# Final Project of System Design

CAS 703

McMaster University

**Team members:**

* Sayed Ahmed
* Sepehr Bayat

# Part 1: Project Proposal

We aim to develop a specialized Domain Specification Language (DSL) that automates the initial and exploratory analysis for regression-based data analytics projects. These types of projects involve common tasks such as Data Cleaning, Initial Analysis, and Exploratory Data Analysis that require significant time and resources to complete.

Our proposed DSL will automate these tasks, allowing developers to focus on more complex analysis. Users can define data sources and relationships using a GUI or text format, with the option to choose from predefined selections for relationships. The DSL will generate code to clean/replace null/missing values, align data types to the majority type, calculate r-square/adjusted r-square, perform data normalization, generate unitary plots, bi-variate plots, and bi-variate regression plots with the target.

The DSL can also be configured to generate code for a pre-configured set of tasks, eliminating the need for developers to create a custom task list. This feature ensures that the DSL is accessible to a wide range of users, regardless of their technical expertise.

Our DSL will improve the efficiency of regression-based data analytics projects, providing a powerful tool for developers to perform complex analyses while saving time and resources.

# Part 2: Creating Meta Model

## Class Diagram of Metamodel

The following is the class diagram of our metamodel. For a better view, we have also included the original image file in the report.   
  
Diagram, engineering drawing, schematic

Description automatically generated

Figure 1 Class Diagram of Metamodel

## Metamodel Description

The followings are the key components of our meta model:

* The root element in our project is Data Curation State Transition Action. Basically, it is where the project starts and connect other states to it. It contains Label and Name as attributes.
* The other state is Data which is the key component in this DSL. This state has some relations with some other states like Data Value and Attributes which respectively used to contain the information regarding the value of data and attributes like data type.
* The other high-level state is Initial Data Analysis. We label some tasks like Null Values and Missing values as an Initial Data Analysis. So, this state has a relation with Null Values and Missing Values state. We also define a transition station to be able to move sequentially between these states. There is no constraint, and you can move from Null Values to Missing Values or vice versa.
* The other high-level state that we have is Exploratory Data Analysis. We categorize some tasks like Bivariate Analysis, Univariate Visualization and Bivariate Visualization as an exploratory data analysis. As we mentioned in the Initial Data Analysis, we can move from different tasks like from Unitary analysis to Bivariate analysis and vice versa. That’s the reason we have a transition state.

## Assumptions

The followings are our assumptions:

* Data values should be integer and cannot be any other data types.
* There are so many data analysis tasks and prediction functionalities, but we are not doing all of them and we are doing some basic and useful functionalities.
* We also categorized tasks into two different types which are initial data analysis and exploratory data analysis.
* Tasks or even can happen sequentially, it means that Null values can happen after the Missing values, and it does not happen parallelly.
* Also, we are setting some constraints that the initial data analysis tasks must be done before the exploratory data analysis tasks.
* Another constraint that we are defining is that the data should be defined and other than that you cannot do any data analysis tasks.

## Alternative Design Decisions

The only alternative decision that we have in our mind is to use only one category of tasks instead of having two categories. Currently we are defining initial data analysis and exploratory data analysis. In that way, we do not use categories, or we only use data analysis event. Then later in the GMF, we can define constraints regarding the order of tasks.

# Part 3: Concrete Syntax and Editor

## Screenshots of the editors

**Graphical user interface, application, Teams

Description automatically generated**

Figure 2 GMF Editor

## Discuss and justify syntax design and editor implementation decisions

We have been taught different editors on this course to use. For our project we use GMF.

In our project, there is no hierarchy, and it is mostly moving from one state to another state regarding some constraints. We decided to use GMF instead of other approaches. Also, Xtext can be a good approach for use, but it needs a lot of manual coding. The good thing about GMF is that it is very easy to use, and developers can easily move between different states by visually designing them.

## The strength and weakness of the selected concrete syntax compared to alternatives.

### Strength

* GMF is a categorical editor and operations over the design can be simply done with the interface. It is very useful and easy to work.
* For our development purpose is very good, since we have some states which are being done sequentially. So, developers can easily draw them and create transitions between states.
* Another good point about GMF is the modelers do not need write any code or type any scripts and it can happen just by drawing.
* The modelers can easily watch the flow and what steps are happening right after each other.

### Weaknesses

* If the modelers want to do so many operations, they must draw so many states and connect them which can be a tedious task.

# Part 4: Implementation of validation constraints

## Explanation of constraints

Based on the discussions that we had in our team we came up with the following constraints that can not be visible in the meta model:

* Data must be provided to the project and other than that modelers cannot do anything. So, the first essential part of any model is the existence of data.
* The data type must be integer and our DSL does not work any other type of data.
* The initial data analysis must be done before the exploratory data analysis tasks. It means that if you want to have any initial data analysis you should make sure that it is happening before the exploratory data analysis tasks. You can ignore the initial data analysis or exploratory task but if you want to have them you should consider the orders that we defined.
* Tasks cannot be done in parallel. It means that you cannot do two initial data analysis tasks simultaneously and then merge them together.

## Implementation of constraints

To execute the EVL we created a reference to the ecore model as you can see in the following image:

**Graphical user interface, text, application, chat or text message

Description automatically generated**

# Model Management operation