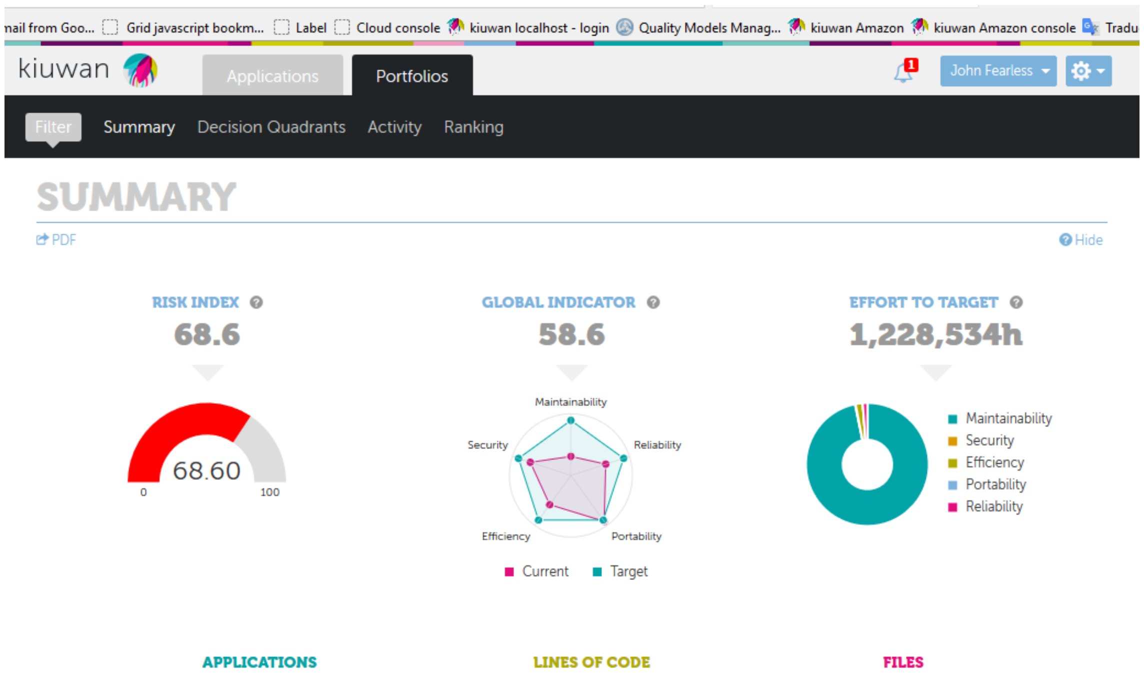
Provider	From	To	Lines of code	Risk index	Global indicator	Effort to target	Accum. effort	Defects
atos			0	0	0	00m	00m	0
indra			+96,267	-14.03	+29.11	-7h 18	+18,418h	+18,968
Unclassified			+4,059,037	+31.88	-11.97	+532,344h	+1,195,822h	+1,526,829

If you outsource your software development or you have different in-house development teams, here is the place where you can define them and manage their activity.

Select a time period and find out everything about you providers' deliveries: lines of code analyzed, your risks, global indicators, effort to target and much more.

Kiuwan is now ready to become the means for a global enterprise providers control service. Start controlling your providers deliveries, SLAs, deadlines and everything outsourcing related.

User defined application portfolios



Another feature that differentiates Kiuwan from the rest.

To create application portfolios, first you have to decide the criteria by which you want to group your applications. For example, the development team or software factory that owns the application, or the business value of your application —this is actually a portfolio group that Kiuwan has by default—; anything that is relevant for you or your organization.

Kiuwan gives two default portfolios: Business value and Providers, but you can create as many portfolios as you want, depending on your own criteria.

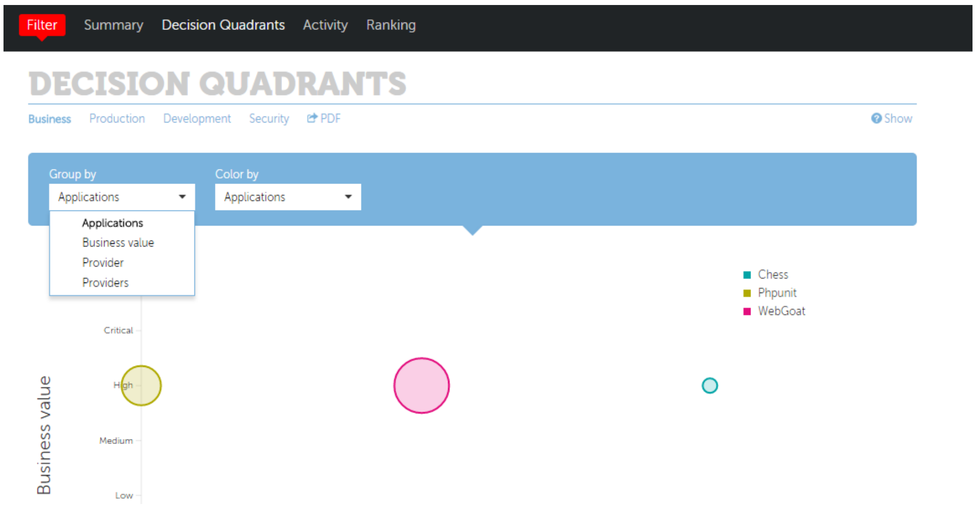
Once you have decided that, you create a portfolio group with the possible values the specific criteria can take. For example, for business value the possible values could be Critical, High, Medium, Low and Very low.

Then, when you create an application —or anytime after that—, you can assign an application to a portfolio by setting the value a criteria is going to have for that application. My On-line Banking application belongs to the business critical application portfolio, for example.

What is so cool about grouping your applications in portfolios? You can manage the health of your applications at this level, meaning that we calculate all the Kiuwan indicators for portfolios and portfolio groups based on the application's data. Kiuwan provides a portfolio perspective where you can track the Kiuwan indicators for the portfolio groups and for the individual portfolios.

For portfolio groups, you can see the contribution of each portfolio to each indicator; and, for each portfolio, you can see the contribution of the individual applications to the portfolio indicators.

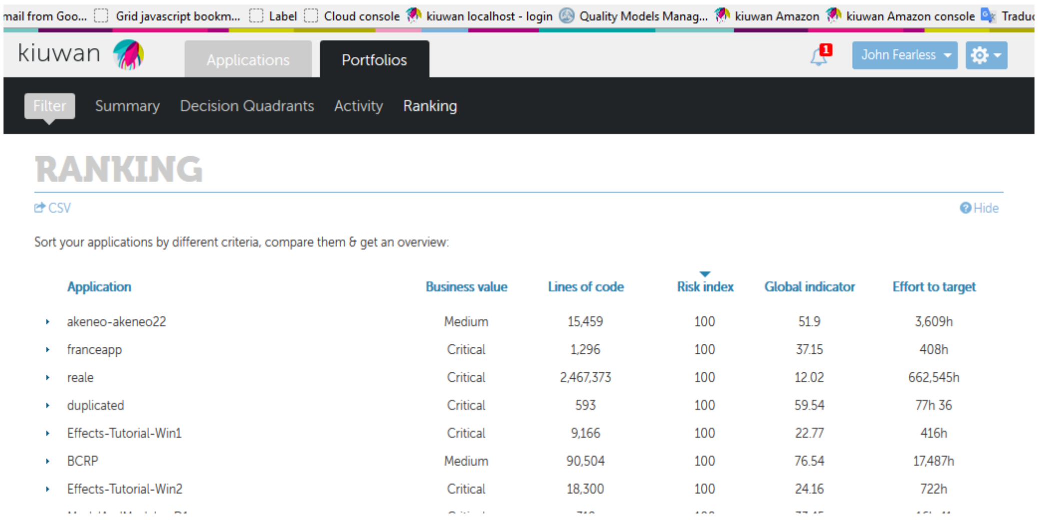
Wait, there is more!



Let us introduce the **decision quadrant**. You can see it for both portfolio groups and portfolios. We position portfolios or applications in a graph with 4 quadrants: **Replace**, **Remediate**, **Observe** and **Conserve**; based on their exposure to development risk and quality. This way you can decide how to improve the health of your portfolios or applications.

The goal should be to *move* all your applications and portfolios to the Conserve quadrant, meaning that your exposure to risk is low and your quality is high. It is a good way to prioritize your development efforts.

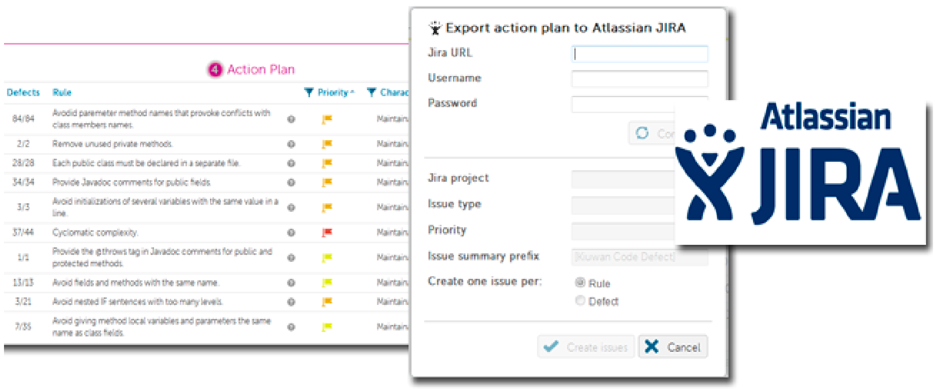
Get a **ranking table of portfolios** as well. At two different levels: portfolio groups giving you the rank of the portfolios in the group, or at the portfolio level giving you the ranking of the applications in a specific portfolio. So you can see what applications or portfolios are the best performers in terms of any of the Kiuwan indicators.



Sort your applications by different criteria, compare them & get an overview:

Application	Business value	Lines of code	Risk index	Global indicator	Effort to target
akeneo-akeneo22	Medium	15,459	100	51.9	3,609h
franceapp	Critical	1,296	100	37.15	408h
reale	Critical	2,467,373	100	12.02	662,545h
duplicated	Critical	593	100	59.54	77h 36
Effects-Tutorial-Win1	Critical	9,166	100	22.77	416h
BCRP	Medium	90,504	100	76.54	17,487h
Effects-Tutorial-Win2	Critical	18,300	100	24.16	722h

Atlassian JIRA integration



Defects	Rule		Priority ^	Charac
84/84	Avodid paremeter method names that provoke conflicts with class members names.	●	🟧	Maintain
2/2	Remove unused private methods.	●	🟧	Maintain
28/28	Each public class must be declared in a separate file.	●	🟧	Maintain
34/34	Provide Javadoc comments for public fields.	●	🟧	Maintain
3/3	Avoid initializations of several variables with the same value in a line.	●	🟧	Maintain
37/44	Cyclomatic complexity.	●	🟥	Maintain
1/1	Provide the @throws tag in Javadoc comments for public and protected methods.	●	🟨	Maintain
13/13	Avoid fields and methods with the same name.	●	🟨	Maintain
3/21	Avoid nested IF sentences with too many levels.	●	🟧	Maintain
7/35	Avoid giving method local variables and parameters the same name as class fields.	●	🟨	Maintain

Export action plan to Atlassian JIRA

Jira URL	
Username	
Password	
Jira project	
Issue type	
Priority	
Issue summary prefix	[Kiuwan Code Defect]
Create one issue per:	[✓ Rule] Defect

Picture this: you decide to have two developers working on improving the maintainability of your application for a couple of weeks. Where do they start? You know you have hundreds of maintainability defects, but which are the ones that affect your application the most? Which are the ones that the developers can handle in those two weeks of work?

That is exactly what the 'What If' analysis feature can do for you! Just say you want to use 160 hours of development time to fix maintainability defects and click the action plan. Kiuwan will give you the list of defects to work on.

Now, wouldn't it be cool to have those defects as open issues in JIRA, so you and the developers can track them? Here you have it! Just click on the Atlassian Jira export icon and off you go. We can connect to your own JIRA instance or to your JIRA On-Demand instance, it doesn't matter. Select your JIRA project, decide the issue type from the ones you have defined in JIRA, assign the priority and create the issues. You can create issues per rule or per defect, it's up to you.

How cool is that? A new cool feature on top of the coolest feature in Kiuwan. That is what our users tell us!

RESTful API

There is a whole world out there that may need the information Kiuwan can provide. Just connect...

Kiuwan has an exciting new feature: a RESTfull API that will allow anyone to pull out data from Kiuwan to use whenever is needed, wherever is needed.

So far, you have all the information about the applications you have analyzed with Kiuwan. Stay tuned for more, since we are expanding the API as you read this.

- List of applications.
- Last application analysis results, including the Kiuwan indicators with details and main metrics.
- Application defects.
- Application files with defects and metrics.
- Defects deltas for any given application's analyses.

You will find it very flexible and useful. For the technical details on how to use it and a description of all the available functionality, check the developers documentation.

Jenkins plugin

Continuous code analysis. That is our mantra: measure the quality of your software and measure it as often as you can. If you have a continuous integration and deployment process driven by Jenkins, we have very good news for you: you can now easily integrate continuous code analysis into your process with Kiuwan's new Jenkins plugin.

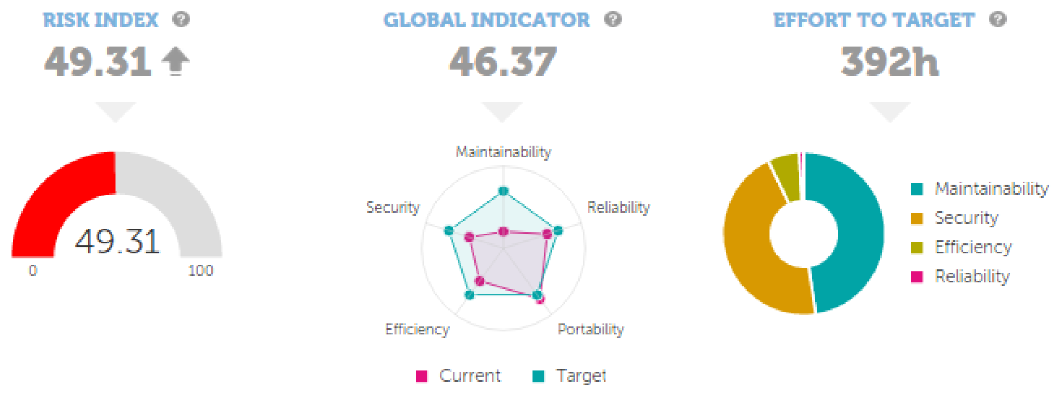
We made the plugin open source, so the community can share it and improve it. You can download it from our GitHub repository or from the Jenkins CI repository.

If you are one of those who care about software quality and use Jenkins to steer your development process, just register for a free Kiuwan account. Bear in mind that to use the plugin from your Jenkins, you will need the Kiuwan local analyzer installed on the same machine. If you want to learn how easy it is to install the Kiuwan local analyzer, check out the tutorial.

Now, download the plugin, configure it in your Jenkins installation and you are ready to go! You will be able to run a Kiuwan static analysis after your build and have a link to the results in the Jenkins build execution page.

If you need more help, you can refer to our Jenkins plugin page in the product documentation.

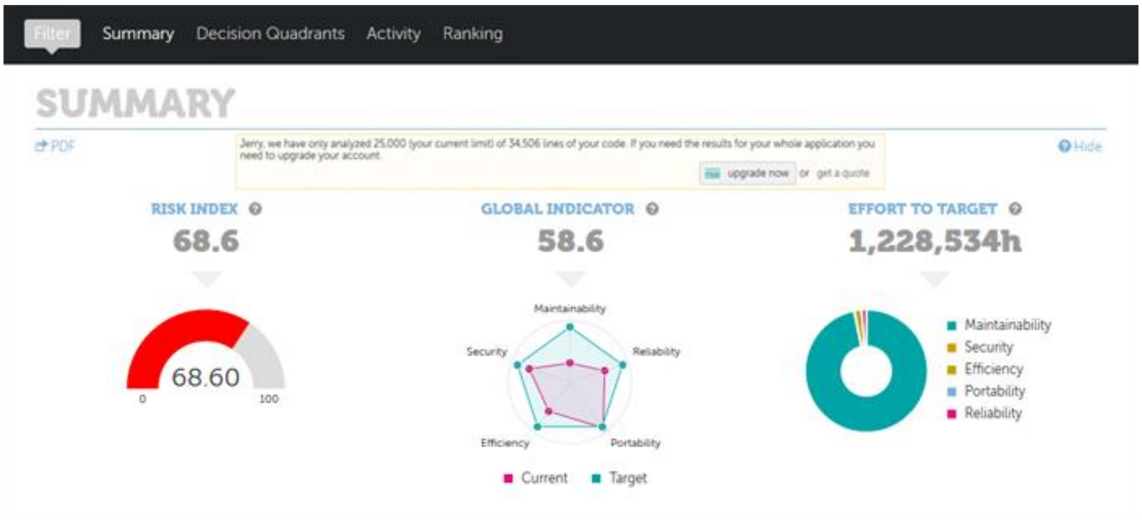
New risk index calculation



Size matters, at least when we are talking about managing software quality. Kiuwan's new risk index have the relative size of your application into account in a new way that will give you a closest estimation of the risk your are facing in your development.

There are two main variables that yield the development risk: quality and effort to fix problems (technical debt, if you will). Before the effort affected linearly to the risk index calculation. Now, the effort is normalized taking the size of the application into account. Thus, small efforts in small applications may imply higher risk, and big efforts in big application may imply lower risk.

Partial analysis results



Jerry, we have only analyzed 25,000 (your current limit) of 34,506 lines of your code. If you need the results for your whole application you need to upgrade your account.

That's right! Now you will always get results from your analysis, even if the application you are analyzing exceeds the size limits of your account.

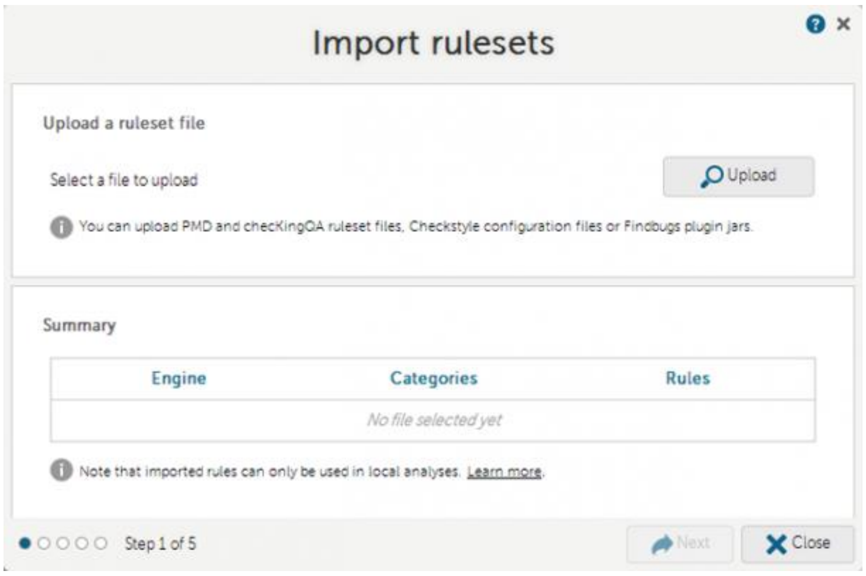
Of course we don't give you the results for the whole application, that wouldn't be fair for our paying customers and would make Kiuwan a free service. We still want to be a profitable business.

Nevertheless, we will give you results taking into account what we found and measured for the first lines of code up to current account limit. So, if you analyze an application with 50Kloc we will give you the results for the first 25Kloc, if you have a free account. This will give you a fair idea of what Kiuwan can do for the quality of your application, but we strongly recommend to use these results with care, since they don't reflect your whole application.

For these partial analyses, the icon to represent the status of the analysis will be an orange thumb sideways, as opposed to the green thumb up for a complete successful analysis or the red thumb down for a failed analysis. Additionally, you will see a banner on all screens warning that the results correspond to the lines of code of your current limit only.

You can always upgrade your account to get full results.

Reuse your PMD, Checkstyle and FindBugs rulesets


Import rulesets

Upload a ruleset file

Select a file to upload

You can upload PMD and checKingQA ruleset files, Checkstyle configuration files or Findbugs plugin jars.

Summary

Engine	Categories	Rules
	No file selected yet	

Note that imported rules can only be used in local analyses. Learn more.

Step 1 of 5

That's right! All the rules that you currently use with your, until today, favorite open source static analysis tool, PMD, Checkstyle or FindBugs, can be imported to your personal Kiuwan rules library. You can add them to your custom quality models, categorize them by software characteristic affected (Efficiency, Maintainability, Reliability, Portability or Security) and assign them a priority and a difficulty to repair to know the effort needed to fix the defects found based on these rules.

That's not all! You can take full advantage of all Kiuwan features, taking into account the defects provided by violations to these rules. The action plan from a 'what if' analysis, the possibility to mute defects coming from this tools to, for example, manage false positives, generate comprehensive reports and more.

Just one little caveat: to take advantage of this, you need to run your analysis locally using the Kiuwan local analyzer since it embeds the corresponding 3rd party engines. You just have to add the jar files with your 3rd party rules implementation in the Kiuwan local analyzer lib.custom directory and your rules will be executed when needed and will incorporate the results to the ones from the Kiuwan analyzer itself.

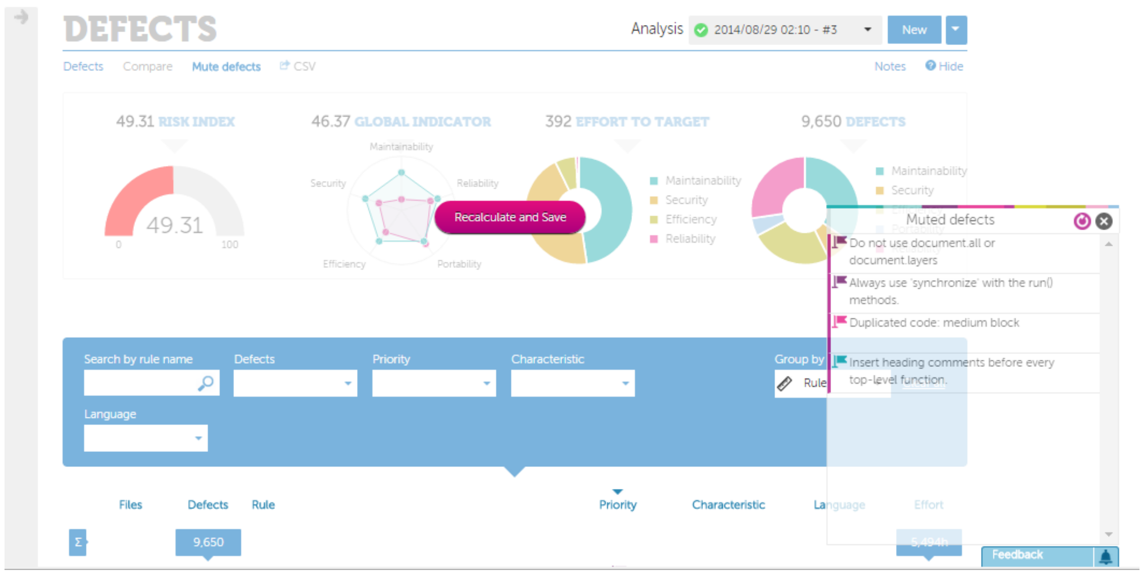
These are the versions of the 3rd tools supported and incorporated to our local analyzer:

- PMD: 4.2.5
- Checkstyle: 4.4
- Findbugs: 1.3.9

You can find the documentation to learn how to use this feature and all the details in the product documentation.

Go ahead, reuse your custom rules and unleash the full power Kiuwan provides to write better applications.

Defects mute



Muted defects:
- Do not use document.all or document.layers
- Always use 'synchronize' with the run() methods.
- Duplicated code: medium block
- Insert heading comments before every top-level function.

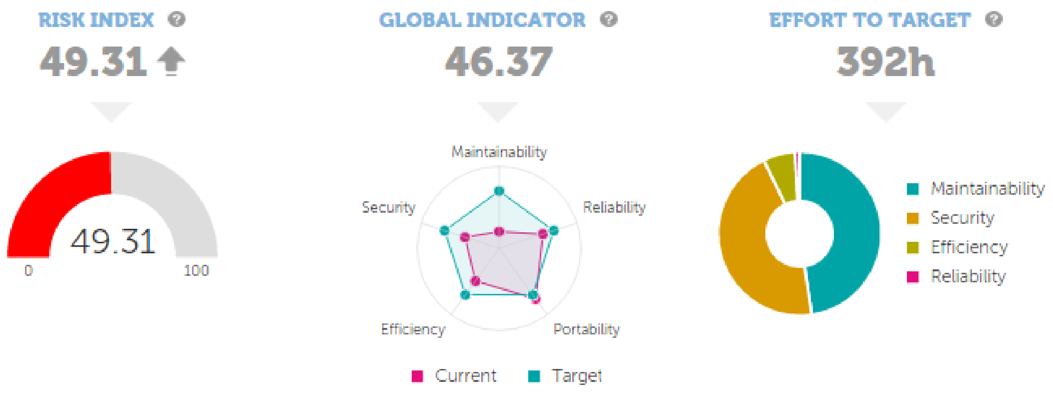
This is probably going to be one of the most popular Kiuwan features.

Now you can mute defects for the whole application, for a specific file or just for a specific line of code in a file, by **dragging and dropping** the rule that triggers the defects, the file or the line of code, it occurs in the defects mute box available directly on the defects page. **All the Kiuwan indicators are recalculated on the fly** and all the **muted defects** are saved to be taken into account in subsequent analyses. It is a feature that will give you a lot of flexibility with a very easy to use interface.

For each mute, you can specify a reason why they are muting the defect and a comment.

Whereas it is a false positive in a file, or a line of code that shouldn't be taken into account, or a file has too many defects of the same kind, or the line of code with the defect is generated code and has to be left out, or any other reason that is good enough for the user.

Trends

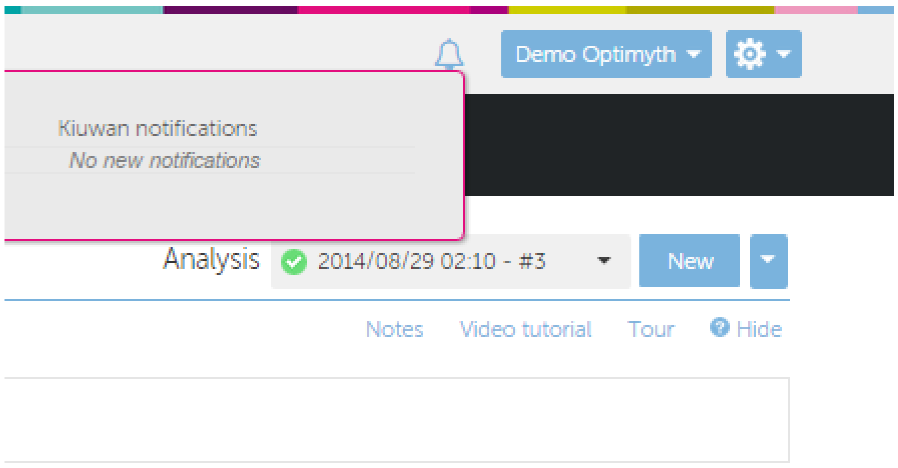


We've been asked to have this feature and we always listen to our users. If it is good for you, it is good for us.

The arrows under each metric and above each indicator will immediately tell you if you have improved or not since your last analysis. Hover on the arrow to see the previous value and your improvement percentage. It may be very simple, but this gives a lot of information for you to take action and continuously measure and improve the quality of your applications.

Every time you see a red arrow, go ahead, run a 'what if' analysis, for example, to get a detailed action plan to improve your quality.

Notifications



Information is gold. As a cloud application user, you want to know what is new in the platform you use every day.

From now on, stay tuned. Check our notifications bell and learn about:

- New deployed features on the platform.
- Recommended & new tutorials.
- New available rules in the CQM quality model.
- New languages suported.
- Interesting blog posts.
- Maintenance windows.
- Other announcements.