

Impact Analysis

- Impact Analysis
 - Starting Components Selectors
 - Navigation Constraints
 - Relations
 - Direction
 - Depth
 - Additional Filters
 - Impacted Components Filter
- Results

Impact Analysis



Impact Analysis is probably the most powerful functionality of Kiuwan Architecture.

Basically, Impact Analysis lets you to deeply inspect your application for whatever criteria you need, allowing you *to discover unknown paths between your components*.

Impact Analysis provides insights on quite different, but important, questions such as:

- To check if software elements are compliant with architectural standards (e.g. detecting illegal dependencies).
- To perform impact analysis (identify items that may be affected by a change in a specific set of items).
- To detect potential design antipatterns ('smells') due to bad couplings.
- To resolve the group of software items that belong to a certain category (classification by tagging).
- To keep track of the code items that make up a certain logical entity.



Impact Analysis "discovers" connections (paths) between components of your source code.

For example, let's suppose you want to know how your J2EE application is accessing the database, i.e. which Java classes are accessing data tables.

If you don't know internally the application, you might be interested to know if the app is using some core classes or freely accessing the database.

Or even more, you could be imposing the use of some core classes to access the database, but you want to discover if there's any class that is not using those core classes and goes directly to the database (something that you probably consider as an architecture violation).

Kiuwan Architecture lets you answer those questions.



First, you should translate your question to Kiuwan Architecture concepts (**Starting and Impacted components**)

- Tables will be the "Starting Components"
- Java classes will be the "Impacted Components"

Kiuwan lets you fully specify conditions on starting and impacted components.



Besides to specify impacted and starting components, you can also specify **Constraints** about how to navigate through the relationships.

By default, Kiuwan will find any path of any type, but you can restrict those paths.

In order to make an Impact Analysis, Kiuwan' Analysis Impact page looks as follows.

ADD STARTING COMPONENTS SELECTOR

Starting components that will be added:

→ Components matching filter: ALL

Name contains

Name NOT contains

Language

Type

Analyzed / Inferred

File

Tag

+ Filters

Clear

▶ Select specific components from this filter

APPLY



The starting Components Selector dialog lets you *specify starting components in two ways*:

1. Select *all the components that match the filter* (now and in the future)
2. Select *a subset of the components* found with the filter

In our example, we are looking for “Table” components as “starting” components. Not specific tables, but any table that exists in the application.

So you specify “Table” as component type, click on **Apply** button and Kiuwan will display the list of components that match that filter.

A

→

B

STARTING COMPONENTS SELECTORS

Components matching filter: type is Table

×

Add

Clear

Instead, if you were looking for some subset of specific tables, you specify Table as component type (as before), but you should click on **“Select specific components from this filter”**

This way, you will get a list of found components where you can pick up those that you are interested in.

ADD STARTING COMPONENTS SELECTOR

Starting components that will be added:

→ Selected components in the list.

▼ Discard current component selection

18 of 18

	Name	Type	Language
<input checked="" type="checkbox"/>	AUTH	Table	
<input type="checkbox"/>	EMPLOYEE	Table	
<input checked="" type="checkbox"/>	MESSAGES	Table	
<input type="checkbox"/>	MFE_IMAGES	Table	
<input type="checkbox"/>	OWNERSHIP	Table	
<input checked="" type="checkbox"/>	PINS	Table	
<input type="checkbox"/>	PRODUCT_SYSTEM_DATA	Table	
<input type="checkbox"/>	ROLES	Table	
<input type="checkbox"/>	SALARIES	Table	
<input type="checkbox"/>	TAN	Table	
<input type="checkbox"/>	TRANSACTIONS	Table	
<input type="checkbox"/>	USERS	Table	
<input type="checkbox"/>	USER_DATA	Table	
<input type="checkbox"/>	USER_DATA_TAN	Table	

APPLY

After clicking on **Apply** button you will get your starting set of components.




STARTING COMPONENTS SELECTORS

3 component(s) hand-picked ('PINS', 'MESSAGES', 'AUTH')

Add
Clear


The difference is clear:

- In the 1st case (*filter*), paths will be searched for all the tables in the application.
- In the 2nd case (*hand-picked*) , only paths from those tables will be searched.

Following our example, we are interested in all the tables, so we will select 1st option.


Next, we should define *Navigation Constraints*. The next section will show you how to do it.

Navigation Constraints

- 

Besides specifying starting components, you can also specify *constraints about how to navigate through the relationships*.

By default, Kiuwan will find any path of any type, but you can restrict those paths.



NAVIGATION CONSTRAINTS

Relations	Direction	Depth
<input type="text"/>	<input type="text" value="Outgoing"/>	<input type="text" value="1"/>

Relations

Any relation between components belongs to a certain type.

You can specify what relations should be considered when searching for paths between components. Kiuwan shows all the different relations found in your code.

2 NAVIGATION CONSTRAINTS

Relations	Direction	Depth
<input type="text"/>	<input type="text" value="Incoming"/>	<input type="text" value="1"/>

COMPONENTS FILTER

Name NOT contains	Language
<input type="text"/>	<input type="text"/>

Direction

Besides a relation type, any relation has a direction: *Incoming* or *Outgoing*.

- By **Incoming**, we mean those relations “ending” at the Starting components
- By **Outgoing**, we mean relations “starting” from the Starting components.

2 NAVIGATION CONSTRAINTS

Relations	Direction	Depth
<input type="text"/>	<input type="text" value="Incoming"/>	<input type="text" value="1"/>

In our example, we are looking for Java classes that are accessing Tables. In this case, usual relations of a java class with a table (insert, delete, etc.) go from the java class to the table, so in this case, we could select “Incoming”.

In case of doubt, select “Any” and reduce afterward your search after inspecting the results.

Depth

A pair of components (A and C) can be connected directly (A-C) or indirectly (through intermediate components, A-B-C).

Depth specifies the maximum number of connection links between components when searching for paths.

For example, given A-B-C-D-E, if you specify depth=3, A-D is a valid path (as well as A-B and A-C, of course), but A-E will not be.

Then, following our example, it makes sense to specify:

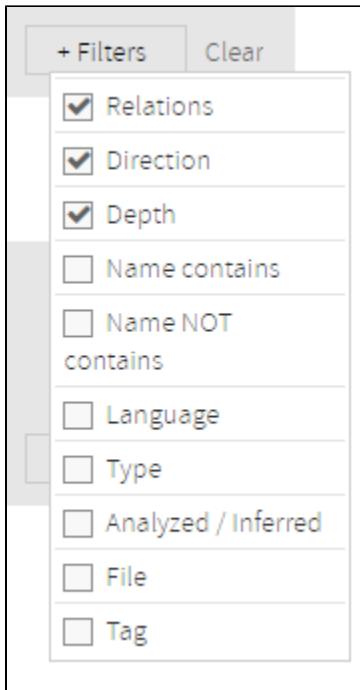
- Relations: empty -> we want all the relations
- Direction: Incoming (or Any) -> those relations going to the tables
- Depth: 1 -> only direct connections

Additional Filters

A pair of components (A and D) can be connected through different intermediate components: A-B-D and A-C-D).

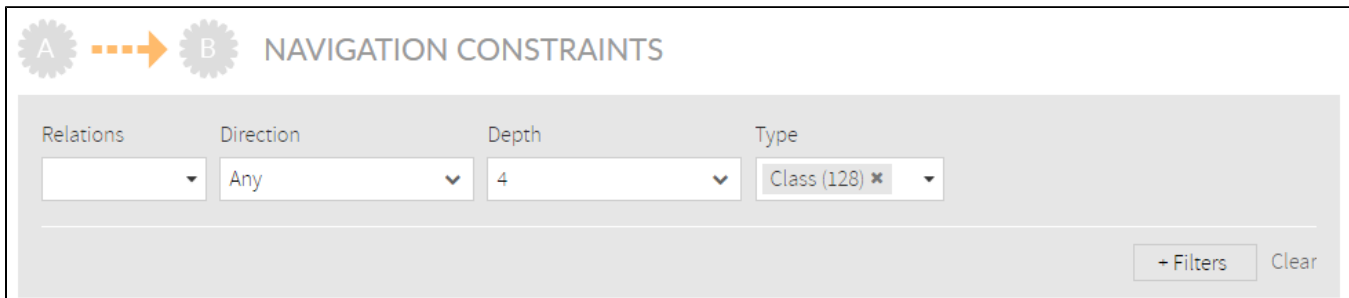
You can even restrict the resulting set of paths specifying that *only paths through a certain type of components are searched*.

To do it, you can click on the **Filters** button, where you can select criteria for searching intermediate components.



A screenshot of a 'Filters' menu. At the top are two buttons: '+ Filters' and 'Clear'. Below them is a list of filter options, each with a checkbox. The first three options, 'Relations', 'Direction', and 'Depth', are checked. The remaining options are 'Name contains', 'Name NOT contains', 'Language', 'Type', 'Analyzed / Inferred', 'File', and 'Tag', all of which are unchecked.

For example, if you select Tables as starting points and depth=4, paths will be found that include procedures and classes. But if you were interested only in paths through classes, you could specify it including Type selector and choosing Class type.



A screenshot of the 'NAVIGATION CONSTRAINTS' interface. At the top, there is a diagram showing a grey gear icon labeled 'A' connected by a dashed orange arrow to another grey gear icon labeled 'B'. Below this, the title 'NAVIGATION CONSTRAINTS' is displayed. Underneath the title is a row of four dropdown menus: 'Relations', 'Direction', 'Depth', and 'Type'. The 'Relations' dropdown is empty. The 'Direction' dropdown is set to 'Any'. The 'Depth' dropdown is set to '4'. The 'Type' dropdown is set to 'Class (128)' with a small 'x' icon next to it. At the bottom right of the interface are two buttons: '+ Filters' and 'Clear'.

Impacted Components Filter



Besides specifying starting components and navigation constraints, you can also specify a *filter for impacted (reached) components*.

If you do not specify any filter, most probably Kiuwan will show you a huge amount of results. To further concrete your query you can use this dialog.

The dialog box is titled "IMPACTED COMPONENTS FILTER" and features a visual representation of component flow from A to B. It contains several input fields for filtering: "Name contains", "Name NOT contains", "Language", "Type", "Analyzed / Inferred", and "File". Each of these fields has a search icon. Below these is a "Tag" field with a dropdown arrow. At the bottom right, there are two buttons: "+ Filters" and "Clear".



This dialog let you compound any filter that will act against the result set.

It works exactly as the Component search filter of Components page. Please visit [Components Search Criteria](#)

In our example, we are looking for Java classes that are accessing data tables, so we should specify that condition in the filter: Language=Java (or Type=Class).

Now, we are ready to run Impact Analysis clicking on **Analyze** button.

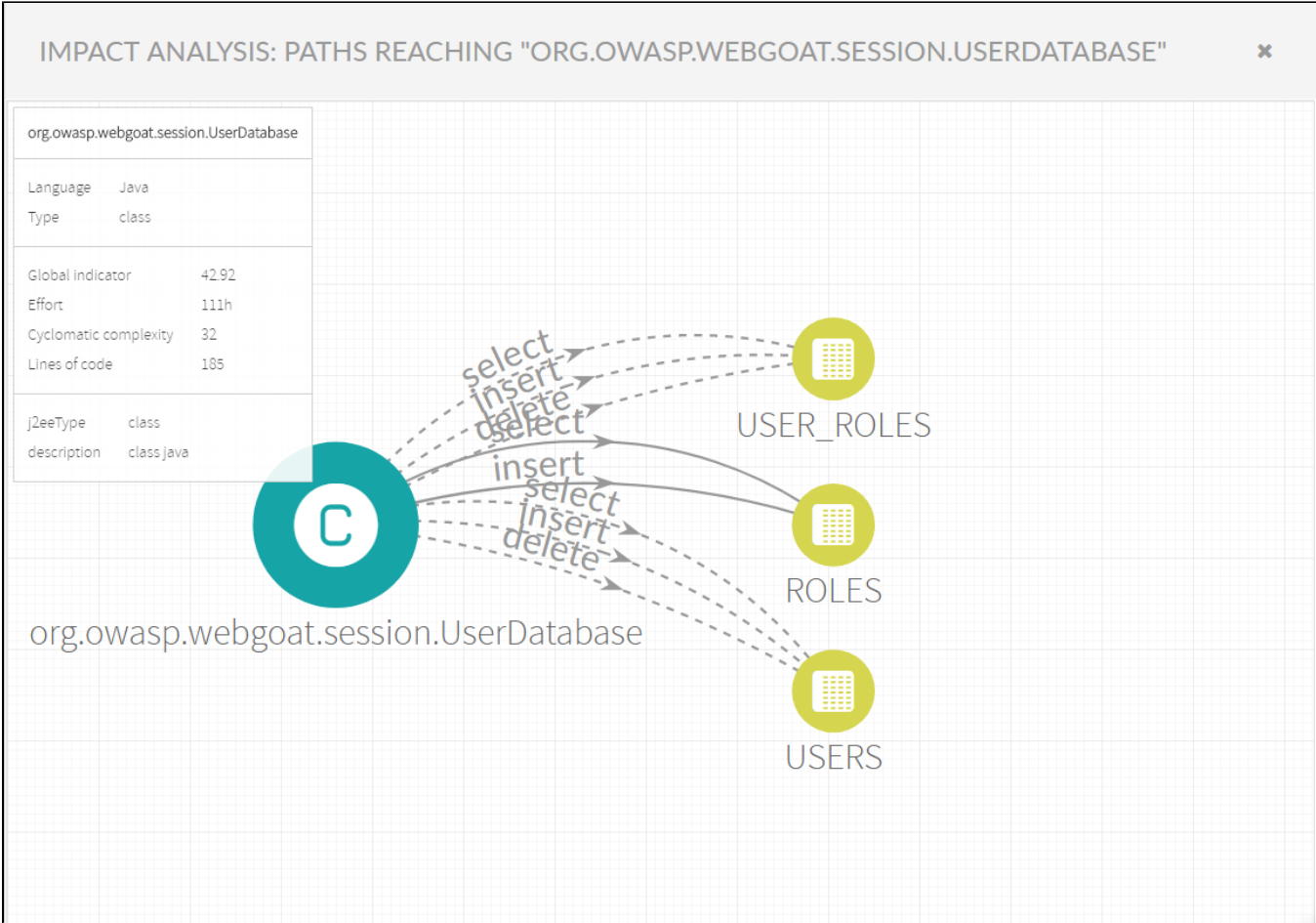
Results



Results section will *list all the "impacted" (reached) components*, i.e.:

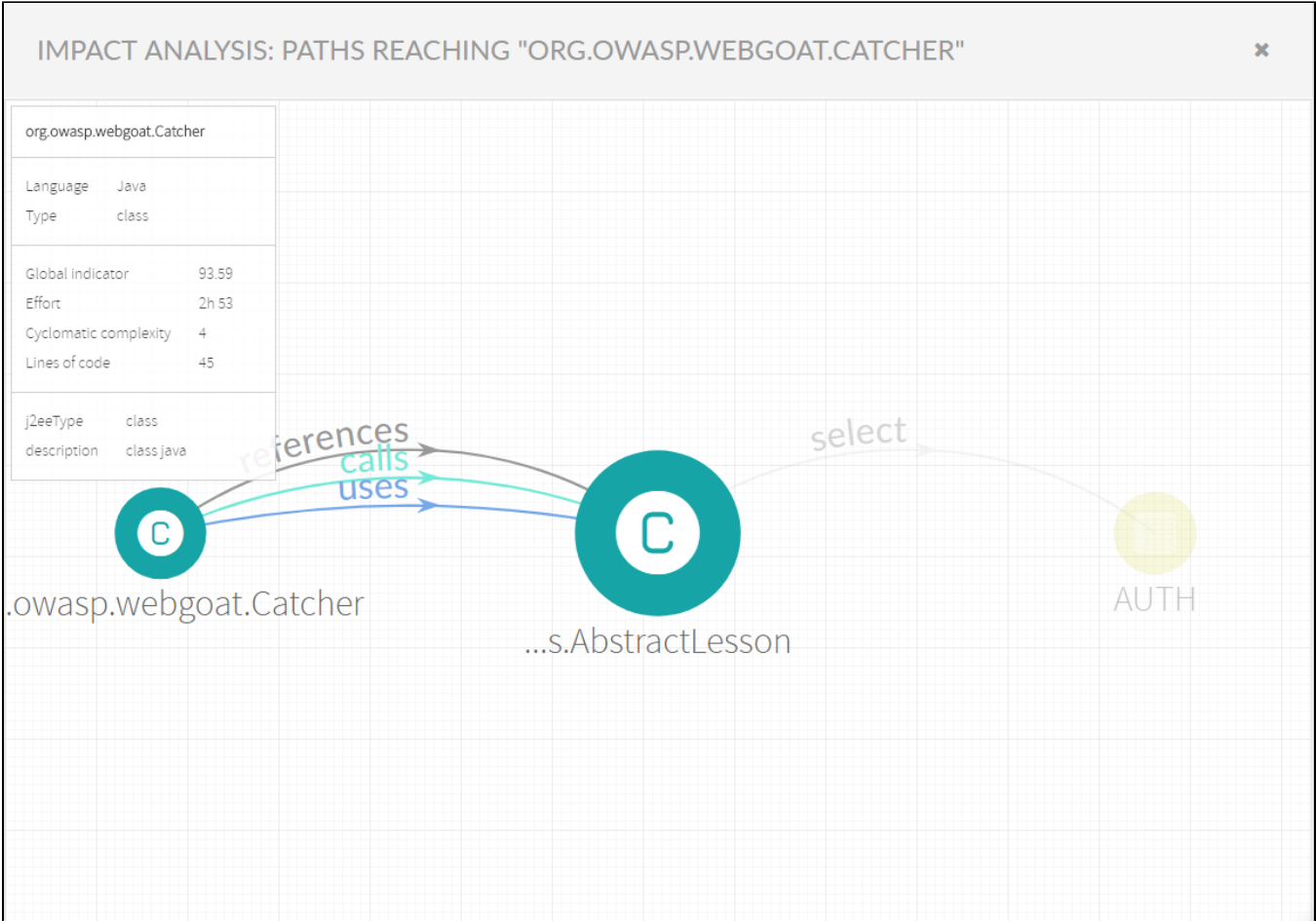
1. reachable from the starting set of components,
2. complying with the specified navigation constrains, and
3. satisfying the specified filters

Kiuwan will show the total number of components as well as a list of them.



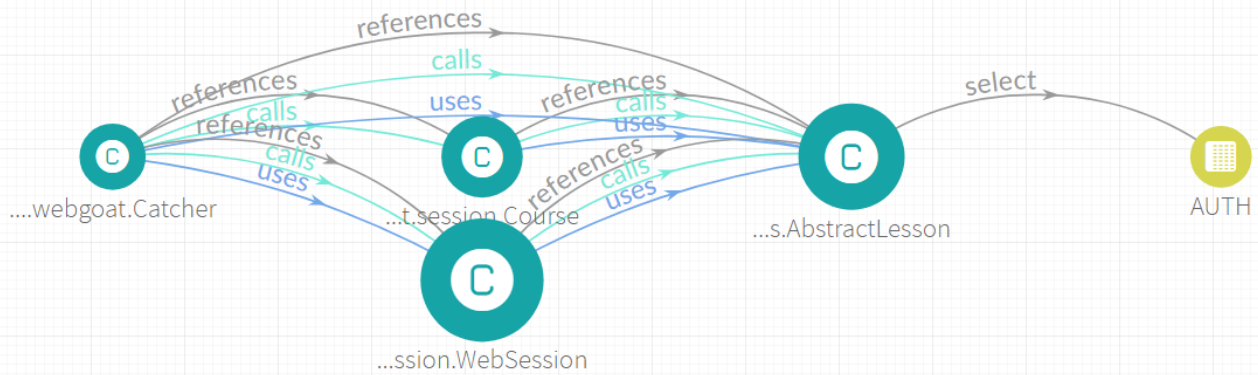
This graphic is quite simple because depth=1, but in case of higher values that graph will be very valuable to your goals.

Example for depth=2



and depth=3

IMPACT ANALYSIS: PATHS REACHING "ORG.OWASP.WEBGOAT.CATCHER"



As you can guess, as we increase the dept value, the number of paths exponentially increases.