CROSSWAY USER MANUAL

1. Introduction

What is Crossway?

- Crossway is a 4GL built code analysis tool tailor-made for Progress OpenEdge applications. It helps users visualize and understand the technical and functional ripple effects of code changes. (TBA by Oana)

Who Should Use It?

- Might want to use the website info as a basis and extend/detail it further

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 executable or installed. need a link to the executable and to the installer.

Installation Steps:

- Follow link to Plugin Installation PDF on wayfare.ro/crossway.

Permissions Required:

- Write privileges for output folder of Crossway artefacts (e.g., C:\CrosswayGen).

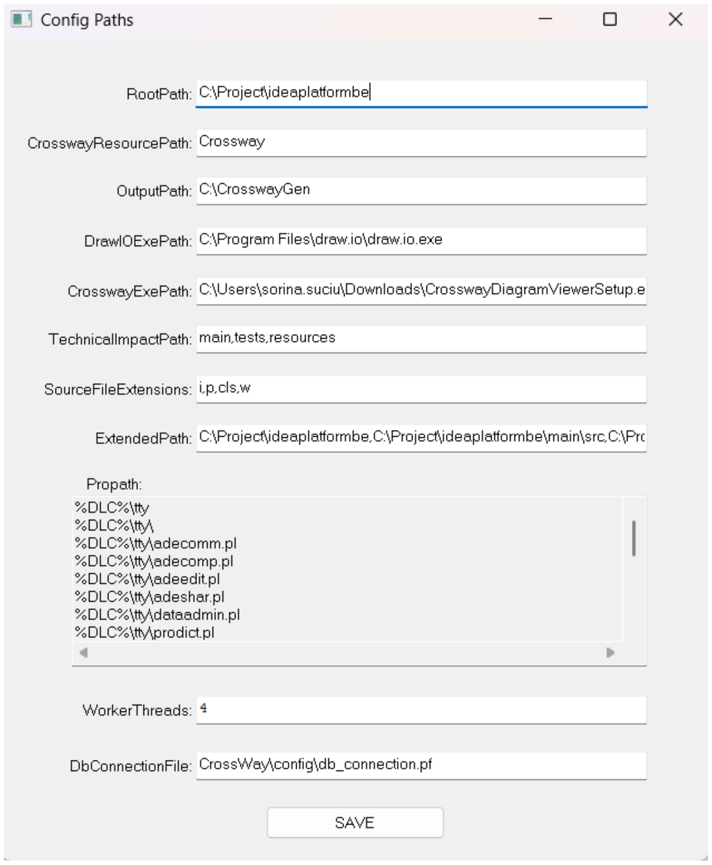
Configuration:

- After the Crossway Plugin was successfully installed in Developer Studio, the next step is ensuring all Crossway settings are properly configured for the workspace.

- First an initialization needs to be performed on the projects you want to use Crossway for (need to show the menu option screenshot) - this initialization step makes copies of all resource files Crossway needs into a .crossway folder directly under the project's @ROOT path (screenshot)
- There are 2 separate configuration files located under the project's .crossway/config folder. These are the local repositories for the Crossway general configuration and UI configuration screens that can be accessed from the Crossway menu → ConfigPaths/Config draw.io.

Config Paths

- Config Path 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.



RootPath:	C:\Project\ideaplatformbe
CrosswayResourcePath:	Crossway
OutputPath:	C:\CrosswayGen
DrawIOExePath:	C:\Program Files\draw.io\draw.io.exe
CrosswayExePath:	C:\Users\sorina.suciu\Downloads\CrosswayDiagramViewerSetup.e
TechnicalImpactPath:	main,tests,resources
SourceFileExtensions:	i,p,cls,w
ExtendedPath:	C:\Project\ideaplatformbe,C:\Project\ideaplatformbe\main\src,C:\Pro

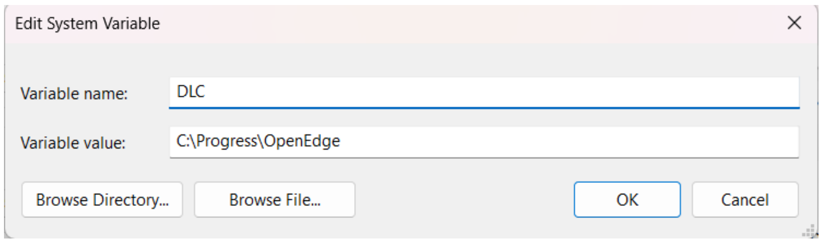
Propath:

```
%DLC%\tty
%DLC%\tty\
%DLC%\tty\adecomm.pl
%DLC%\tty\adecomp.pl
%DLC%\tty\adeedit.pl
%DLC%\tty\adeshar.pl
%DLC%\tty\dataadmin.pl
%DLC%\tty\prodict.pl
```

WorkerThreads:	4
DbConnectionFile:	CrossWay\config\db_connection.pf

- Root Path:
    - Absolute path to the OpenEdge workspace or directly to the project root folder.
- Crossway Resource Path:

    - Relative path to the Crossway folder. In this example, inside IdeaPlatformBE, there is a folder called Crossway.
- Output Path:
    - Absolute path to a folder where CrossWay files are generated. In this example the folder where Crossway will generate the output is called CrosswayGen.
    - *Note:* Make sure that you have permission to write files in the chosen folder.
- draw.ioExe Path:
    - Absolute path to the draw.io.exe file.
    - The draw.io.exe file cand be downloaded and installed from here.
- CrosswayExe Path:
    - Absolute path to the Crossway Diagram Viewer.
    - Crossway Diagram Viewer se descarca de unde? vine odata cu pluginul?
- Technical Impact Path:
    - A comma-separated list of workspace-relative folder paths of the files Crossway should process (e.g., src folders of multiple projects in the same workspace).
    - The project in this example contains a ‘main’ folder where all the methods are implemented, a ‘tests’ folder for the unit tests and a resource folder.
- 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, C:\Project\ideaplatformbe\resources
- 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.



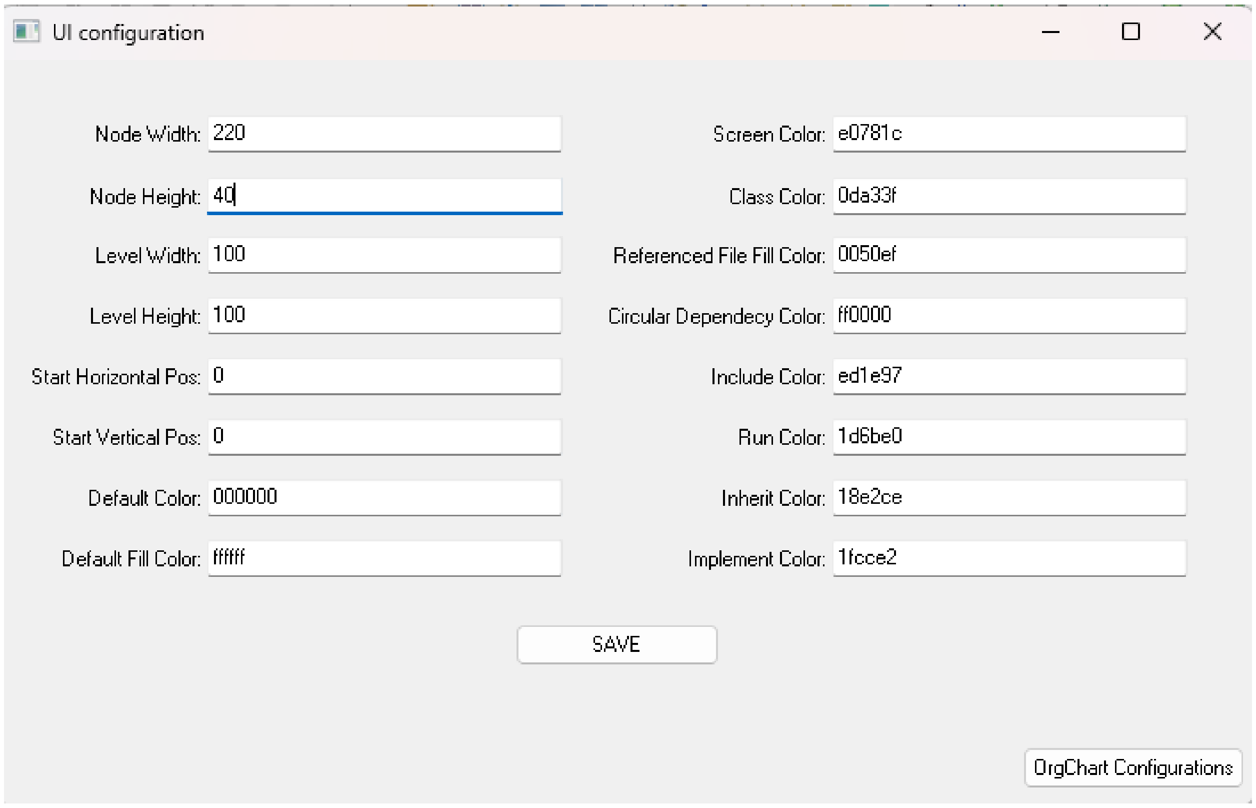
Variable name: DLC

Variable value: C:\Progress\OpenEdge

- 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.

Config draw.io

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



Node Width: 220

Node Height: 40

Level Width: 100

Level Height: 100

Start Horizontal Pos: 0

Start Vertical Pos: 0

Default Color: 000000

Default Fill Color: ffffff

Screen Color: e0781c

Class Color: 0da33f

Referenced File Fill Color: 0050ef

Circular Dependecy Color: ff0000

Include Color: ed1e97

Run Color: 1d6be0

Inherit Color: 18e2ce

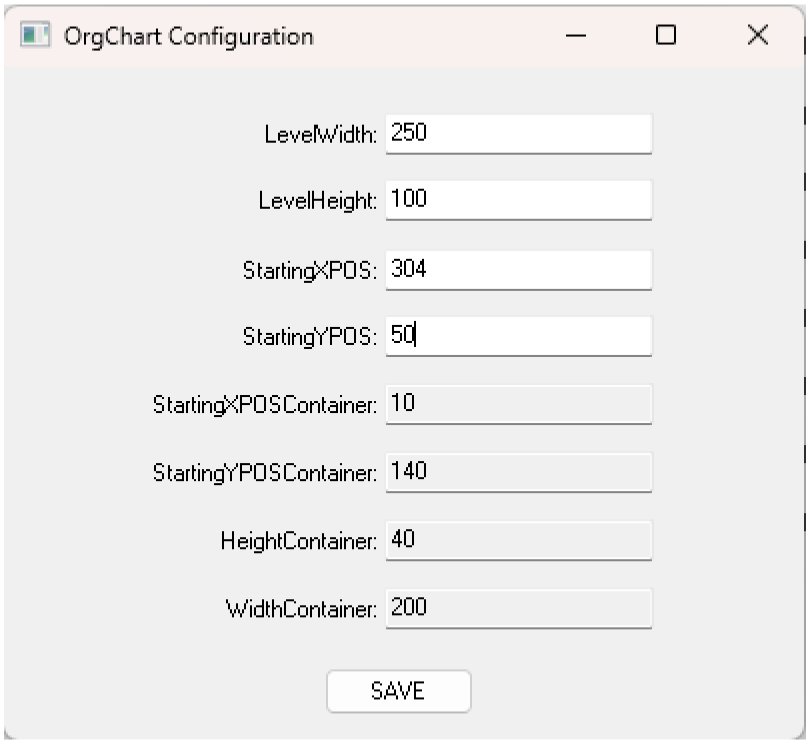
Implement Color: 1fcce2

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 color for nodes and links.
Default Fill Color	Default 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.

Config OrgChartConfigurations

- Besides the usual configuration, the user can also set the OrgChart parameters for the container around the Inherits 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.



LevelWidth:	250
LevelHeight:	100
StartingXPOS:	304
StartingYPOS:	50
StartingXPOSContainer:	10
StartingYPOSContainer:	140
HeightContainer:	40
WidthContainer:	200

De inlocuit imaginea dupa ce se schimba labels

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

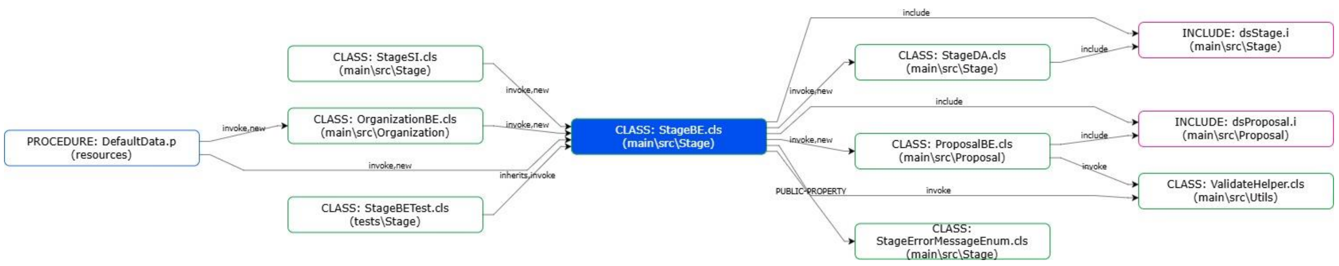
- In OpenEdge, XREF refers to a feature of the compiler that generates a file containing information about how source code elements (like procedures and classes) interact with database objects, indexes, and other program elements.
- XREF provides information about source code relationships, indexes, 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.
- After the XREF file is up to date, the next step is to right click on the project's root and go to Progress OpenEdge → Restart OpenEdge AVM.

Generating a Diagram

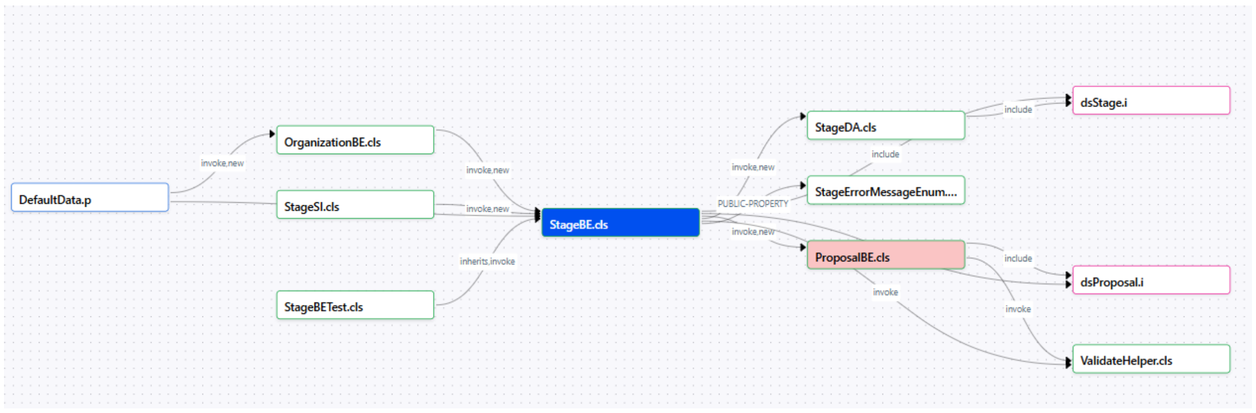
Impact Diagram

- 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 Diagram Viewer.

- Go inside a class that you want to generate a diagram for, right click and press Crossway → Generate 'Impact' Diagram/Generate Diagram.
- A simple impact diagram for a class called StageBE is shown below.
    - draw.io Diagram:

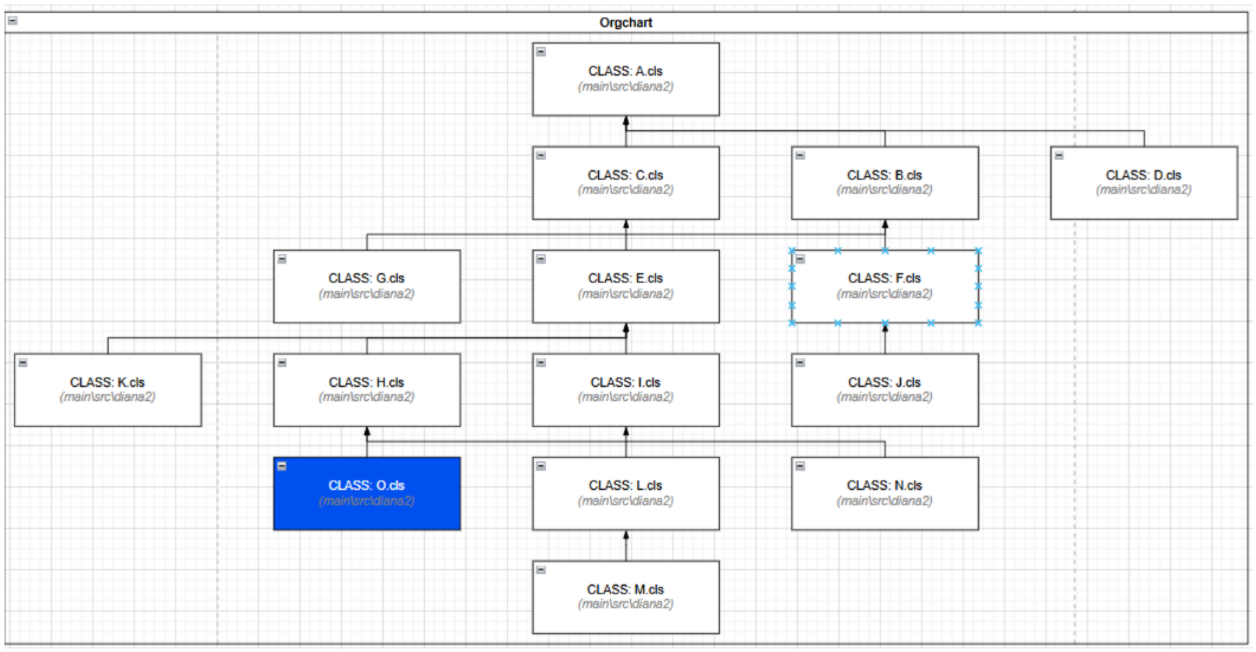

PROCEDURE: DefaultData.p (resources)
CLASS: StageSI.cls (main\src\Stage)
CLASS: OrganizationBE.cls (main\src\Organization)
CLASS: StageBETest.cls (tests\Stage)
CLASS: StageBE.cls (main\src\Stage)
CLASS: StageDA.cls (main\src\Stage)
CLASS: ProposalBE.cls (main\src\Proposal)
CLASS: StageErrorMessageEnum.cls (main\src\Stage)
INCLUDE: dsStage.i (main\src\Stage)
INCLUDE: dsProposal.i (main\src\Proposal)
CLASS: ValidateHelper.cls (main\src\Utils)
invoke.new
inherits,invoke
include
invoke
PUBLIC-PROPERTY

    - Crossway Diagram Viewer:



Inherits Diagram

- The inherits diagram shows the hierarchy of superclasses and subclasses.
- The inherits 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 'Inherits' Diagram.
- An example of an Inherits Diagram can be seen below.
- *Note*: Here the container from OrgChart Properties can be observed.



Ce alte diagrame mai sunt?

4. Extensive Use Cases

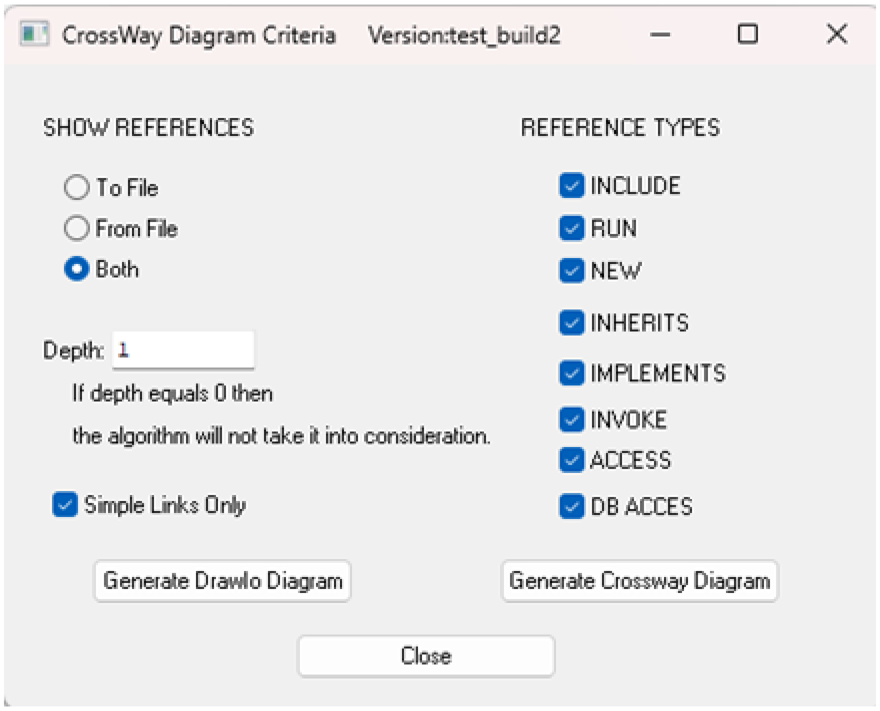
For now, by using Crossway you can generate three types of diagrams: Impact Diagram, Implements Diagram and Inherits Diagram. In the future, there will be possible to generate other diagrams such as Flow Diagram, Package Diagram or Database Relation Diagram.

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 Diagram Viewer.

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


CrossWay Diagram Criteria Version:test_build2

SHOW REFERENCES

- ○ To File
- ○ From File
- ● Both

Depth: 1

If depth equals 0 then the algorithm will not take it into consideration.

☑ Simple Links Only

REFERENCE TYPES

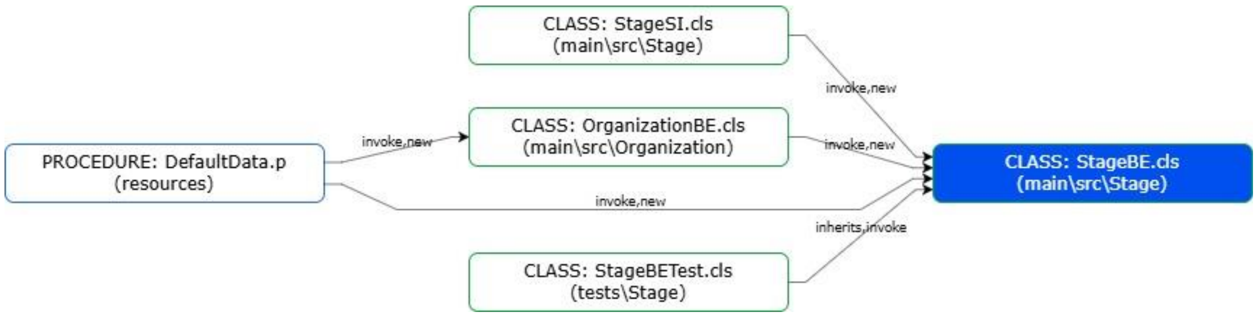
- ☑ INCLUDE
- ☑ RUN
- ☑ NEW
- ☑ INHERITS
- ☑ IMPLEMENTS
- ☑ INVOKE
- ☑ ACCESS
- ☑ DB ACCES

Buttons

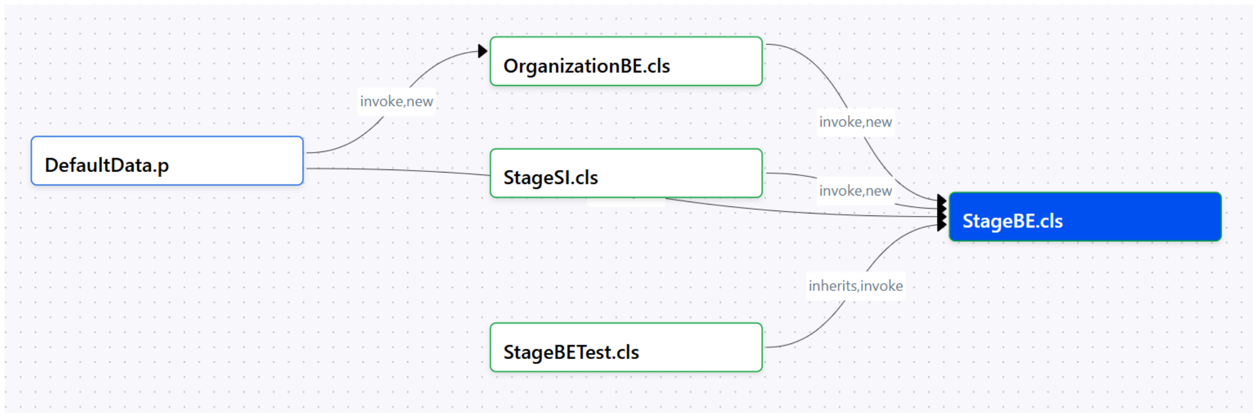
- Generate draw.io Diagram
  - Generates the impact diagram using draw.io tool.
- Generate Crossway Diagram
  - Generates the impact diagram using Crossway Diagram Viewer tool.
  - Note: The button will open Crossway Diagram Viewer. From here, to see the diagram, you have to click on ‘Add new’ button → Upload from device → go inside CrosswayGen folder → drag and drop the desired file. In this case, the file will be called StageBE.cls.json.

Show References

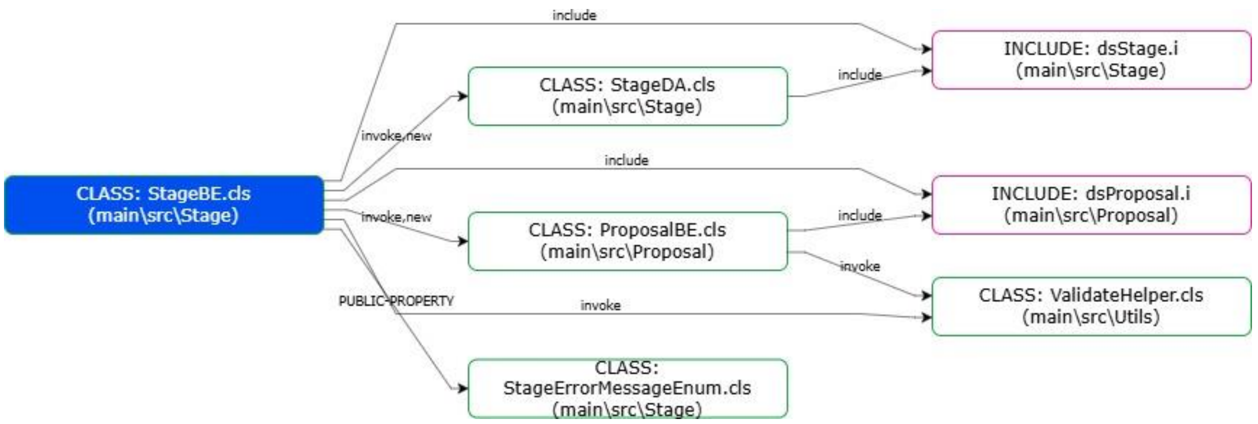
- To file
  - Performs a backward traversal from the root node, generating an impact diagram that includes all predecessor nodes (i.e., nodes that have a directed path leading to the root node).
  - A simple example for StageBE.cls where StageBE.cls is the root node and the ‘To’ option is checked.
 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


CLASS: StageSI.cls (main\src\Stage)
CLASS: OrganizationBE.cls (main\src\Organization)
PROCEDURE: DefaultData.p (resources)
CLASS: StageBE.cls (main\src\Stage)
CLASS: StageBETest.cls (tests\Stage)
invoke,new
invoke,new
invoke,new
invoke,new
inherits,invoke

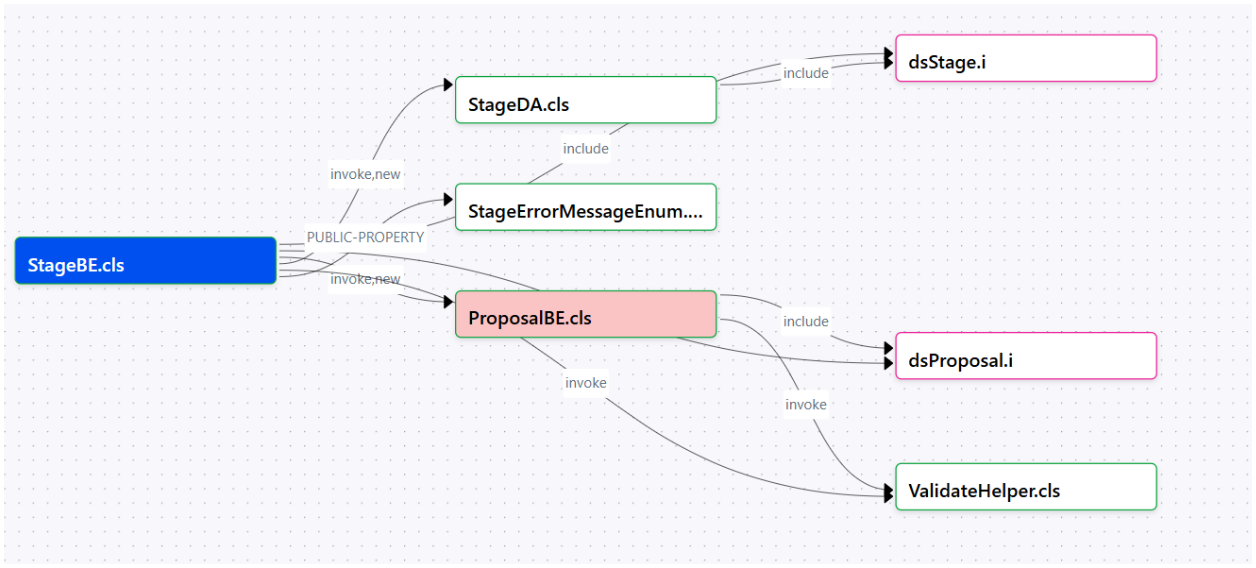
- Crossway Diagram Viewer:



- 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 Diagram:


include
INCLUDE: dsStage.i (main\src\Stage)
CLASS: StageDA.cls (main\src\Stage)
include
invoke,new
include
CLASS: StageBE.cls (main\src\Stage)
INCLUDE: dsProposal.i (main\src\Proposal)
invoke,new
CLASS: ProposalBE.cls (main\src\Proposal)
include
invoke
CLASS: ValidateHelper.cls (main\src\Utils)
PUBLIC-PROPERTY
invoke
CLASS: StageErrorMessageEnum.cls (main\src\Stage)

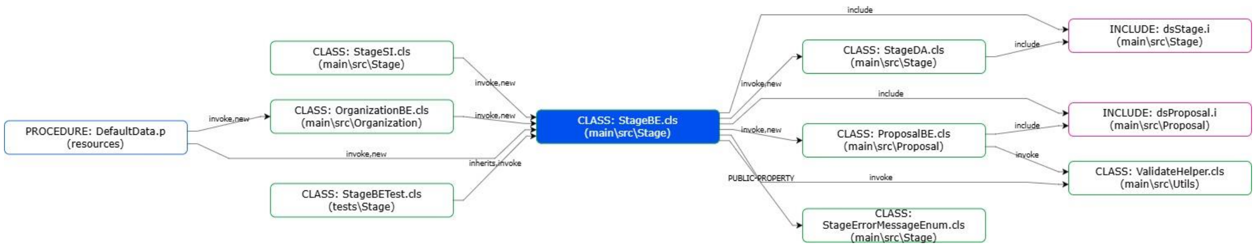
  - Crossway Diagram Viewer:



∉ Both

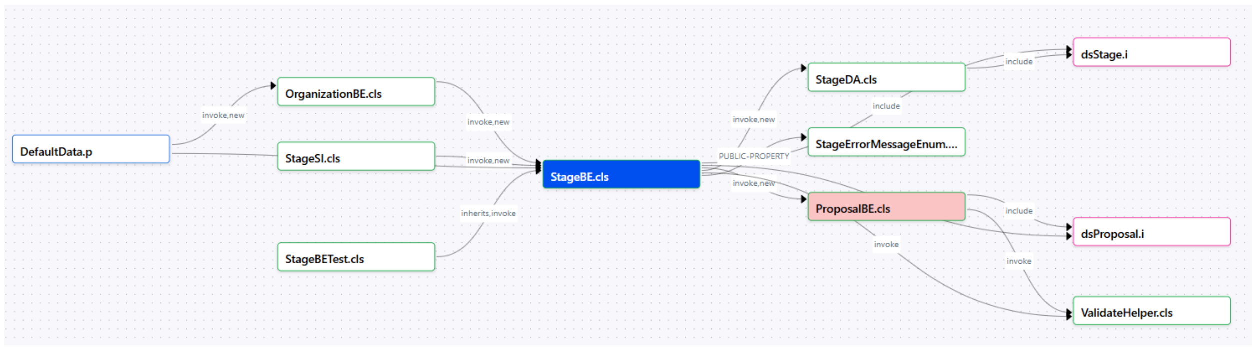
⊄ Performs both a forward and backward transversal from the root node generating an impact diagram that contains all predecessors and all successors of the root node. (i.e., nodes that have a directed path leading to the root node and nodes that are reachable from the root node via directed paths).

⊄ draw.io Diagram:



PROCEDURE: DefaultData.p (resources)
CLASS: StageSI.cls (main\src\Stage)
CLASS: OrganizationBE.cls (main\src\Organization)
CLASS: StageBETest.cls (tests\Stage)
CLASS: StageBE.cls (main\src\Stage)
CLASS: StageDA.cls (main\src\Stage)
CLASS: ProposalBE.cls (main\src\Proposal)
CLASS: StageErrorMessageEnum.cls (main\src\Stage)
INCLUDE: dsStage.i (main\src\Stage)
INCLUDE: dsProposal.i (main\src\Proposal)
CLASS: ValidateHelper.cls (main\src\Utils)
invoke,new
invoke,new
invoke,new
invoke,new
inherits,invoke
include
invoke,new
include
include
invoke,new
include
invoke
PUBLIC-PROPERTY
invoke

⊄ Crossway Diagram Viewer:



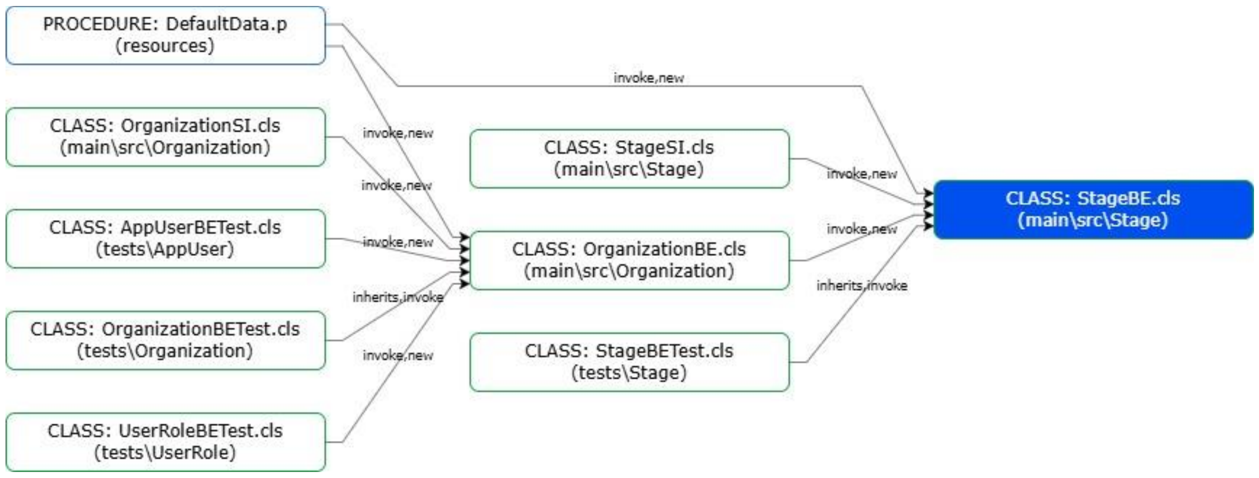
∉ Depth

⊄ Refers to the distance, in number of links, between a node and the referenced node based on the direction ('to' or 'from').

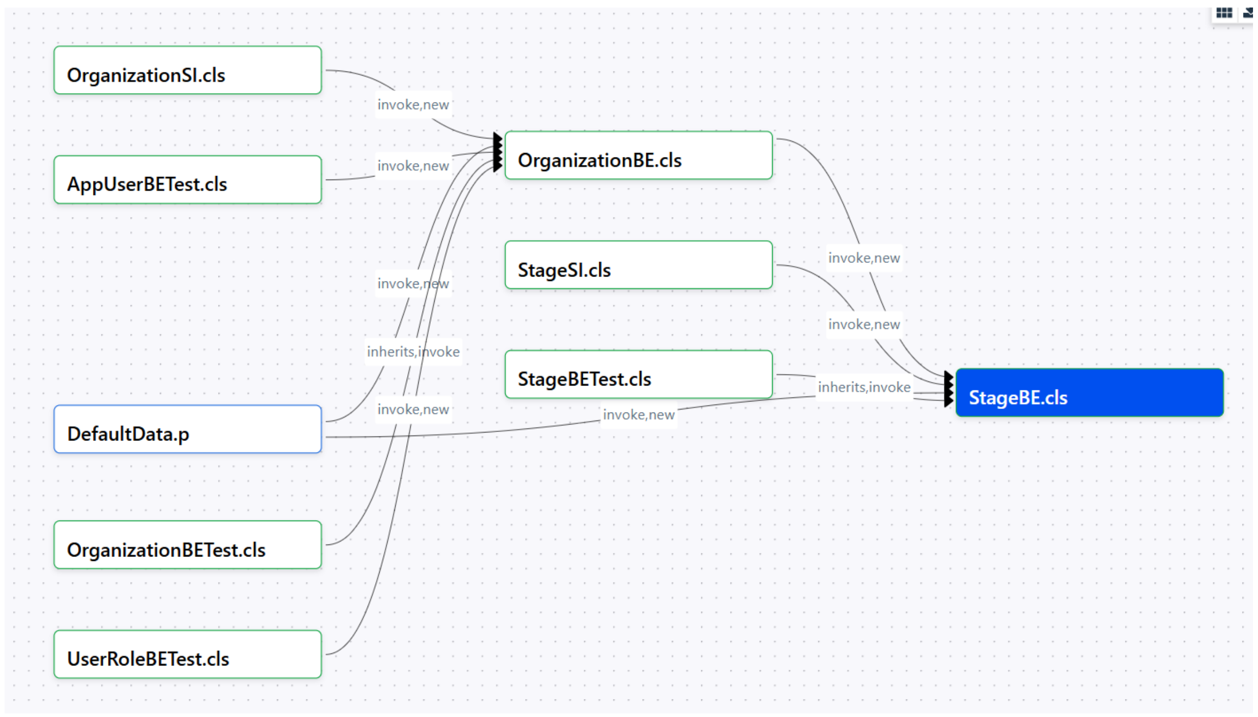
⊄ For example, every node that is connected to the root node via one link, no matter the chosen direction, will have a depth equal to 1 (all the nodes in the image above). Nodes that are connected via 2 links to the main node will have a depth equal to 2. An example for depth equals to 2 and 'To File'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

⊄ If the depth is equal to zero, then the diagram will display all the levels that exist.

⊄ draw.io Diagram:


PROCEDURE: DefaultData.p (resources)
CLASS: OrganizationSI.cls (main\src\Organization)
CLASS: AppUserBETest.cls (tests\AppUser)
CLASS: OrganizationBETest.cls (tests\Organization)
CLASS: UserRoleBETest.cls (tests\UserRole)
CLASS: StageSI.cls (main\src\Stage)
CLASS: OrganizationBE.cls (main\src\Organization)
CLASS: StageBETest.cls (tests\Stage)
CLASS: StageBE.cls (main\src\Stage)
invoke,new
invoke,new
invoke,new
invoke,new
inherits,invoke
invoke,new
invoke,new
invoke,new
inherits,invoke

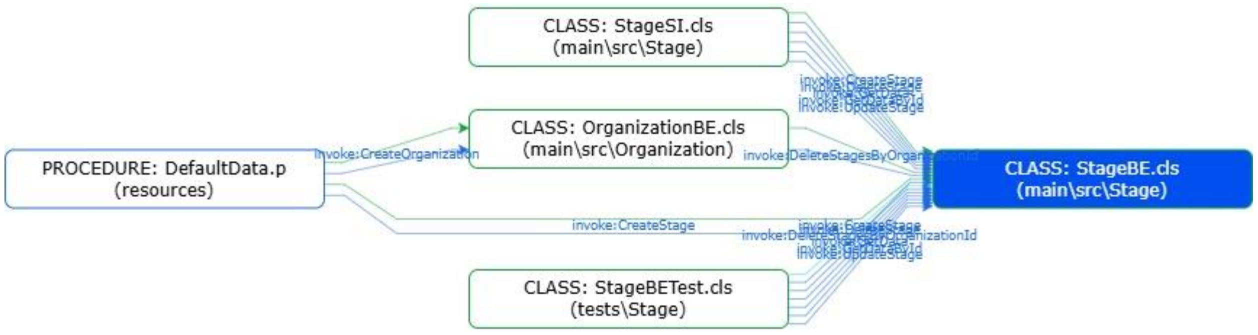
⊄ Crossway Diagram Viewer:



- ⊄ Between DefaultData.p and StageBE.cls there is a direct link which means the depth is 1.
- ⊄ Between AppUserBETest.cls StageBE.cls there are 2 links meaning the depth is 2 (AppUserBETest.cls → OrganizationBE.cls → StageBE.cls).

∉ Simple Links Only

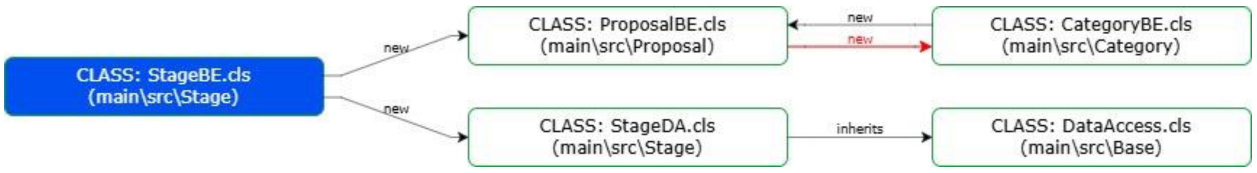
- ⊄ Each link displays the relationships between the two nodes it connects (invoke, new, inherits, etc.).
- ⊄ If Simple Links is checked it means that the links inside the diagram will include every relationship on just one line (as seen in the diagram examples above).
- ⊄ If Simple Links is unchecked,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


CLASS: StageSI.cls (main\src\Stage)
CLASS: OrganizationBE.cls (main\src\Organization)
PROCEDURE: DefaultData.p (resources)
CLASS: StageBETest.cls (tests\Stage)
CLASS: StageBE.cls (main\src\Stage)
invoke:CreateStage
invoke:UpdateStage
invoke:CreateOrganization
invoke:DeleteStagesByOrganizationId
invoke:CreateStage
invoke:DeleteStagesByOrganizationId
invoke:CreateStage
invoke:UpdateStage

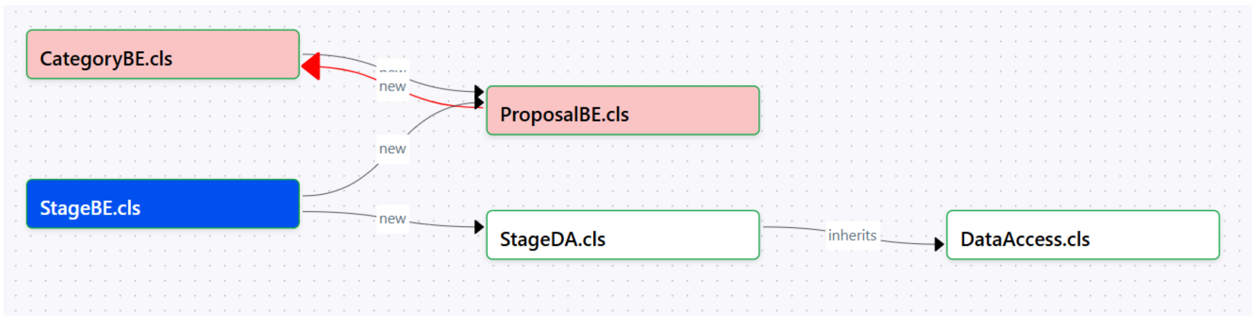
    - Crossway Diagram Viewer: se aplica si pentru el? Ca pare ca nu

Reference Types

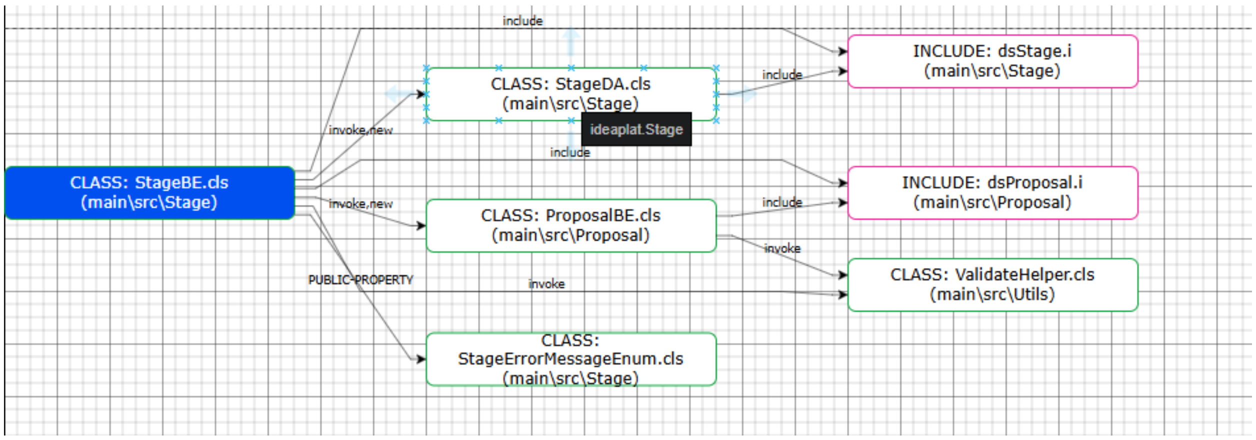
- Reference types refer to the relationship types that can exist between two nodes. A class or procedure can include a dataset for example, or can run an external procedure, it can create new instances of another class, or it can inherit another class, etc.
- If the user wants to focus on just some features 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 Diagram:


CLASS: StageBE.cls (main\src\Stage)
new
CLASS: ProposalBE.cls (main\src\Proposal)
new
new
CLASS: CategoryBE.cls (main\src\Category)
new
CLASS: StageDA.cls (main\src\Stage)
inherits
CLASS: DataAccess.cls (main\src\Base)

    - Crossway Diagram Viewer:



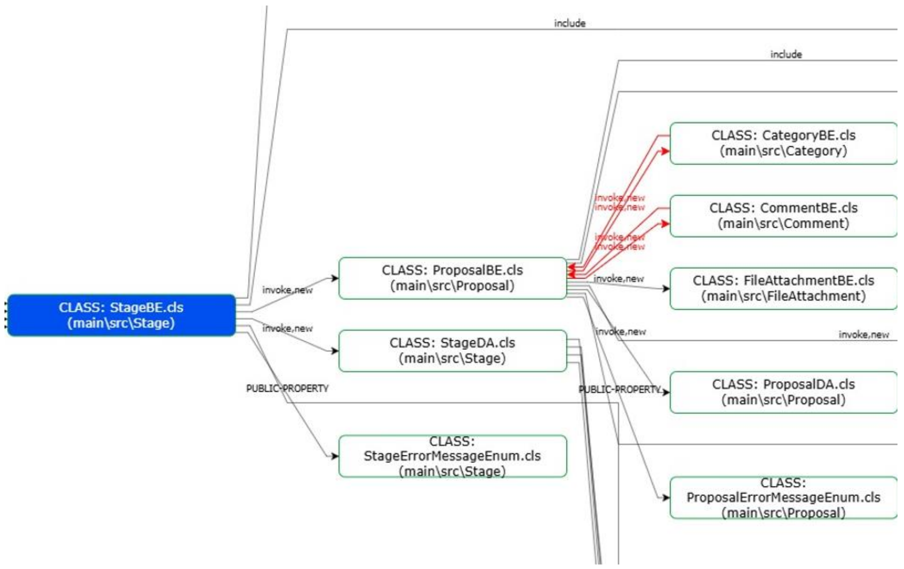
- DB Access reference type will provide the database and table (database.table) that the class is referencing. For seeing the database references, you can hover over the node and all the databases and table used will be displayed.
    - draw.io Diagram:


include
include
CLASS: StageDA.cls (main\src\Stage)
ideaplat.Stage
INCLUDE: dsStage.i (main\src\Stage)
invoke,new
include
CLASS: StageBE.cls (main\src\Stage)
INCLUDE: dsProposal.i (main\src\Proposal)
invoke,new
include
CLASS: ProposalBE.cls (main\src\Proposal)
invoke
PUBLIC-PROPERTY
invoke
CLASS: ValidateHelper.cls (main\src\Utils)
CLASS: StageErrorMessageEnum.cls (main\src\Stage)

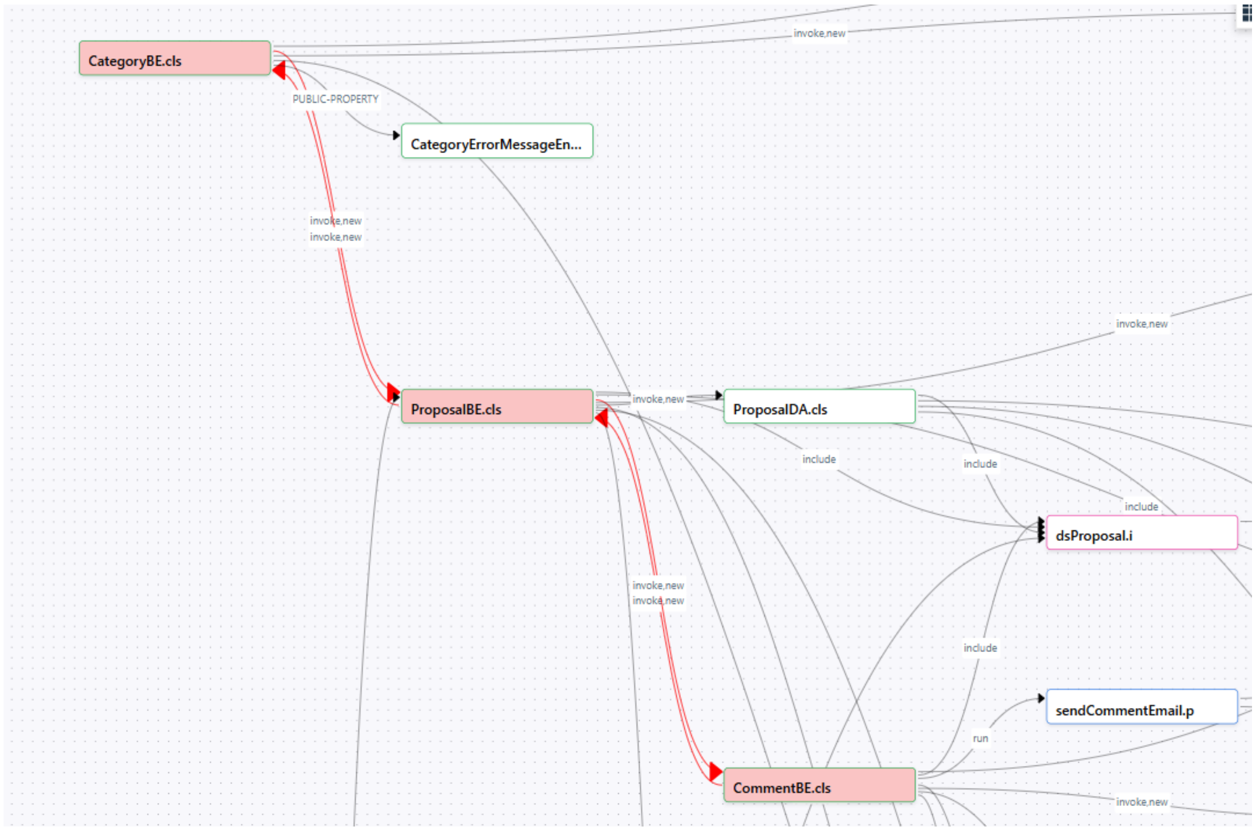
- Crossway Diagram Viewer: se aplica? Pare ca nu

Circular Dependencies

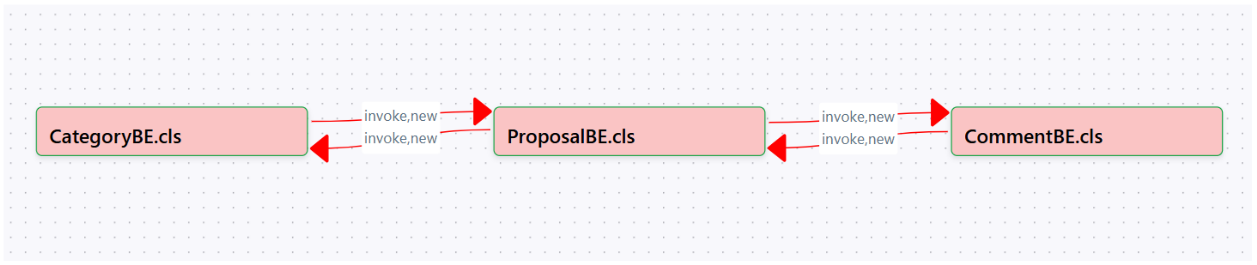
- A circular dependency is represented by two red arrows and it means that the classes reference each other.
- In the above example, circular dependencies can be seen between ProposalBE and CategoryBE and also between ProposalBE and CommentBE.
  - draw.io Diagram:


include
include
CLASS: CategoryBE.cls (main\src\Category)
invoke,new
invoke,new
CLASS: CommentBE.cls (main\src\Comment)
invoke,new
invoke,new
invoke,new
CLASS: ProposalBE.cls (main\src\Proposal)
CLASS: FileAttachmentBE.cls (main\src\FileAttachment)
CLASS: StageBE.cls (main\src\Stage)
invoke,new
invoke,new
invoke,new
invoke,new
CLASS: StageDA.cls (main\src\Stage)
PUBLIC-PROPERTY
PUBLIC-PROPERTY
CLASS: ProposalDA.cls (main\src\Proposal)
CLASS: StageErrorMessageEnum.cls (main\src\Stage)
CLASS: ProposalErrorMessageEnum.cls (main\src\Proposal)

  - CrosswayDiagramViewer:



- Crossway Diagram Viewer provides an option to display just the circular dependencies:



Implements Diagram

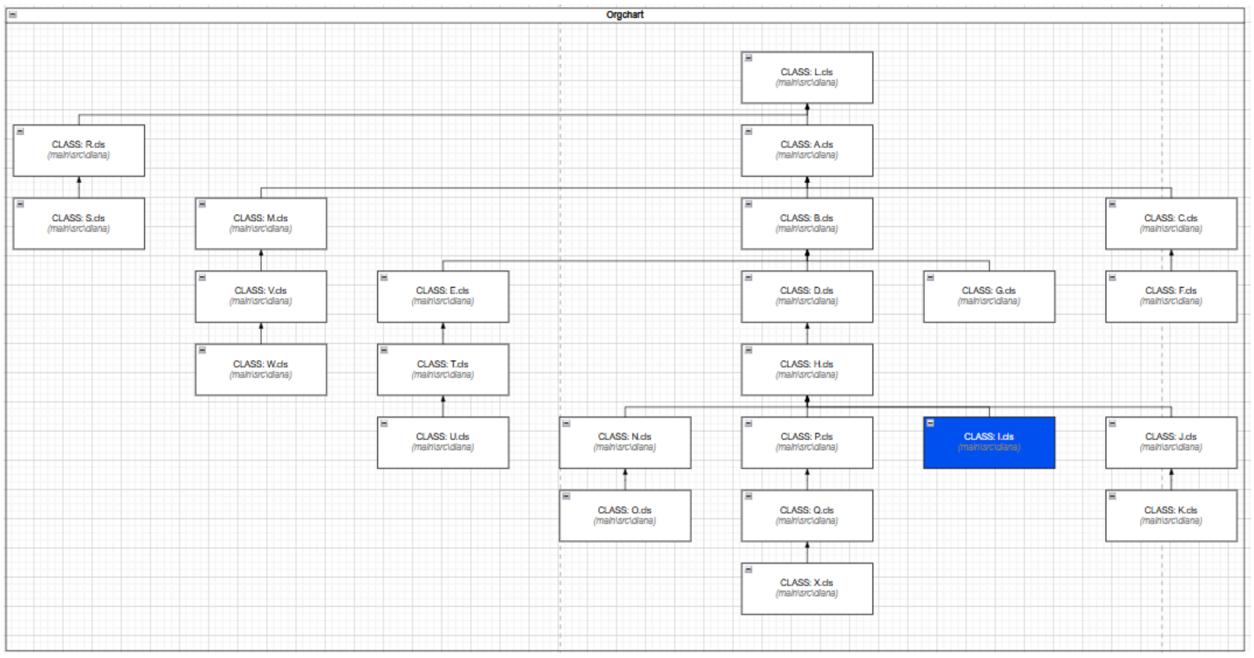
Ai poze la diana in folder

Inherits Diagram

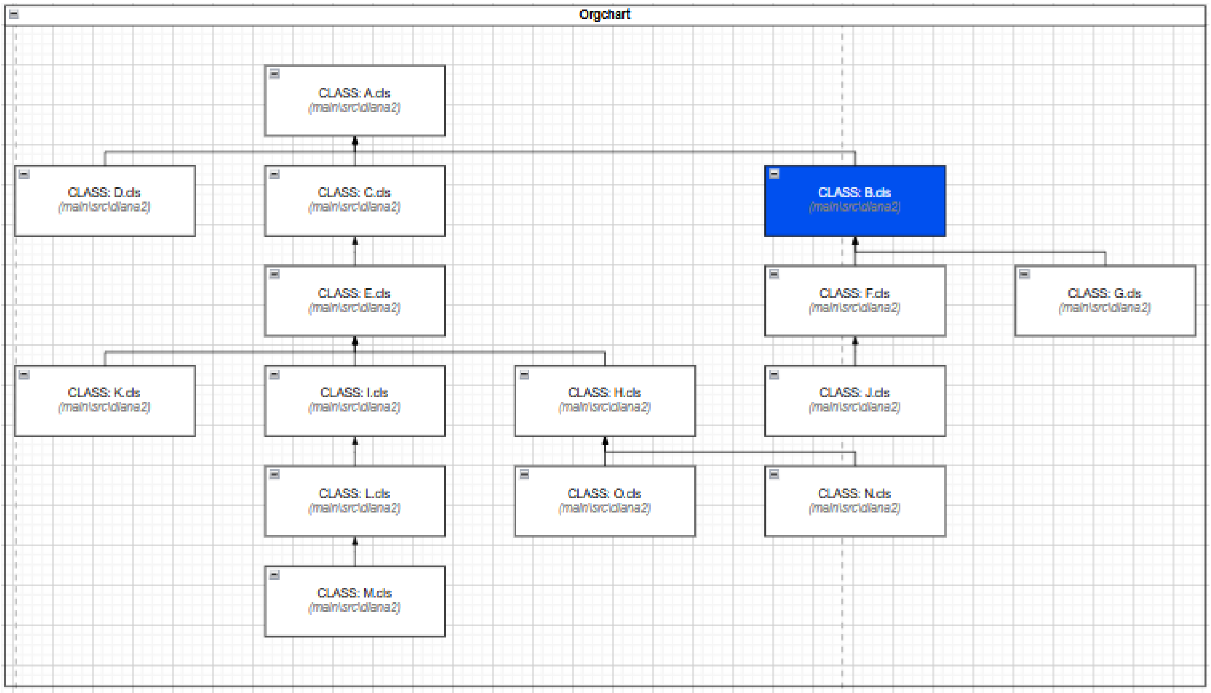
An inheritance diagram visually represents the hierarchical relationships between classes in object-oriented programming. It shows how classes derive attributes and behaviors from one another, illustrating the flow of properties and methods from parent (super) classes to child (sub) classes.

After pressing the Generate 'Inherits' Diagram button, draw.io will open and display the diagram. For the purpose of presenting this diagram type, we will use a custom project with multiple classes that include inheritance.

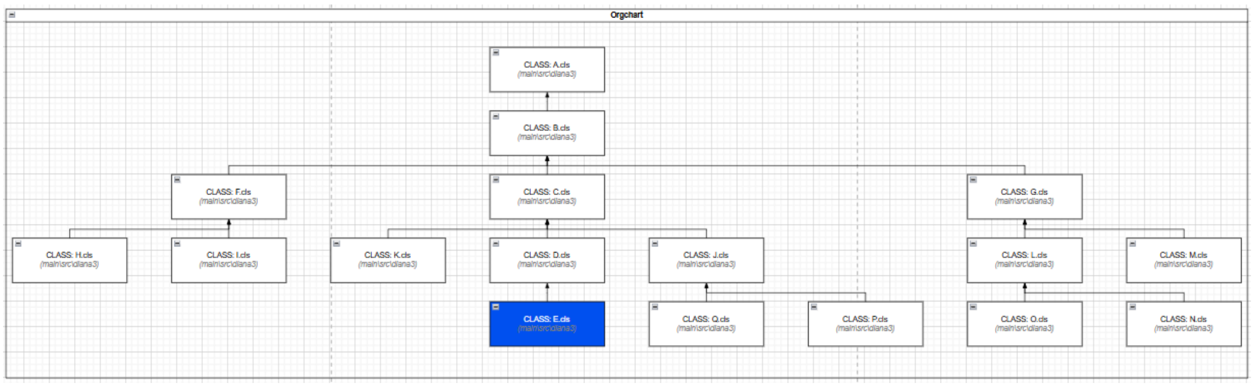
Example 1:


Orgchart
CLASS: L.cls (main\src\diana)
CLASS: R.cls (main\src\diana)
CLASS: A.cls (main\src\diana)
CLASS: S.cls (main\src\diana)
CLASS: M.cls (main\src\diana)
CLASS: B.cls (main\src\diana)
CLASS: C.cls (main\src\diana)
CLASS: V.cls (main\src\diana)
CLASS: E.cls (main\src\diana)
CLASS: D.cls (main\src\diana)
CLASS: G.cls (main\src\diana)
CLASS: F.cls (main\src\diana)
CLASS: W.cls (main\src\diana)
CLASS: T.cls (main\src\diana)
CLASS: H.cls (main\src\diana)
CLASS: U.cls (main\src\diana)
CLASS: N.cls (main\src\diana)
CLASS: P.cls (main\src\diana)
CLASS: I.cls (main\src\diana)
CLASS: J.cls (main\src\diana)
CLASS: O.cls (main\src\diana)
CLASS: Q.cls (main\src\diana)
CLASS: K.cls (main\src\diana)
CLASS: X.cls (main\src\diana)

Example 2:



Example 3:



Ai poze la diana in folder

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.

- For the Impact Diagram using draw.io there is support for automatic arrangement of nodes within the diagram: draw.io → Layout → Arrange → Horizontal Flow.
- Add more Useful notes/hints for the user

6. Troubleshooting & FAQ

Common errors and how to resolve them

- Why is my diagram empty?
    - ???
- Missing table relationship?
    - ???

7. Contact & Feedback

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