



The Labnaf Strategy & Architecture Framework

Labnaf Language Transformer

User Guide

See also the “**Labnaf Language Transformer - Reference Guide**”

WARNING

NEVER use the language transformer on your production repository before performing all necessary tests.

ALWAYS test your language transformer commands using a repository backup.

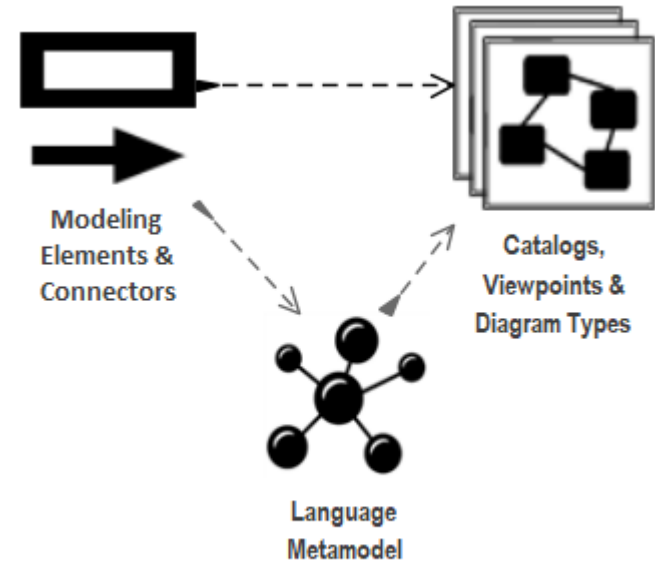
ALWAYS carefully check the resulting transformations and possible side effects. For example items could be deleted because you misspelled a type.

ALWAYS remember that type and stereotype names are case sensitive.

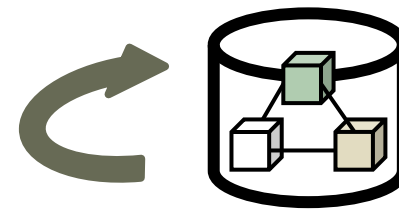
The Labnaf Language Transformer has been tested using Sparx Systems' Enterprise Architect versions 13.5 to 15.1

Labnaf Customization Steps

1. Customize the language following your organization requirements

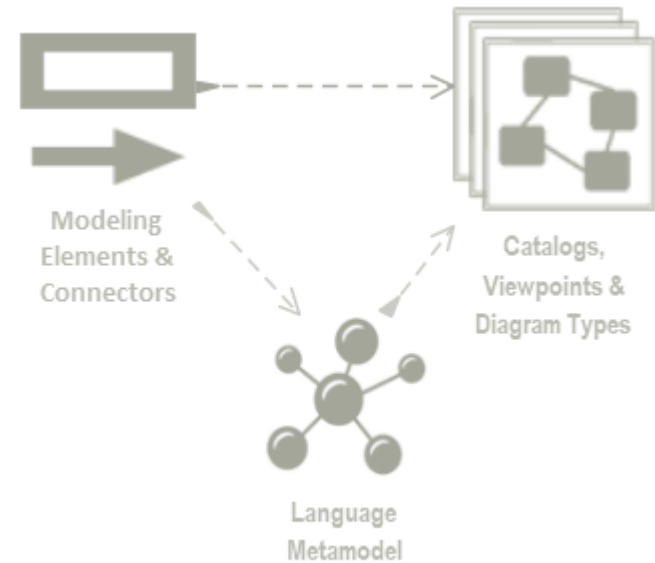


2. Adapt existing repository content

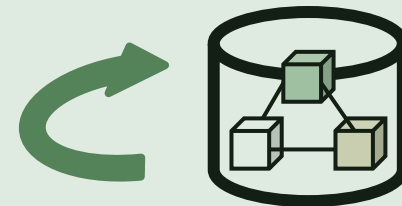


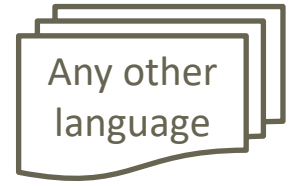
Labnaf Language Transformer

1. Customize the language

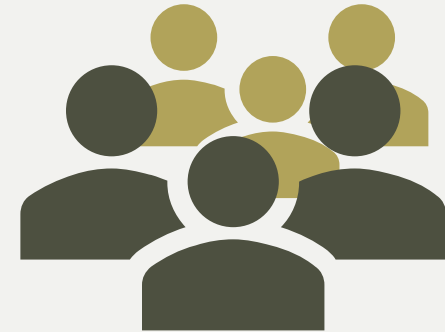


2. Adapt existing repository content

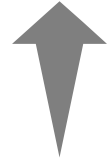
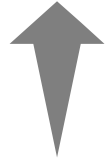
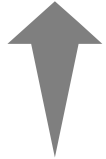
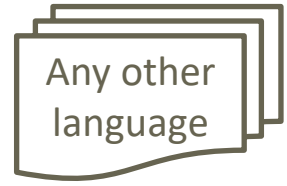




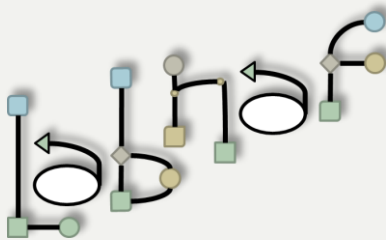
Transform any language...



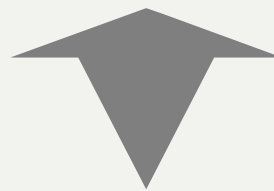
We all speak the same language



... in any direction



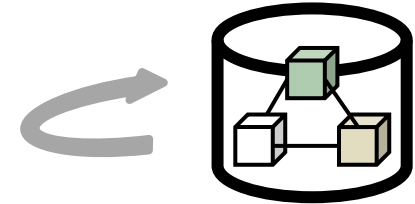
One **Single**
Strategy & Architecture
Modeling **Language**



You are never
locked in
Labnaf !

The Language Transformer adapts the language in existing repository content

- **ChangeElementType**
- **ChangeConnectorType**
- **ChangeDiagramType**
- **ChangeDiagramTypesDefinedInCsv**
- **TemplateMetamodelFromActiveMetamodel**
- **TvRename**
- **TvDelete**



See the **Language Transformer Reference Guide** for command specific information.

Prerequisites

Install the Language Transformer using the

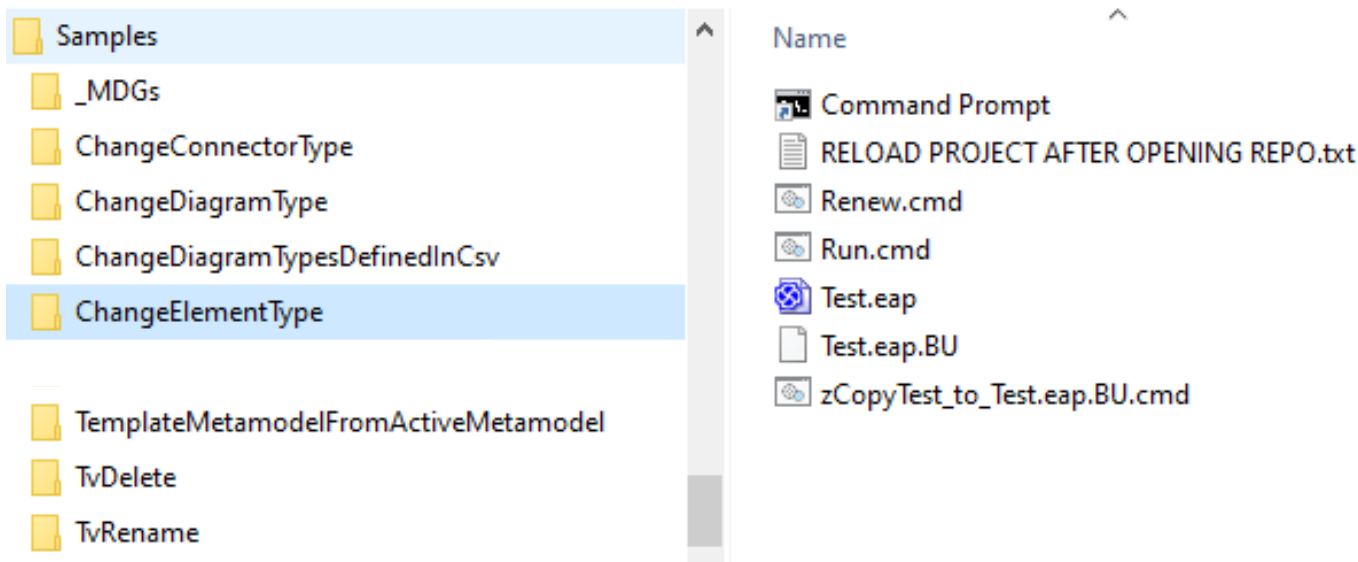
Installer/LabnafLanguageTransformerSetup.msi

Unzip the sample language transformations files stored in

Doc/SampleLanguageTransformations.zip

Prerequisites

Review the content of the Sample language transformations folder that you unzipped



For each Language Transformer command, there is a sample folder with an example. Folder names are language transformer command names.

The folders have a common structure. The file names are about the same. The content of the “Run.cmd” and “Test.eap” files are specific to each folder/command.

Development Lifecycle

- To transform a language in a repository, you will need a combination of transformer commands.
- We advise to address each command separately and in that order:
 - **ChangeElementType**
 - **ChangeConnectorType**
 - **ChangeDiagramType** or **ChangeDiagramTypesDefinedInCsv**
 - **TvRename**
 - **TvDelete**

The development lifecycle is the same for each command.

Keep backups of your different versions.

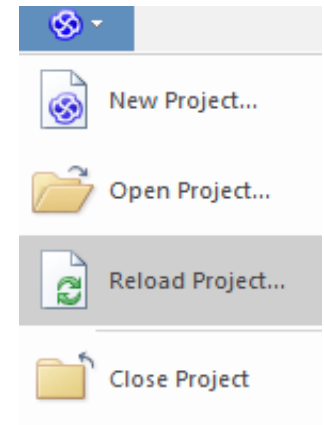
Develop and Test Your Script (cont.)

Edit and run the test script

- Edit **Run.cmd**. Put your own transformation commands
- Execute **Run.cmd**

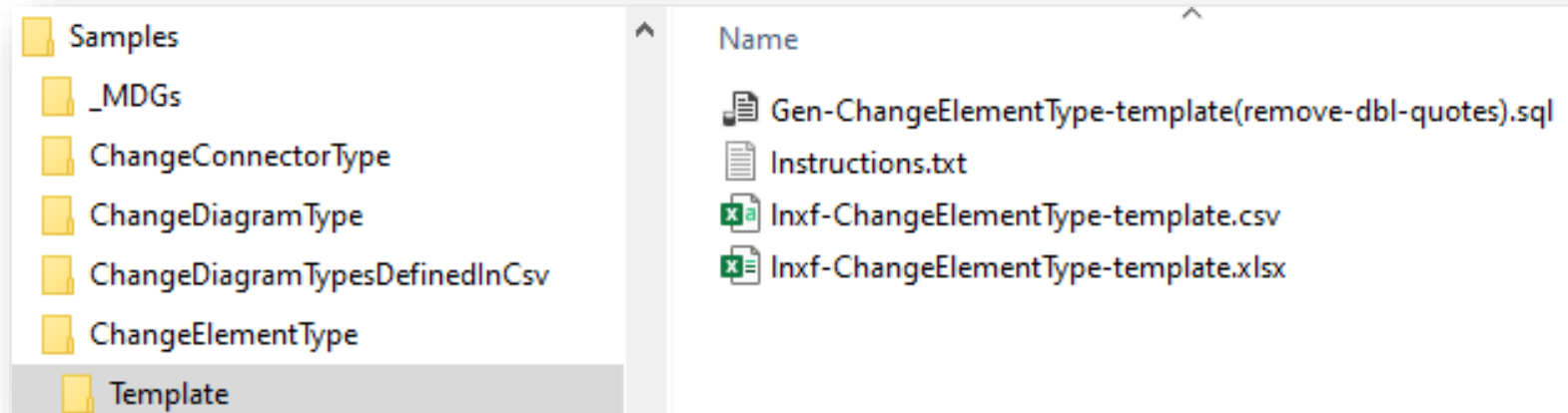
See the language transformation results

- Open **Test.eap**
- **In Sparx EA**
 - Reload the project
 - => the shapes are updated in the diagram
 - Remove the shapes from the diagram and move the back in the diagram
 - => the icons are updated in the project browser



Keep backups of your different versions.

Use the Excel Templates to define your mappings



- Map source to target types and stereotypes in Excel

	A	B	C	D	E	F	G	H	I
1	Inxf	ChangeElementType	{RepositoryFilePath}	{FromType}	{FromStereoType}	Component	LABN_AccessPoint	{ToTagName}	{ToTagValue}
2	Inxf	ChangeElementType	{RepositoryFilePath}	{FromType}	{FromStereoType}	Activity	LABN_Activity	{ToTagName}	{ToTagValue}
3	Inxf	ChangeElementType	{RepositoryFilePath}	{FromType}	{FromStereoType}	Component	LABN_Application	{ToTagName}	{ToTagValue}
4	Inxf	ChangeElementType	{RepositoryFilePath}	{FromType}	{FromStereoType}	Collaboration	LABN_ApplicationAsAService	{ToTagName}	{ToTagValue}
5	Inxf	ChangeElementType	{RepositoryFilePath}	{FromType}	{FromStereoType}	Component	LABN_ApplicationComponent	{ToTagName}	{ToTagValue}

- Create your transformation commands based on the Excel

Develop and Test Your Script

Adapt the test data as needed

- Execute **Renew.cmd**
 - to copy the backup database into a fresh **Test.eap**
- Open and edit **Test.eap**.
 - Replace the content by you own test data
 - Close **Test.eap**
- Execute "**zCopyTest_to_Test.eap.BU.cmd**"
 - to create a new backup database from **Test.eap**. So you will then be able to Renew the test database with your own data before running each test.

Keep backups of your different versions.

Test on real data

- Backup your PROD repository into an Access repository (EAP).
- Replace “Test.eap” by that backup repository.
- Execute the transformation script “Run.eap” and see the result

Run your transformation script on the PROD repository

- Ensure nobody is working on the repository
- Make a backup of you PROD repository
- Replace “Test.eap” by a shortcut to your PROD repository.
- Execute the transformation script “Run.eap” and see the results

Keep backups of your different versions.

To create a **new metamodel** from scratch

- Use **TemplateMetamodelFromActiveMetamodel**