CROSSWAY USER MANUAL

1. Introduction

What is CrossWay?

- CrossWay is a Progress Developer Studio for OpenEdge 12.8 Eclipse plugin enabling users to generate various visual representations of the 4GL code interaction within the PDSOE workspace

Who Should Use It?

- The purpose of the plugin is to be used by developers, business analysts, testers, software architects and other roles requiring the analysis over the impact of the 4GL code changes and the interaction between the 4GL files of the projects within the same workspace

2. Installation & Setup Requirements

Minimum System Specifications:

- Progress OpenEdge 12.8 installed (Developer Studio component included).
- Progress OpenEdge 12.8 compilable code base.

Required Dependencies:

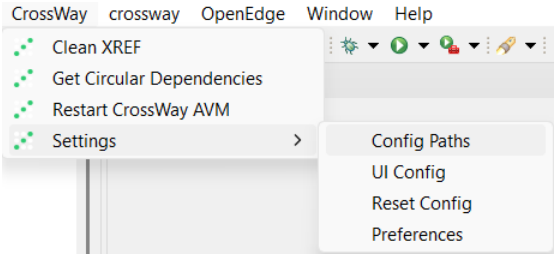
- draw.io executable or installed.
- CrossWay Visualizer installed.

Installation Steps:

- Follow steps for installing the CrossWay Plugin: CROSSWAY PLUGIN INSTALLATION MANUAL.

Configuration:

- After the CrossWay Plugin was successfully installed in Developer Studio, an initialization of the plugin will take place over the Openedge project which is intended to be analyzed through CrossWay.
- First select the Openedge project in Project Explorer and then access CrossWay → Settings → Config Paths option in the top menu of Developer Studio. This will automatically determine and copy the settings of the Openedge project (PROPATH entries, DB connections, RootPath, Technical Impact Path etc.) to the CrossWay configuration.
- You can confirm and adjust the project's CrossWay configuration by accessing the CrossWay → Settings → Config Paths option in the top menu.



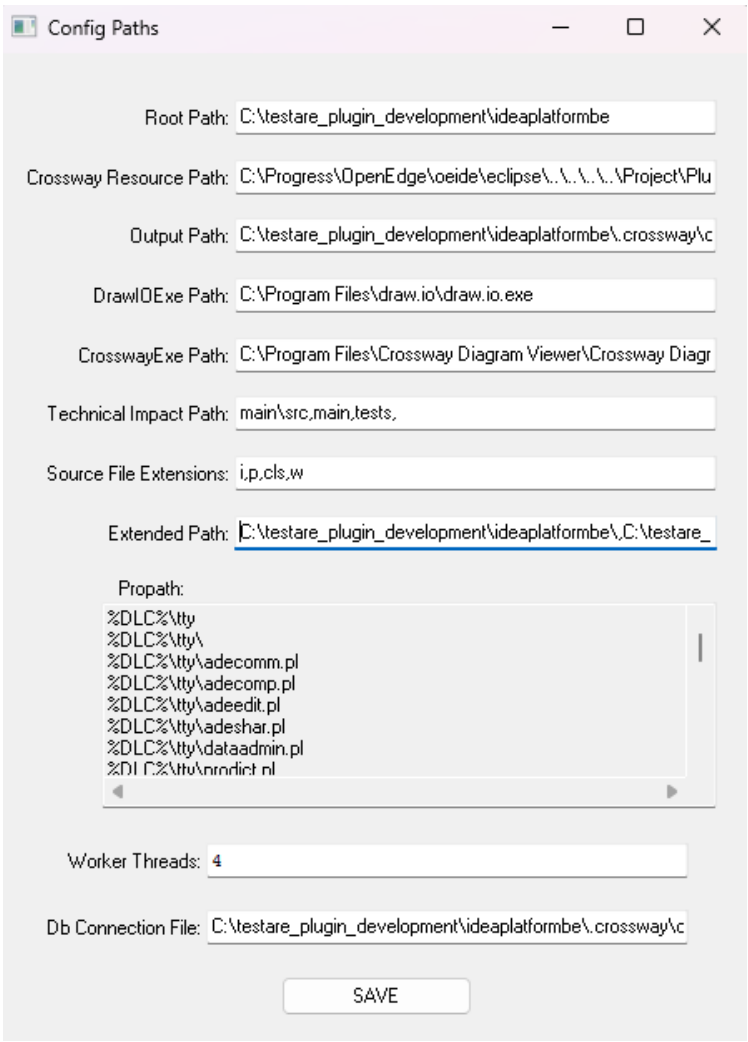
- There is also a separate configuration screen for the UI settings that can be accessed from the CrossWay menu → Settings → UI Config.

- The CrossWay menu → Settings → Reset Config will enable users to go back to the default determined settings in case manual changes over the configurations were performed.

- The CrossWay menu → Settings → Preferences options allows the user to specify the 'draw.io.exe' and 'CrossWay Visualizer.exe' locations on the file system. Setting the values here will automatically update the CrossWay initialized Openedge projects' configuration accordingly.

Config Paths

- Config Paths window has multiple options for configuring all the paths necessary for CrossWay to work properly.
- Below, the paths are configured for a project called IdeaPlatformBE which is an OERA OpenEdge Project that also contains unit tests.



- Root Path:

  Absolute path to the OpenEdge project root folder.
- CrossWay Resource Path:

  Absolute path to the CrossWay folder which contains all the necessary libraries to use CrossWay. The location is under the plugin installation folder on C:\Progress\OpenEdge.
- Output Path:

  Absolute path to a folder where CrossWay files are generated. In this example the folder where CrossWay will generate the output is under the .CrossWay folder under the project that is using CrossWay.
- draw.ioExe Path:

  Absolute path to the draw.io.exe file.

  The draw.io.exe file can be downloaded and installed from here.
- CrossWayExe Path:

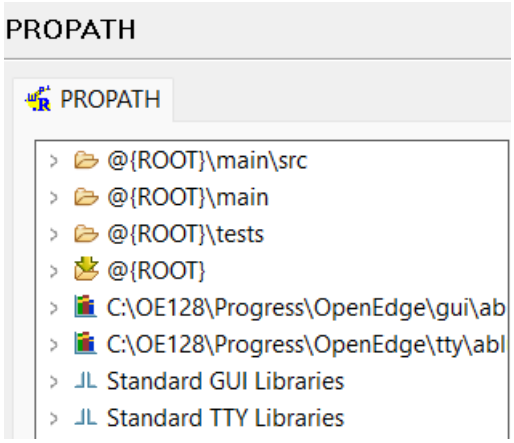
  Absolute path to the CrossWay Visualizer.exe file.

CrossWay Visualizer can be downloaded and installed from here.

- Technical Impact Path:

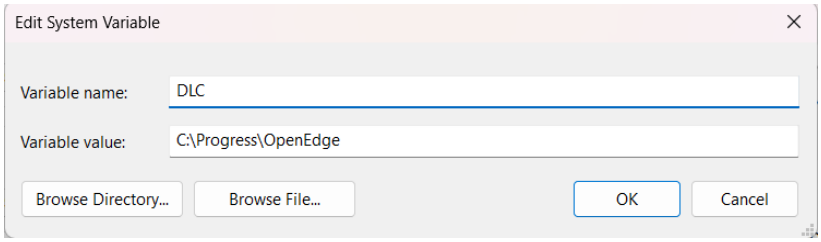
  A comma-separated list of workspace-relative folder paths of the files CrossWay should process (e.g., src, main and tests folders from the project). The values are populated with values from Propath. If the Propath looks like in the next screenshot, then the Technical Impact Path will contain "main\src,main,tests,".

  The "," from the end represents the @ROOT in the Propath. The algorithm takes into consideration the relative path starting from the @ROOT, meaning that the root of the project itself will be an empty string.



- Source File Extensions:
  - A list of all file extensions to consider when reading files from folders.
- Extended Path:
  - A comma-separated list of propath entries from current project settings that allow for all pieces of code from TechnicalImpactPath to compile.
  - In this example the ExtendedPath contains the following:
    - C:\Project\ideaplatformbe
    - C:\Project\ideaplatformbe\main\src
    - C:\Project\ideaplatformbe\tests
    - C:\Project\ideaplatformbe\main
- Propath:
  - The Propath needed for background progress sessions collecting the XREF information.
  - For the propath parameter to work, you must declare an environment variable called DLC with the value of the OpenEdge folder.
  - Click on Windows Key, search for 'environment variables', then click on Edit the system environment variables.

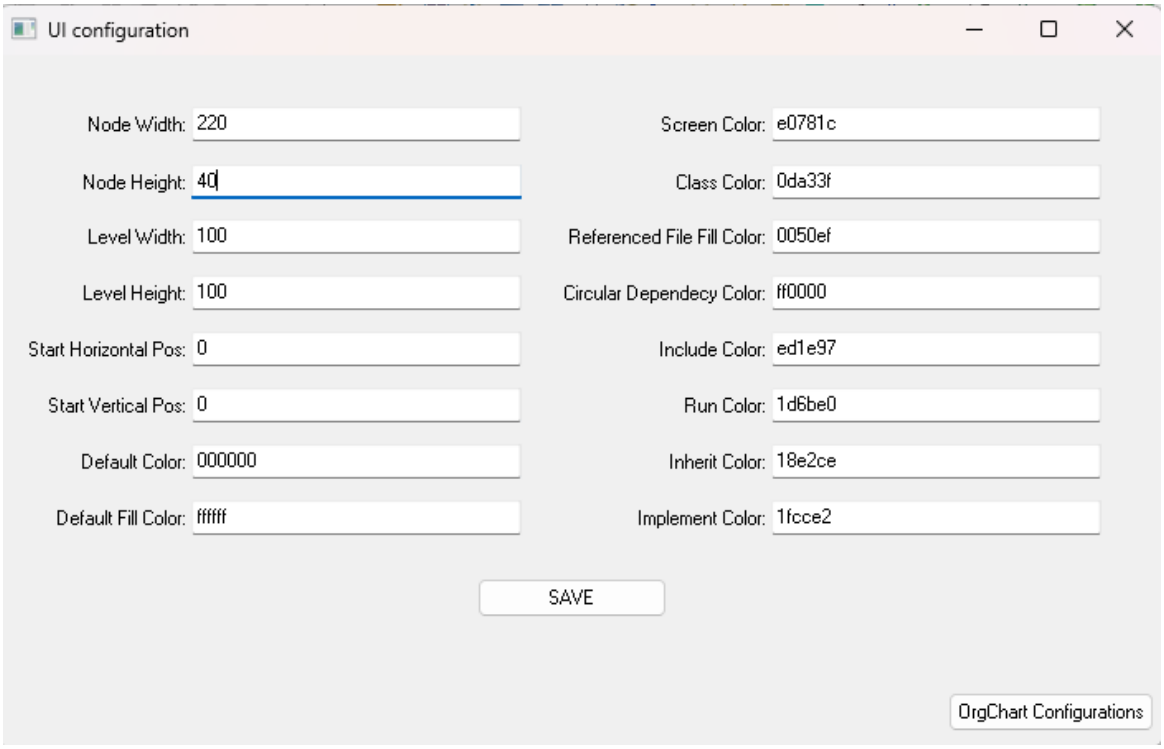
- Go to ‘Advanced’ tab, press Environment Variables and add a new variable under the System Variables section providing the absolute path to the OpenEdge folder.



- WorkerThreads:
  - Number of background agents to use for collecting the XREF information.
- DbConnectionFile:
  - The relative path to the db_connection.pf file inside the .CrossWay folder.

UI config

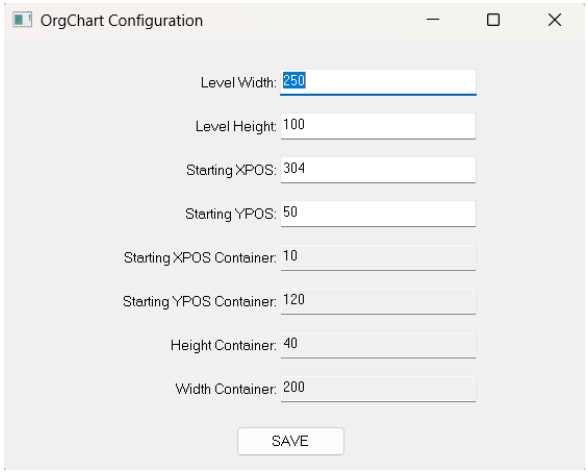
- This UI Configuration window provides multiple options to configure all the visuals for the diagram such as dimensions, spacing, link colors, node background color, node border color depending on different cases and what the node or link represents.



Property	Meaning
Node Width	The node width.
Node Height	The node height.
Level Width	Horizontal distance between 2 nodes.
Level Height	Vertical distance between 2 nodes.
Start Horizontal Pos	Horizontal position of the first node of the diagram.
Start Vertical Pos	Vertical position of the first node of the diagram.
Default Color	Default font color for nodes.
Default Fill Color	Default fill color for nodes.
Screen Color	Border color for nodes representing a .w file.
Class Color	Border color for nodes that represent a class and for corresponding links.
Referenced File Fill Color	Color for the referenced file node.
Circular Dependency Color	Link color between nodes that have a circular dependency.
Include Color	Border color for nodes that represent an include file and for corresponding links.
Run Color	Border color for nodes that represent a procedure and for corresponding links.
Inherit Color	Color for inherit links.
Implement Color	Color for implement links.

OrgChart Configurations

- Besides the usual configuration, the user can also set the OrgChart parameters for the container around the Inheritance Diagram.
- It can be opened by clicking the 'OrgChart Configurations' button in the above UI Configuration screen.
- The last four parameters are not meant for the user to configure.

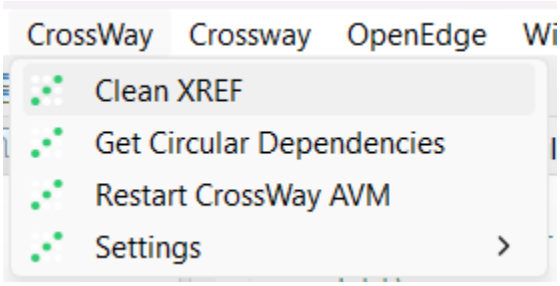


Property	**Meaning**
Level Width	Horizontal distance between 2 nodes.
Level Height	Vertical distance between 2 nodes.
Starting XPOS (Start Horizontal Pos)	Horizontal position of the first node of the diagram.
Starting YPOS (Start Vertical Pos)	Vertical position of the first node of the diagram.
Starting XPOS Container	Horizontal start position of the container. Not configurable by the user.
Starting YPOS Container	Vertical start position of the container. Not configurable by the user.
Height Container	Container's height.

	Not configurable by the user.
Width Container	Container's width. Not configurable by the user.

3. Quick Startup

Important Note: Before generating diagrams (except for the Database diagram), you need to run Clean XREF option. For running this option click on the project's root, go to CrossWay and press Clean XREF.



Clean XREF

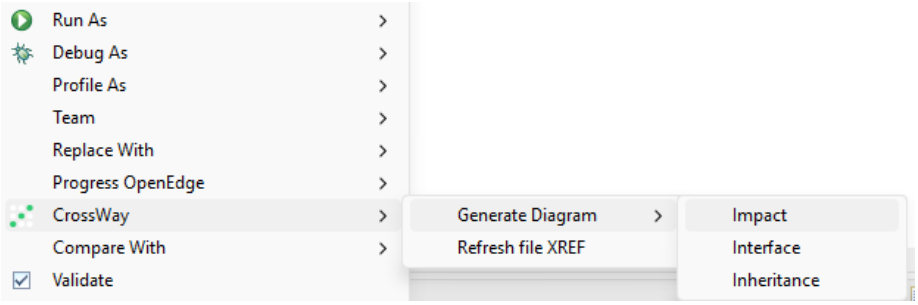
- In OpenEdge, XREF refers to a feature of the compiler that generates a file containing information about how source code elements (like procedures and classes) interact with database objects, indexes, and other program elements.
- XREF provides information about source code relationships, indexes, class and variable references.
- Clean XREF option regenerates the file each time it is pressed to match the current changes. Based on all the information collected by the Clean XREF option, diagrams can be generated.
- Select the project's root and go to CrossWay → Clean XREF.

Generating a Diagram

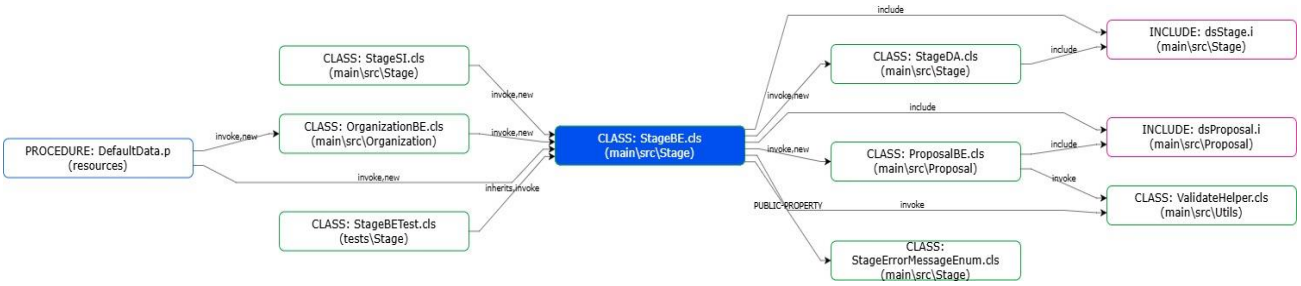
Impact Diagram

- The impact diagram shows the relationships and dependencies between classes, procedure, include files, etc.

- The impact diagram can be displayed by using both draw.io or CrossWay Visualizer.
- Go inside a class that you want to generate a diagram for, right click and press CrossWay → Generate Diagram → Impact.

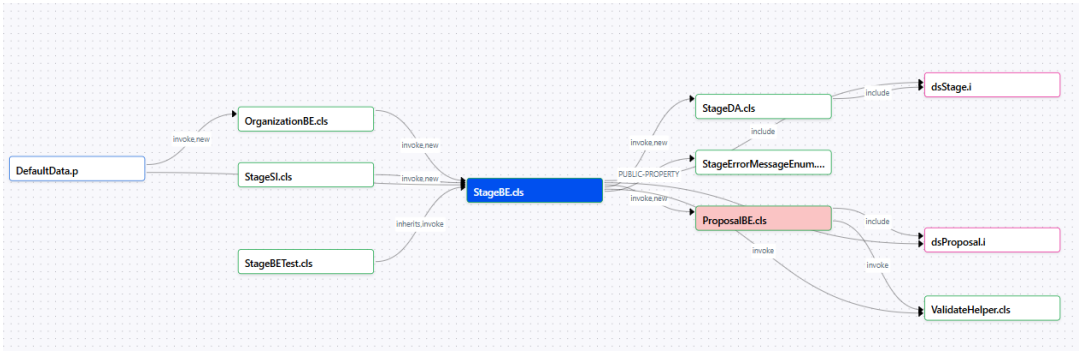


- A simple impact diagram for a class called StageBE is shown below.
    - draw.io Diagram:



- Under draw.io you have an option to arrange a diagram as you want. For this Impact diagram you can use Arrange > Layout >Horizontal Flow

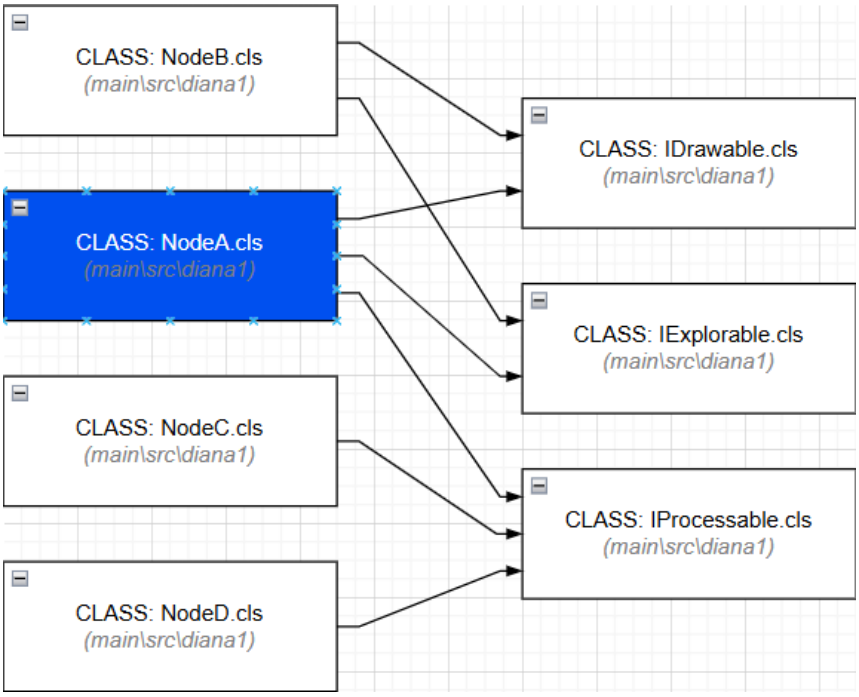
    - CrossWay Visualizer:



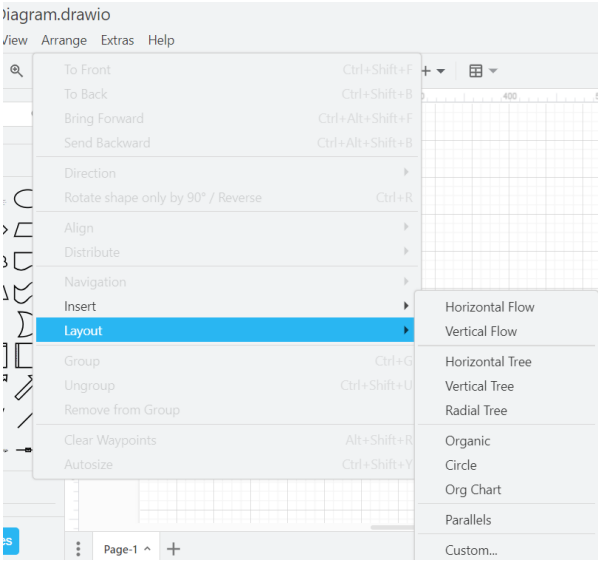
Interface Diagram

- The interface diagram shows the connections between classes and the interfaces they're implementing.
- The interface diagram can be displayed only by using draw.io.

- Go inside a class that you want to generate a diagram for, right click and press CrossWay → Generate Diagram → Interface.



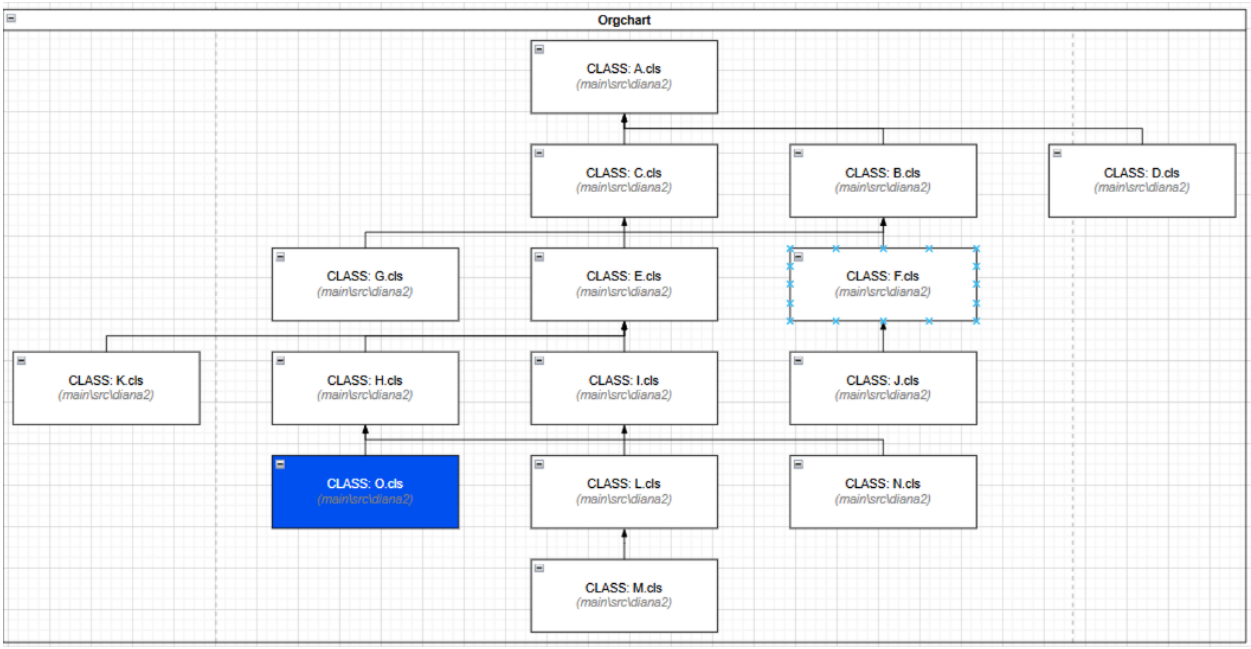
- Under draw.io you have an option to arrange a diagram as you want. For this Interface diagram you can use Arrange > Layout >Vertical Flow



Inheritance Diagram

- The inheritance diagram shows the hierarchy of superclasses and subclasses.
- The inheritance diagram can be displayed only by using draw.io.

- Go inside a class that you want to generate a diagram for, right click and press CrossWay → Generate Diagram → Inheritance.
- An example of an Inheritance Diagram can be seen below.
- *Note*: Here the container from OrgChart Properties can be observed.



4. Extensive Use Cases

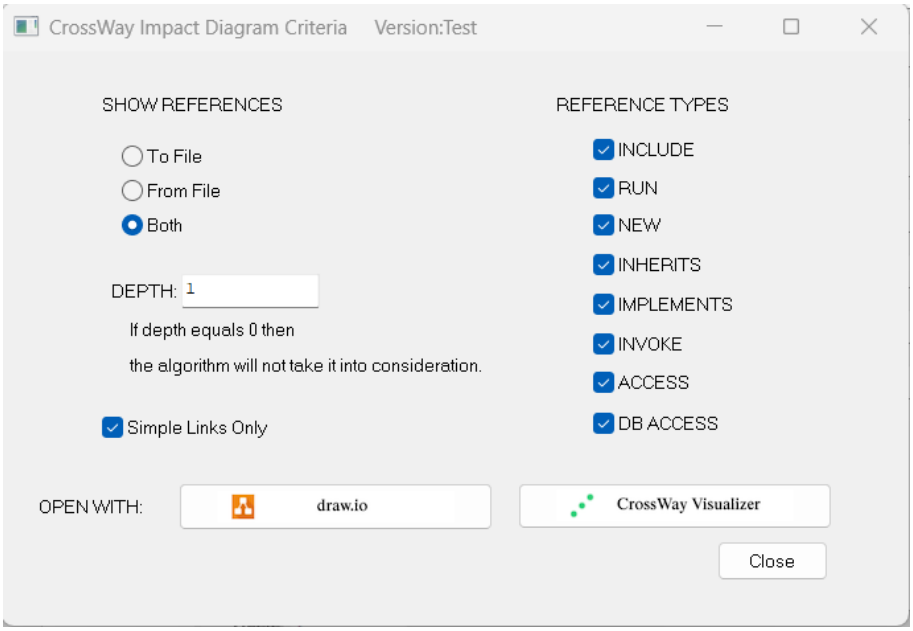
For now, by using CrossWay you can generate three types of diagrams: Impact Diagram, Interface Diagram and Inheritance Diagram. In a future release, it will be possible to generate other diagrams such as Flow Diagram, Package Diagram or Database Relation Diagram.

Impact Diagram

An impact diagram visually represents how changes in one area (the "from" node) can affect another (the "to" node) and it is often used to analyze relationships and dependencies in complex systems.

The impact diagram displays impact links and the corresponding nodes from one root element. For the purpose of this explanation, we will consider the referenced class (root node) to be a class named StageBE.cls. The diagrams will be generated using both draw.io and CrossWay Visualizer.

After pressing right click → CrossWay → Generate Diagram → Impact button, the following screen will pop up. Here you can configure the way the diagram is generated and which dependencies should be displayed.



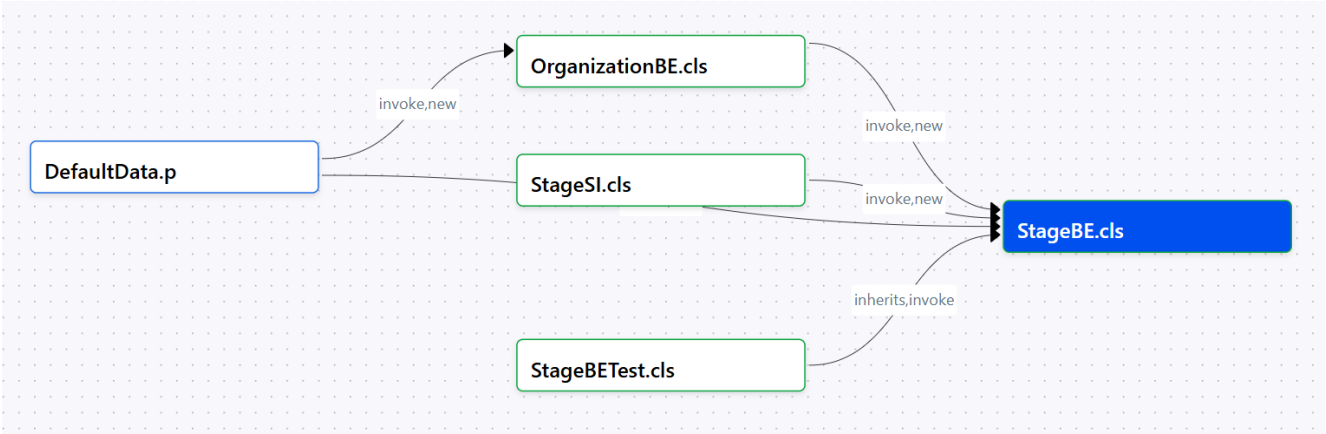
Open with options

- Draw.io
  - Generates the impact diagram using draw.io tool.
- CrossWay Visualizer
  - Generates the impact diagram using CrossWay Visualizer tool.

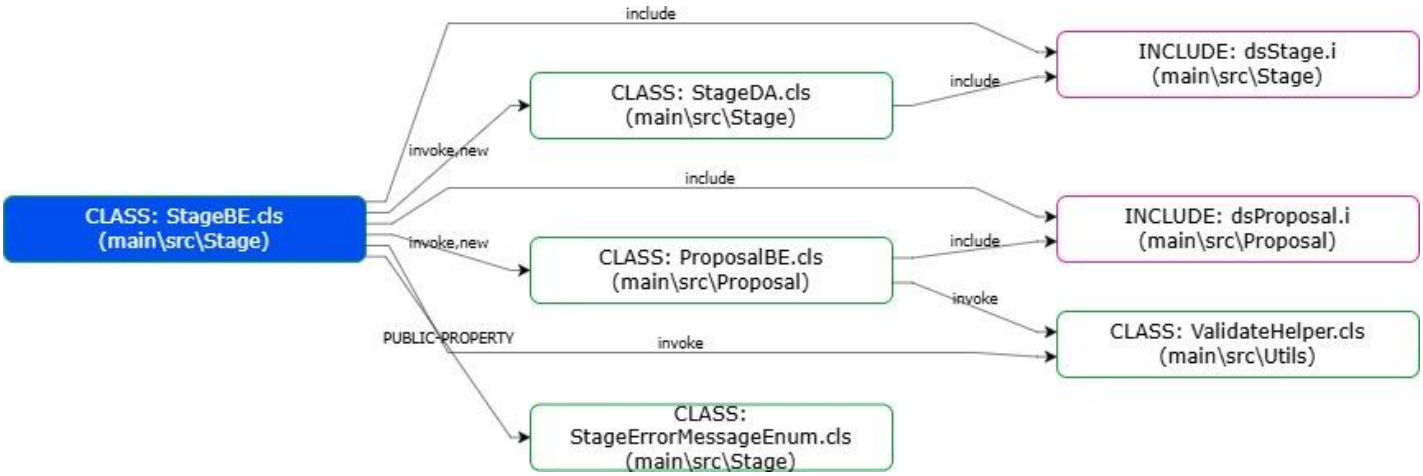
Show References

- To file
  - Performs a backward traversal from the root node, generating an impact diagram that includes all predecessor nodes (i.e., nodes that have a directed path leading to the root node).
  - A simple example for StageBE.cls where StageBE.cls is the root node and the 'To' option is checked.
  - draw.io:

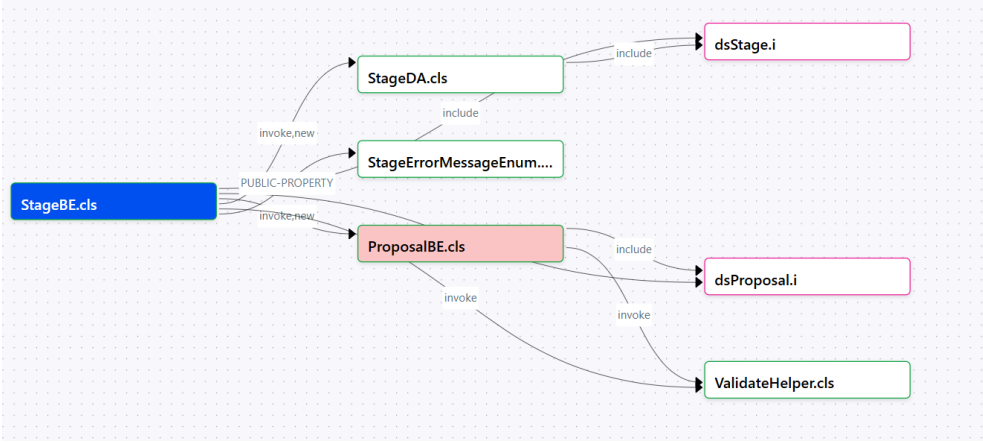
- CrossWay Visualizer:



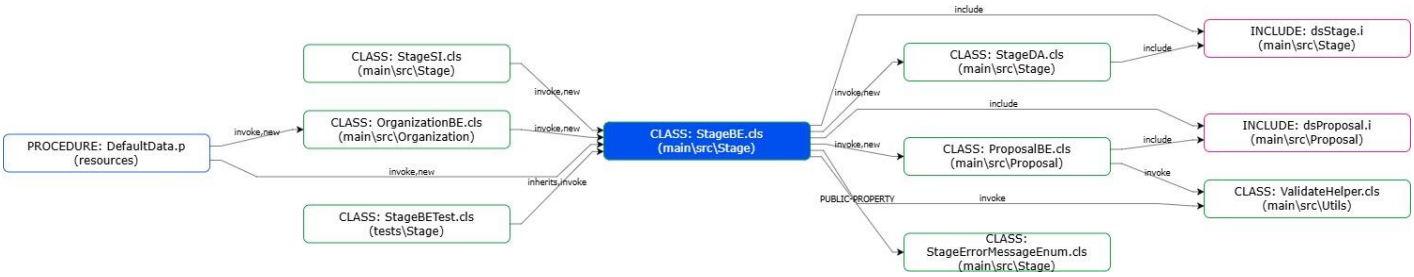
- From file
  - Performs a forward traversal from the root node, generating an impact diagram that includes all successor nodes (i.e., nodes that are reachable from the root node via directed paths).
  - A simple example for StageBE.cls where StageBE.cls is the root node and the 'From' option is checked.
  - draw.io:



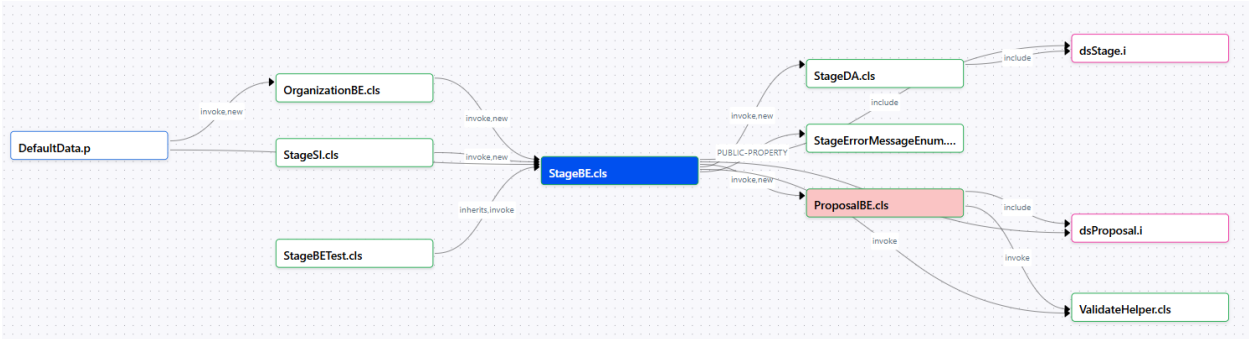
- CrossWay Visualizer:



- Both
  - Performs both a forward and backward transversal from the root node generating an impact diagram that contains all predecessors and all successors of the root node. (i.e., nodes that have a directed path leading to the root node and nodes that are reachable from the root node via directed paths).
  - draw.io:

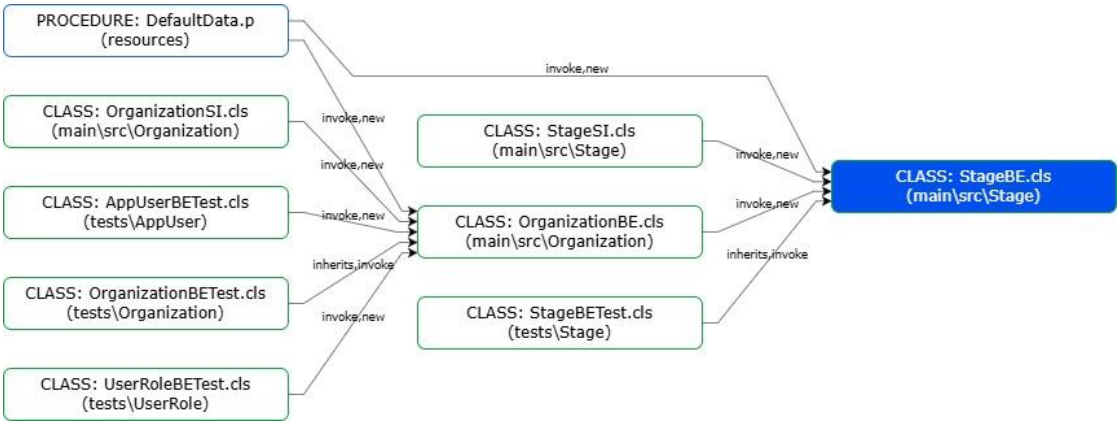


  - CrossWay Visualizer:

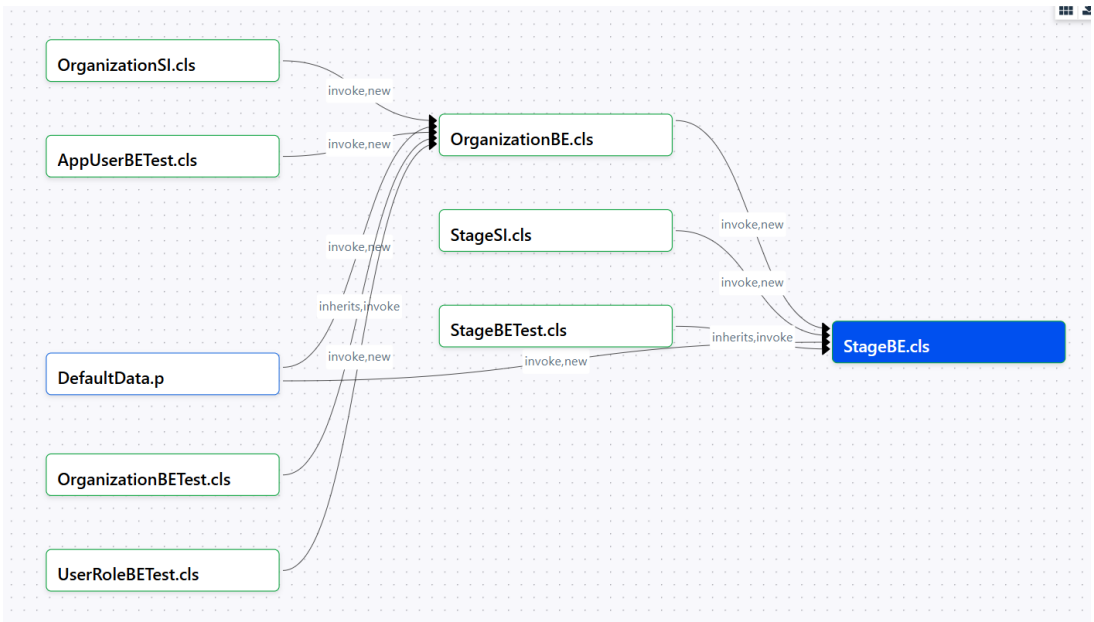


- Depth
  - Refers to the distance, in number of links, between a node and the referenced node based on the direction (‘to’ or ‘from’).

- For example, every node that is connected to the root node via one link, no matter the chosen direction, will have a depth equal to 1 (all the nodes in the image above). Nodes that are connected via 2 links to the main node will have a depth equal to 2. An example for depth equals to 2 and 'To File' can be seen below.
- If the depth is equal to zero, then the diagram will display all the levels that exist.
- draw.io:



- CrossWay Visualizer:

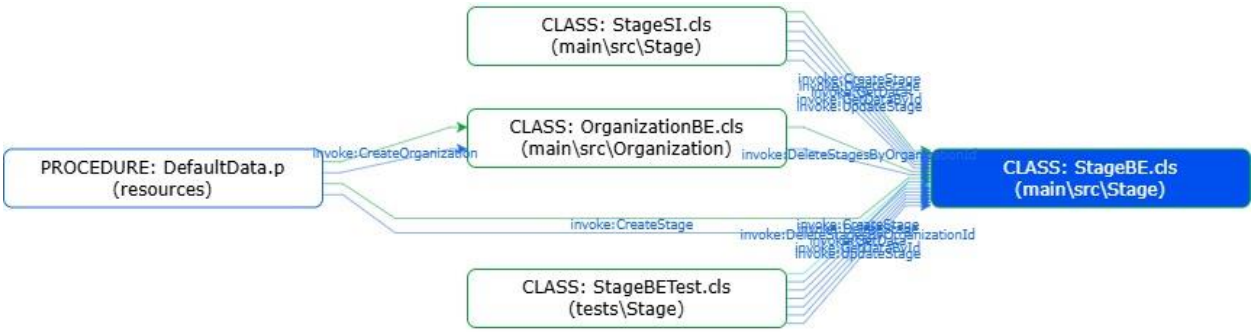


- Between DefaultData.p and StageBE.cls there is a direct link which means the depth is one.

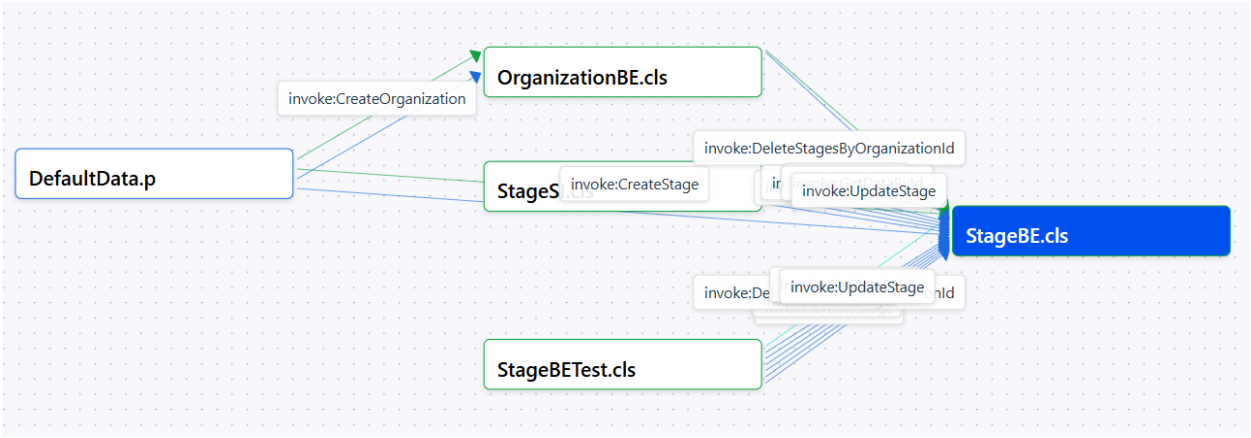
- ⊄ Between AppUserBETest.cls StageBE.cls there are two links meaning the depth is two (AppUserBETest.cls → OrganizationBE.cls → StageBE.cls).

∉ Simple Links Only

- ⊄ Each link displays the relationships between the two nodes it connects (invoke, new, inherits, etc.).
- ⊄ If Simple Links is checked it means that the links inside the diagram will include every relationship on just one line (as seen in the diagram examples above).
- ⊄ If Simple Links is unchecked, it means that each relationship will have its own link.
- ⊄ draw.io:



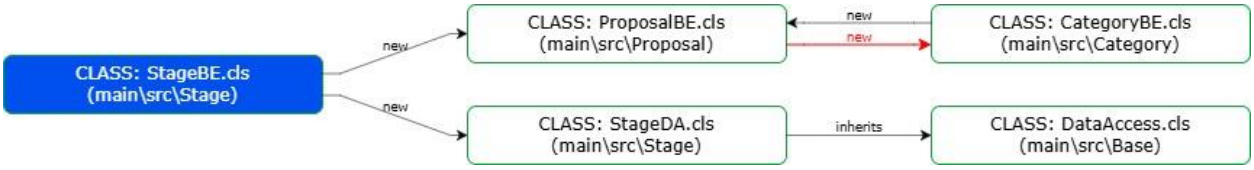
- ⊄ CrossWay Visualizer:



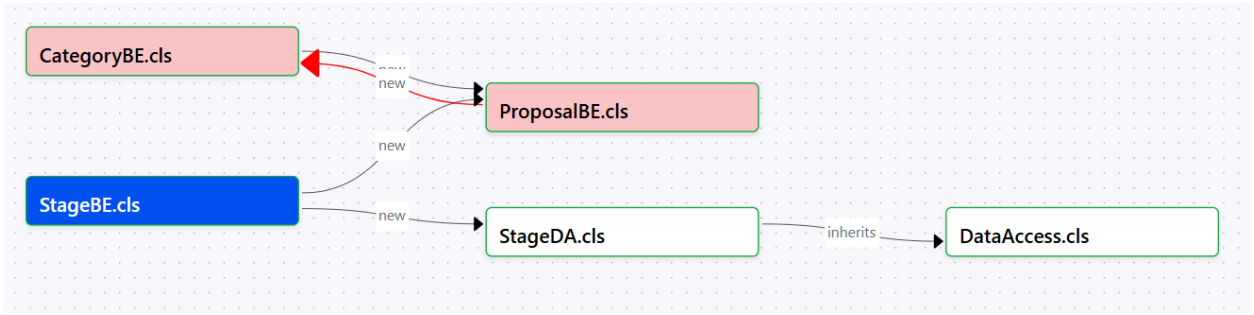
Reference Types

∉ Reference types refer to the relationship types that can exist between two nodes. A class or procedure can include a dataset for example, or can run an external procedure, it can create new instances of another class, or it can inherit another class, etc.

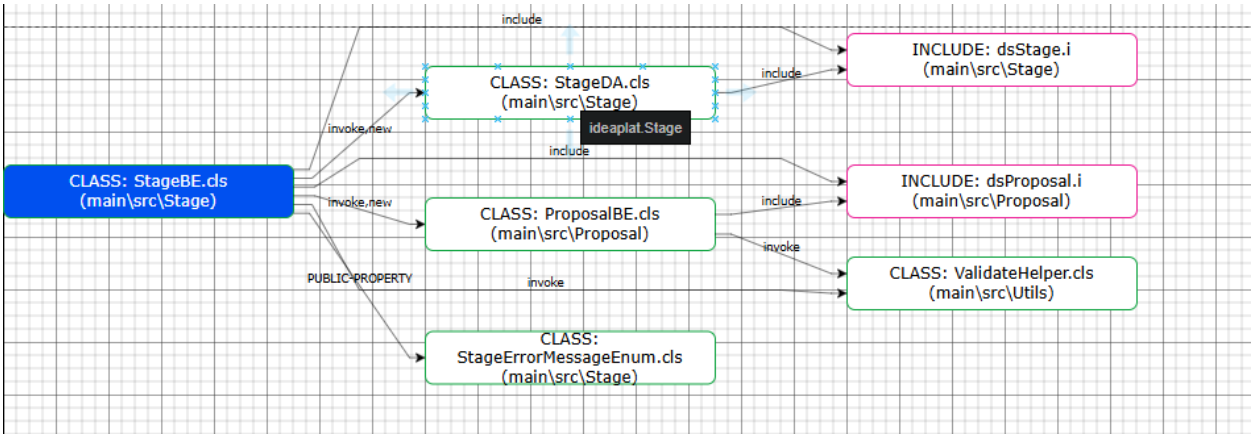
- If the user wants to focus on just some features that can be displayed in the diagram, the unnecessary features can be unchecked. The diagram will display only the links and corresponding nodes that met the checked condition.
- For example, if we select just 'New' and 'Inherits' options for a diagram 'From File' with depth equal to two, the final result will look like this:
    - draw.io:



    - CrossWay Visualizer:



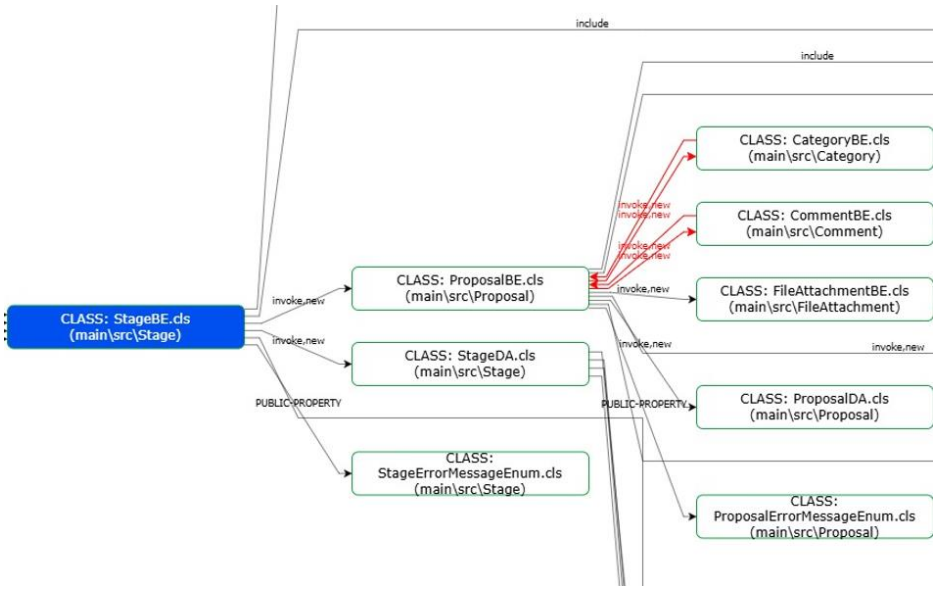
- DB Access reference type will provide the database and table (database.table) that the class is referencing. For seeing the database references, you can hover over the node and all the databases and table used will be displayed.
    - draw.io Diagram:



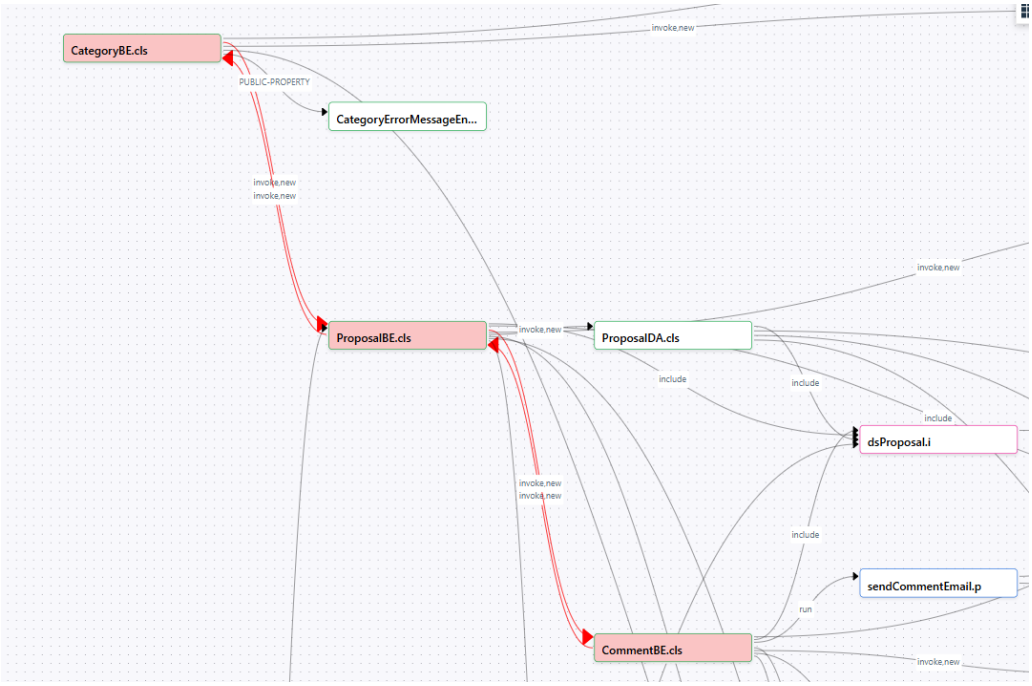
    - CrossWay Visualizer doesn't currently support this functionality.

Circular Dependencies

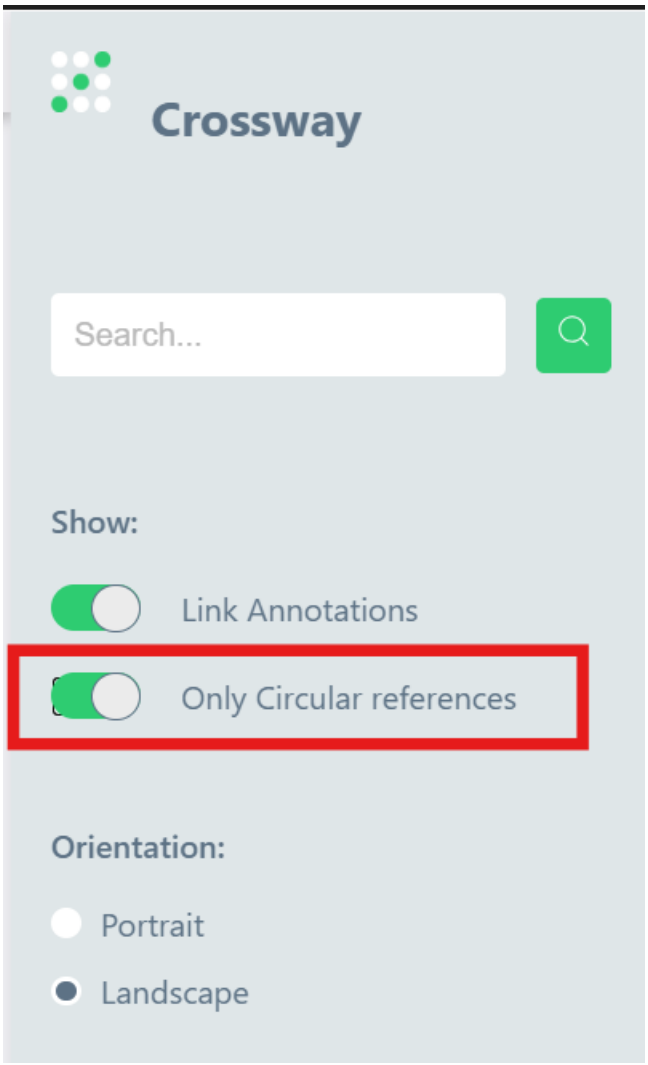
- A circular dependency is represented by two red arrows and it means that the classes reference each other.
- In the above example, circular dependencies can be seen between ProposalBE and CategoryBE and also between ProposalBE and CommentBE.
    - draw.io:

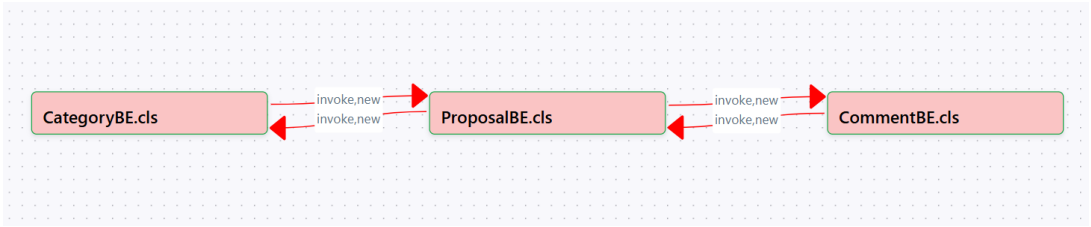


    - CrossWay Visualizer:



- CrossWay Visualizer provides an option to display just the circular dependencies:





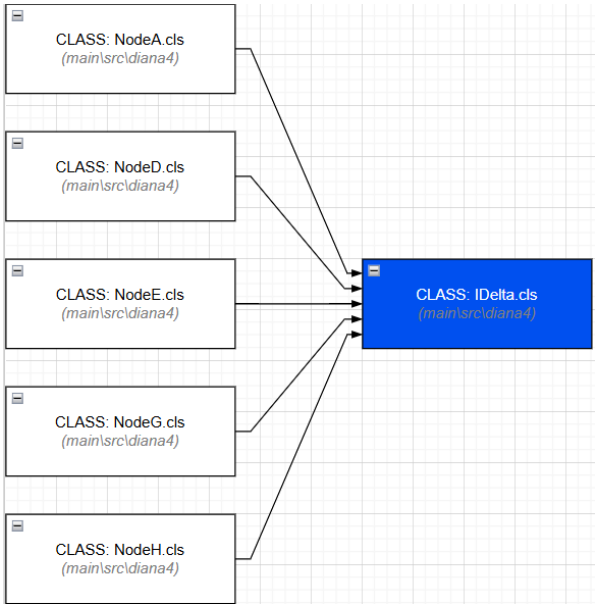
Interface Diagram

An interface diagram is a diagram used to show the relationship between an interface and the classes that implement that interface.

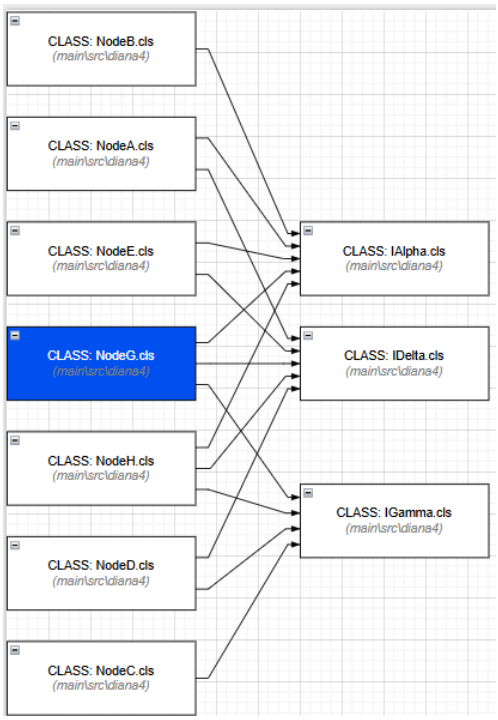
After pressing right click → CrossWay → Generate Diagram → Interface inside a project's class, the diagram will be generated and opened with draw.io.

Some examples of interface diagrams can be seen below for a custom project that includes multiple OpenEdge interfaces and classes that implement those interfaces.

- Example 1



- Example 2

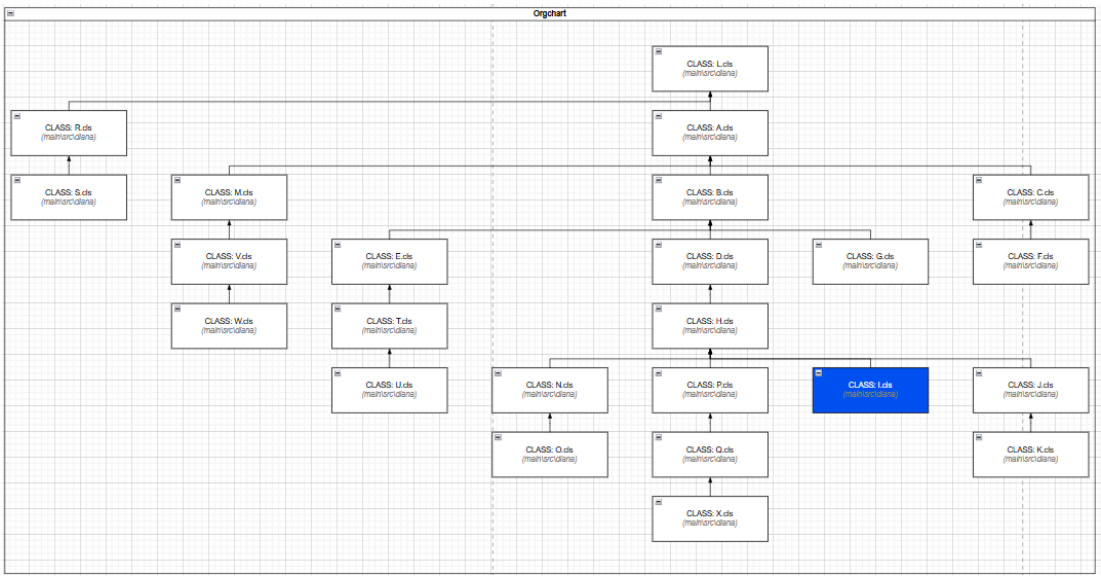


Inheritance Diagram

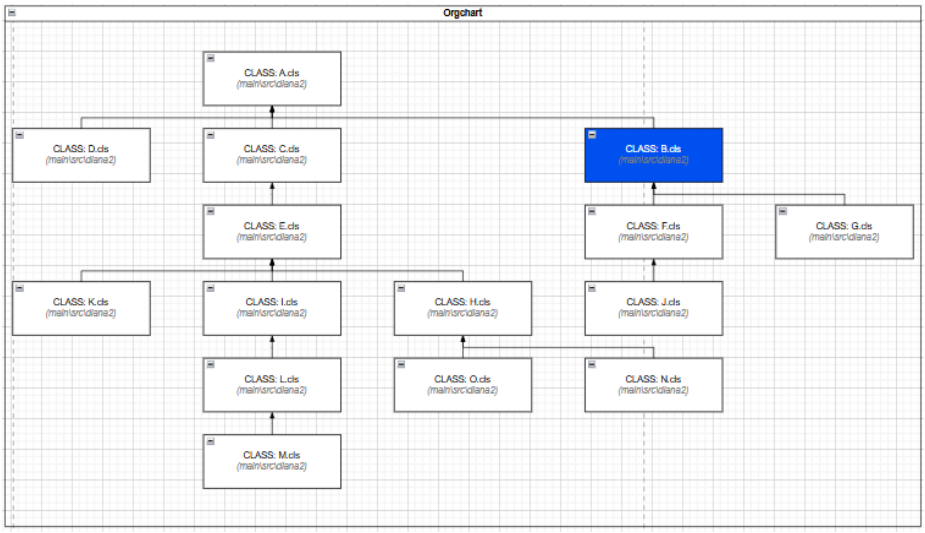
An inheritance diagram visually represents the hierarchical relationships between classes in object-oriented programming. It shows how classes derive attributes and behaviors from one another, illustrating the flow of properties and methods from parent (super) classes to child (sub) classes.

After pressing right click → CrossWay → Generate Diagram → Inheritance inside a project's class, draw.io will open and display the diagram. To present this diagram type, we will use a custom project with multiple classes that include inheritance.

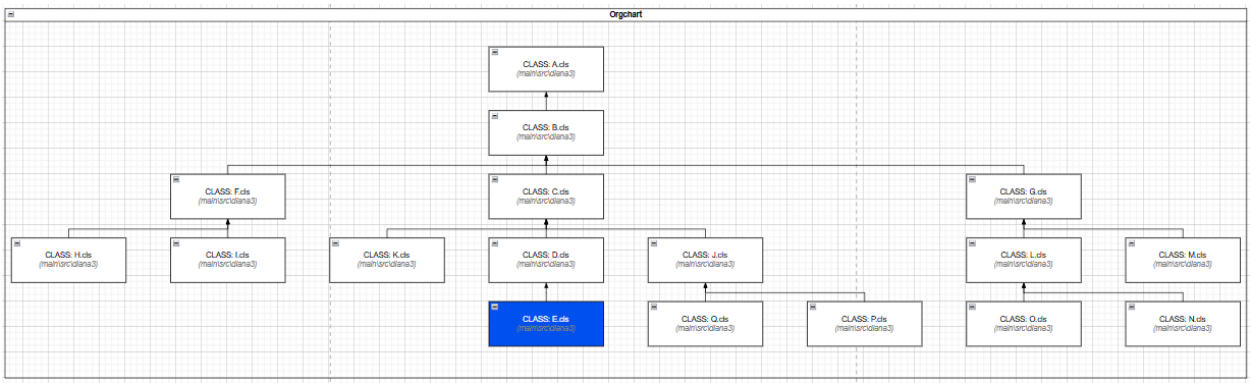
- Example 1:



- Example 2:



- Example 3:



5. Best Practices

- Whenever multiple changes occur over several files, it is not always necessary to generate the diagram for each file for the changes to be taken into consideration. Instead, use right click → CrossWay → Refresh XREF option on the ones which don't need a diagram to be generated for, and only use generate diagram options on the ones that it is needed for.
- After generating the first diagram, if you don't close the diagram, you need to Restart CrossWay AVM by going to CrossWay → Restart CrossWay AVM to can generate another one.
- If more diagrams for the same file are needed, after generating the first diagram, you need to rename it before generating the second one.
- For the Impact Diagram using draw.io there is support for automatic arrangement of nodes within the diagram: draw.io → Layout → Arrange → Horizontal Flow.

6. Troubleshooting & FAQ

Common errors and how to resolve them

  - Please see the common errors and questions about CrossWay on FAQ document, here.

7. Contact & Feedback

- Company contact info (TBA by Oana)
- Suggestions and feature request form