DESIGN AND DEVELOPMENT OF THE IDEALVIS ADAPTATION ENGINE

D16 (SOFTWARE)

IDEALVis Consortium

http://idealvis.inspirecenter.org/

European Union
European Regional Development Fund

Republic of Cyprus

Structural Funds
of the European Union in Cyprus

ΙΔΡΥΜΑ ΕΡΕΥΝΑΣ ΚΑΙ ΚΑΙΝΟΤΟΜΙΑΣ

Executive Summary

This deliverable demonstrates the architecture of the IDEALVis adaptation engine, which is the core component of the IDEALVis framework, recommending and generating the visualization engine instructions for best fit data visualizations. This engine incorporates the adaptation rules/mapping rules which were extracted/generated by analyzing the impact of human factors on data visualization types and different data visualization elements. Such rules contain the logic that dictates: (i) which is the best fit data visualization in terms of type (e.g., bar- or line-chart); and (ii) which visual elements should be altered/enabled on that data visualization, according to the user's user model and the underlying analysis task being performed. This deliverable summarizes the main operations of the adaptation engine, presenting its architecture using appropriate diagrams and illustrations, helping the reader grasp the different inputs that the engine receives, as well as the resulting processed outputs it generates. Additionally, this deliverable highlights some of the design decisions taken when developing the adaptation engine.

Table of Contents

List of Figures

1 Introduction

This deliverable is structured according to the execution flow of the adaptation procedure, illustrated in Figure 1. More specifically, the deliverable is structured into three sections, each one discussing the major steps of the adaptation engine execution. The three adaptation steps include the different inputs taken by the adaptation engine, the processing performed on these inputs and finally the adaptation engine's processed outputs. It must be noted that the diagram in Figure 1 summarizes the main processing steps and does not reflect the exact structure of the code (i.e., in terms of class composition/communication) or the exact component naming as it is found in code. For instance, the communication between two components in the diagram, in reality might be coded so that the two components use multiple other intermediate structures in order achieve communication. This deviation from the exact structure is essential for illustrating the flow of key information so each separate key component can be easily described in isolation.

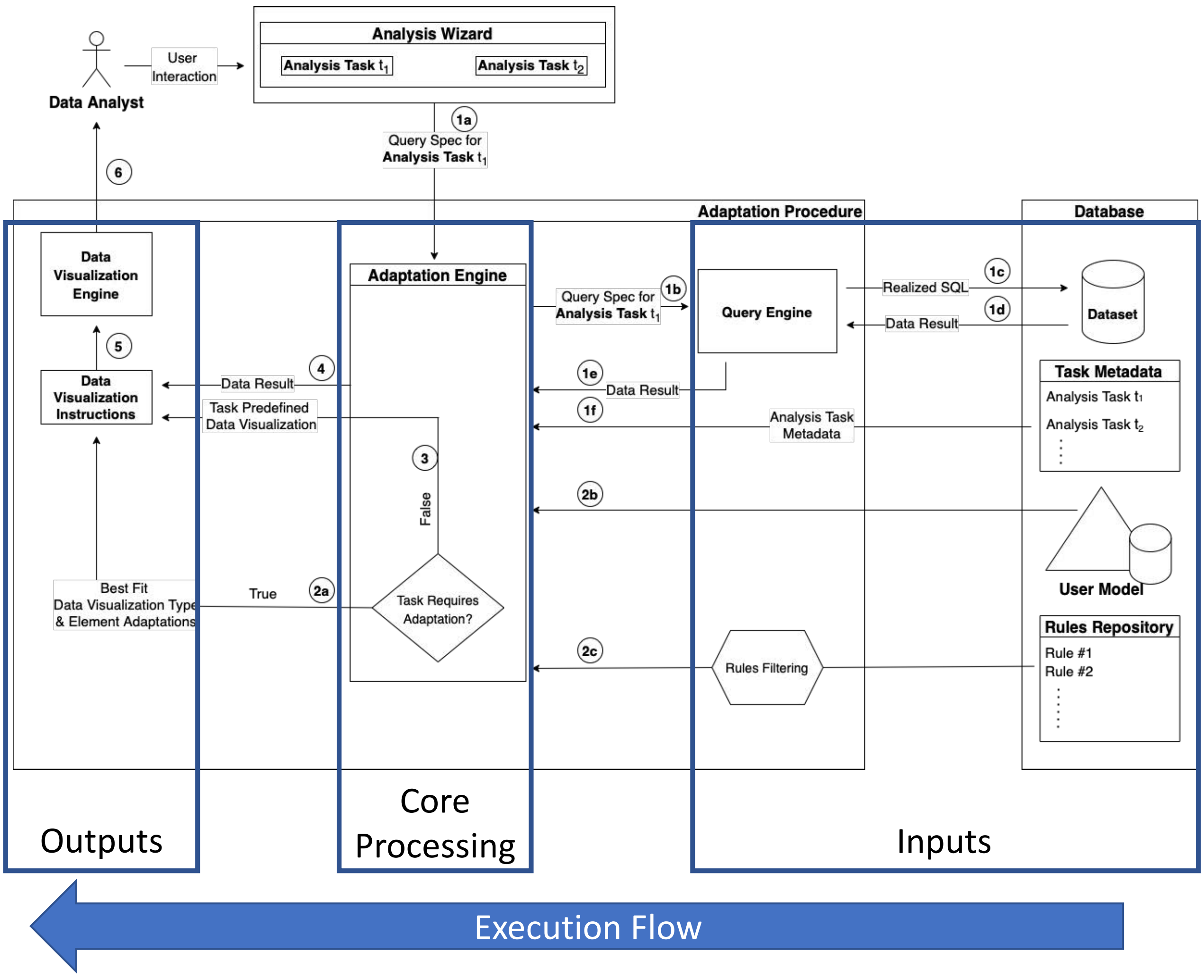


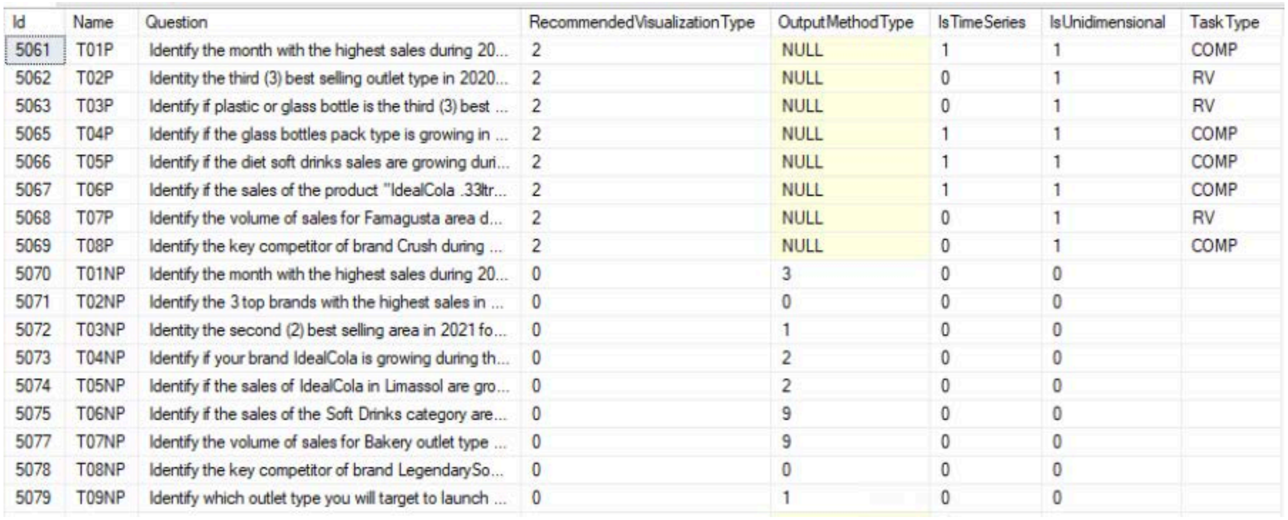
Figure 1 - The Overall Adaptation Procedure

2 Inputs to Adaptation Engine

The key inputs to the adaptation engine are the unique analysis task identifier that the user is performing at the current moment (i.e., the current task), the current task's metadata, the user's model and two sets of adaptation rules. The Query Engine process that provides the data presented in Figure 1 (point 1e) is not described as the adaptation engine is used as a passthrough of the data to the visualization engine (i.e., no specific processing is done on the data). Below we further describe each of the inputs and also present each input as it is stored in the database i.e., using data tables from the platform's database.

2.1 Task Unique Identifier and Task Metadata

The table presented in Figure 2 illustrates a set of analysis tasks and their metadata as they are stored in the IDEALVis database (only essential table attributes shown). When a user is addressing an analysis task via the Analysis Wizard interface the system is aware on which task the user is working on. Moreover, each task has a unique identifier *Id* which helps the adaptation engine retrieve or query the metadata for the current task a user is performing.



Id	Name	Question	RecommendedVisualizationType	OutputMethodType	IsTimeSeries	IsUnidimensional	TaskType
5061	T01P	Identify the month with the highest sales during 20...	2	NULL	1	1	COMP
5062	T02P	Identity the third (3) best selling outlet type in 2020...	2	NULL	0	1	RV
5063	T03P	Identify if plastic or glass bottle is the third (3) best ...	2	NULL	0	1	RV
5065	T04P	Identify if the glass bottles pack type is growing in ...	2	NULL	1	1	COMP
5066	T05P	Identify if the diet soft drinks sales are growing duri...	2	NULL	1	1	COMP
5067	T06P	Identify if the sales of the product "IdealCola .33ltr...	2	NULL	1	1	COMP
5068	T07P	Identify the volume of sales for Famagusta area d...	2	NULL	0	1	RV
5069	T08P	Identify the key competitor of brand Crush during ...	2	NULL	0	1	COMP
5070	T01NP	Identify the month with the highest sales during 20...	0	3	0	0	
5071	T02NP	Identify the 3 top brands with the highest sales in ...	0	0	0	0	
5072	T03NP	Identity the second (2) best selling area in 2021 fo...	0	1	0	0	
5073	T04NP	Identify if your brand IdealCola is growing during th...	0	2	0	0	
5074	T05NP	Identify if the sales of IdealCola in Limassol are gro...	0	2	0	0	
5075	T06NP	Identify if the sales of the Soft Drinks category are...	0	9	0	0	
5077	T07NP	Identify the volume of sales for Bakery outlet type ...	0	9	0	0	
5078	T08NP	Identify the key competitor of brand LegendarySo...	0	0	0	0	
5079	T09NP	Identify which outlet type you will target to launch ...	0	1	0	0	

Figure 2 - Analysis Tasks Table (partial result)

The key metadata required by the adaptation engine include the following attributes, which are part of the table in Figure 2:

- **RecommendedVisualizationType:** This attribute can hold one of the following coded values: 0 representing that for the current task there is a preselected data visualization; 1 representing that the task allows the user to select their own data visualization for the task; and 2 representing that the task requires a system recommended/adapted data visualization to be generated. Values 0 and 1 signify that the adaptation engine will not be activated.
- **OutputMethodType:** This attribute represents an enumeration that supports specific data visualization types. For example, when a task has a preselected visualization, this field receives the type of data visualization that should be returned by the system for that task. When the task has a value of 2 in the RecommendedVisualizationType attribute (i.e., task

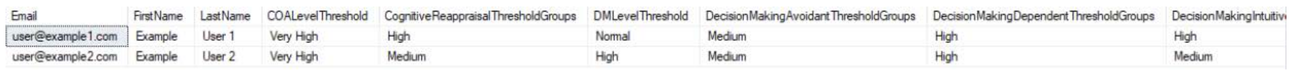
requires a system recommended data visualization) the OutputMethodType value is NULL meaning that the data visualization to be used is not known prior to running the analysis, since the adaptation engine will decide on which is the data visualization to be used for the task based on the user's user model.

- **IsTimeSeries:** Defines if the data of the task are time series data. This is used for later filtering appropriate adaptation rules regarding time series data.
- **IsUnidimensional:** Defines if the data of the task have a single or multiple dimensions. This is used for later filtering appropriate adaptation rules based on the complexity of the task's data.
- **TaskType:** Defines the task's type (i.e., Simple Comparison or COMP). This is used for later filtering appropriate adaptation rules based on the task's type.

NOTE: tasks that have a predefined OutputMethodType do not have values in the last three metadata fields (i.e., IsTimeSeries, IsUnidimensional and TaskType), since those metadata are only required by the adaptation engine for tasks that actually require adaptation.

2.2 The User Model

The table presented in Figure 3 illustrates a set of example user models as they are stored in the IDEALVis database (only some table attributes shown due to the large number of attributes being held by the UserModels table). When a user is addressing an analysis task that requires an adapted data visualization, the adaptation engine will fetch the corresponding row from the UserModels table representing the user's user model.



Email	FirstName	LastName	COALevelThreshold	CognitiveReappraisalThresholdGroups	DMLevelThreshold	DecisionMakingAvoidantThresholdGroups	DecisionMakingDependentThresholdGroups	DecisionMakingIntuitiv
user@example1.com	Example	User 1	Very High	High	Normal	Medium	High	High
user@example2.com	Example	User 2	Very High	Medium	High	Medium	High	Medium

Figure 3 - User Models Table

2.3 Adaptation Rule Sets

The adaptation rules are divided into two sets: (i) the *visualization type* adaptation rules; and (ii) the *visual element* adaptation rules. The visualization type adaptation rules are the ones that decided which is the best fit data visualization type to be used for the current task and they are the first ones that are processed by the adaptation engine. These rules are filtered according to (i) the user's user model and (ii) the current task's metadata (i.e., TaskType, IsTimeSeries, IsUnidimensional). Similarly, the *visual element* adaptation rules are used to guide the decision of which visual elements are to be altered/enabled on the data visualization type that was selected in the previous step. These rules are filtered according to: (i) the element of interest (i.e., the visual element the adaptation engine is currently trying to evaluate for the current situation e.g., DARK_MODE); (ii) the user's user model; and (iii) the best fit data visualization's type (e.g., Bar).
The rules are stored in a fuzzy rule-based classification form, which considered the outcomes of the analysis presented in D11. This representation serves two purposes: (i) to facilitate efficient execution of the rules; and (ii) to improve the explainability of the inference process carried out by the system.

A simplified example of a rule is presented below:

IF {TaskType=”Comparison” and WorkingMemory=”High”} THEN visualization type={Column=0.96, Bar=0.04”}

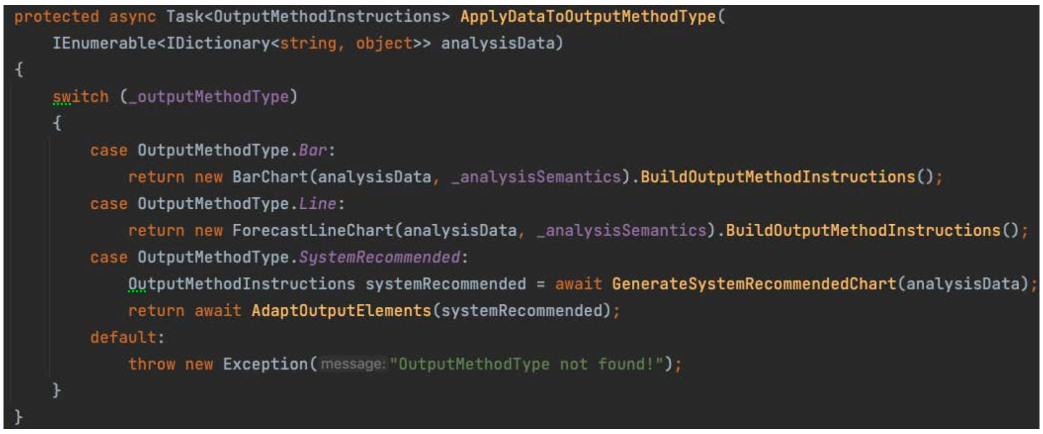
The rule is interpreted as follows: if the active task is of comparison type (e.g., compare the sales of two different periods) and the current user has high working memory, then “vote” for the column visualization type with 96% and for the bar visualization type with 4%. As it will be described in the next Section, the ensemble of all rules, considering all aspects of the user model, yields the final result, i.e., the visualization type with the highest weighted vote.

3 Adaptation Engine Core Processing

With the adaptation engine inputs discussed in the previous section, this section breaks down the inner processing, or the core processing and steps taken by the adaptation engine. The overall goal of the adaptation engine is to essentially process all its inputs and transform them in data visualization instructions, which the data visualization engine can parse for rendering the best fit data visualization. The inner working and processes of the adaptation engine are broken down in sequential steps in the next sub-sections.

3.1 *Adaptation Decision*

Initially the adaptation engine retrieves the current task's (i.e., the task which the user is working on) metadata. According to the task's metadata (described in the previous section), the adaptation engine will either initiate adaptation or terminate. In the case of termination, it will offer either the preselected data visualization (as defined by the task's metadata) or the user-selected data visualization. For instance, the code segment presented in Figure 4 illustrates how the adaptation engine being implements one of the visualization types output. Essentially, during the final step of the analysis the data generated from the user's query is being added to an OutputMethodInstructions object, which is used by the visualization engine to render the analysis data visualization output. Depending on the selected/preselected data visualization type a specific builder of output method instructions is called. When a specific analysis task requires adaptation, the system directly executes the SystemRecomended OutputMethodType (i.e., last switch case in the code) which is the core of the adaptation engine. The code executed essentially takes the analysis data and in two distinct steps it packs the data inside the adapted data visualization instructions or OutputMethodInstructions that will be used by the visualization engine to render the analysis adapted data visualization. In the next sub-sections, we provide information about the two distinct steps of the adaptation engine (seen in Figure 4) which are the GnerateSystemRecommendedChart procedure and the AdaptOutputElements procedure. The first procedure decides which is the best type of data visualization for the current task and user, and the second decides which are the visual elements to be altered/enabled on the selected best fit data visualization.

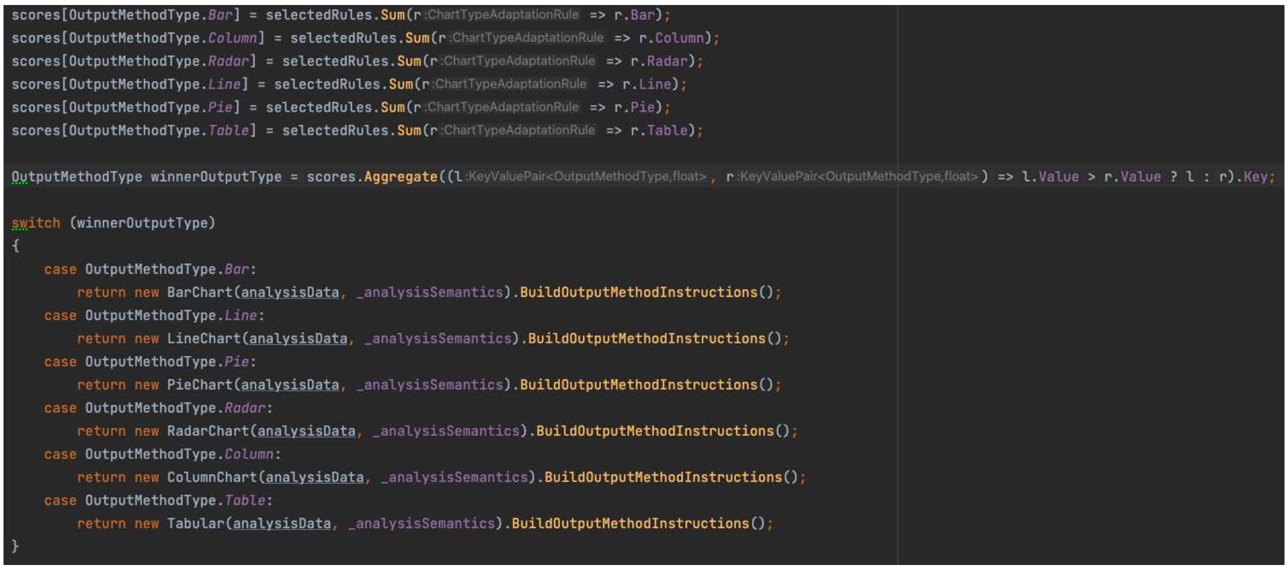


```
protected async Task<OutputMethodInstructions> ApplyDataToOutputMethodType(
    IEnumerable<IDictionary<string, object>> analysisData)
{
    switch (_outputMethodType)
    {
        case OutputMethodType.Bar:
            return new BarChart(analysisData, _analysisSemantics).BuildOutputMethodInstructions();
        case OutputMethodType.Line:
            return new ForecastLineChart(analysisData, _analysisSemantics).BuildOutputMethodInstructions();
        case OutputMethodType.SystemRecommended:
            OutputMethodInstructions systemRecommended = await GenerateSystemRecommendedChart(analysisData);
            return await AdaptOutputElements(systemRecommended);
        default:
            throw new Exception(message: "OutputMethodType not found!");
    }
}
```

Figure 4 - Visualization Instruction Builders Example Code

3.2 *Selecting the Best Fit Data Visualization*

Initially, for selecting the best type data visualization the adaptation engine filters down the visualization type rules using the participants user model and the analysis task's metadata as described in Section 2.3. Each row in the filtered rules dataset represents a single rule. Moreover, each of the resulting rules contains a set of data visualization rankings. Those rankings are represented by the following visualization type adaptation rules table attributes (Bar, Radar, Column, Line, Pie, and Table). Essentially, each rule represents a condition that applies to the current user and analysis task they are currently addressing. Since the user model contains multiple dimensions for each user, the set of resulting rules after the adaptation engine filters the rules, may have as many rules as the number of user model dimensions. For that reason, rule scores for each data visualization type are aggregated in an ensemble manner, and the winner data visualization is selected as seen in the example code of Figure 5. Finally, based on the winner data visualization type the appropriate OutputMethodInstructions builder is called. The data visualization instructions built in this phase contain the default/not-altered visual elements for the winner/best fit data visualization type. Therefore, the process described in the next section takes those resulting instructions from this process and enhances them with instructions regarding the appropriate visual element that should be enabled on the best fit data visualization.

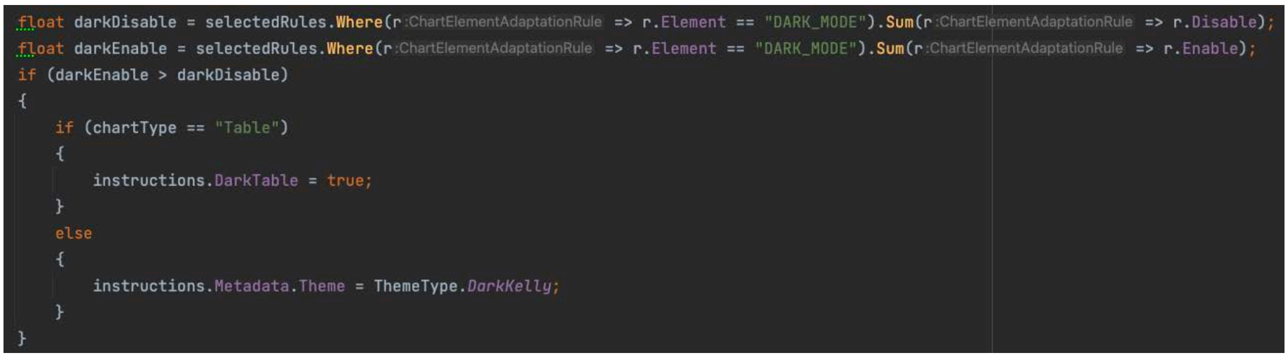


```
scores[OutputMethodType.Bar] = selectedRules.Sum(r:ChartTypeAdaptationRule => r.Bar);
scores[OutputMethodType.Column] = selectedRules.Sum(r:ChartTypeAdaptationRule => r.Column);
scores[OutputMethodType.Radar] = selectedRules.Sum(r:ChartTypeAdaptationRule => r.Radar);
scores[OutputMethodType.Line] = selectedRules.Sum(r:ChartTypeAdaptationRule => r.Line);
scores[OutputMethodType.Pie] = selectedRules.Sum(r:ChartTypeAdaptationRule => r.Pie);
scores[OutputMethodType.Table] = selectedRules.Sum(r:ChartTypeAdaptationRule => r.Table);

OutputMethodType winnerOutputType = scores.Aggregate((l:KeyValuePair<OutputMethodType,float> , r:KeyValuePair<OutputMethodType,float>) => l.Value > r.Value ? l : r).Key;

switch (winnerOutputType)
{
    case OutputMethodType.Bar:
        return new BarChart(analysisData, _analysisSemantics).BuildOutputMethodInstructions();
    case OutputMethodType.Line:
        return new LineChart(analysisData, _analysisSemantics).BuildOutputMethodInstructions();
    case OutputMethodType.Pie:
        return new PieChart(analysisData, _analysisSemantics).BuildOutputMethodInstructions();
    case OutputMethodType.Radar:
        return new RadarChart(analysisData, _analysisSemantics).BuildOutputMethodInstructions();
    case OutputMethodType.Column:
        return new ColumnChart(analysisData, _analysisSemantics).BuildOutputMethodInstructions();
    case OutputMethodType.Table:
        return new Tabular(analysisData, _analysisSemantics).BuildOutputMethodInstructions();
}
```

Figure 5 - Ranking the Best Fit Data Visualization Example Code

3.3 Selecting Visual Element Adaptations

The next step taken by the analysis engine is the selection of visual element alterations that are to be enabled on the best fit data visualization. In this step, we only describe the process performed for a single visual element as the process is very similar across all elements. Initially, for selecting if a visual element alteration is to take place on the best fit data visualization, the adaptation engine filters down the visualization elements rules using the participants user model, the visual element of interest (e.g., DARK_MODE) and the best fit data visualization's type as described in Section 2.3. Similarly, to the visualization type, each of the resulting rules contains a pair of rankings that dictate if the element is to be enabled for the specific rule condition. Those rankings are represented by the enable/disable visualization element adaptation rules. Since the user model contains multiple dimensions for each user, the set of resulting rules after the adaptation engine filters the rules, may contain as many as the number of user model dimensions. For that reason, rule scores returned for each visual element are aggregated and the visual element is only enabled on the data visualization if the aggregated value for Enabled is greater than that of Disabled as see in Figure 6. Finally, once the decision has been taken on whether an element is to be enabled on the best fit data visualization, the OutputMethodInstructions object (i.e., instructions variable in Figure 6) is modified accordingly. Note that the OutputMethodInstructions object was passed to this procedure from the previous procedure where the adaptation engine ranked the best fit data visualization type. The described process is the same across all different visual elements, with the only difference being in the OutputMethodInstructions object properties that need to be modified for each individual visual element.

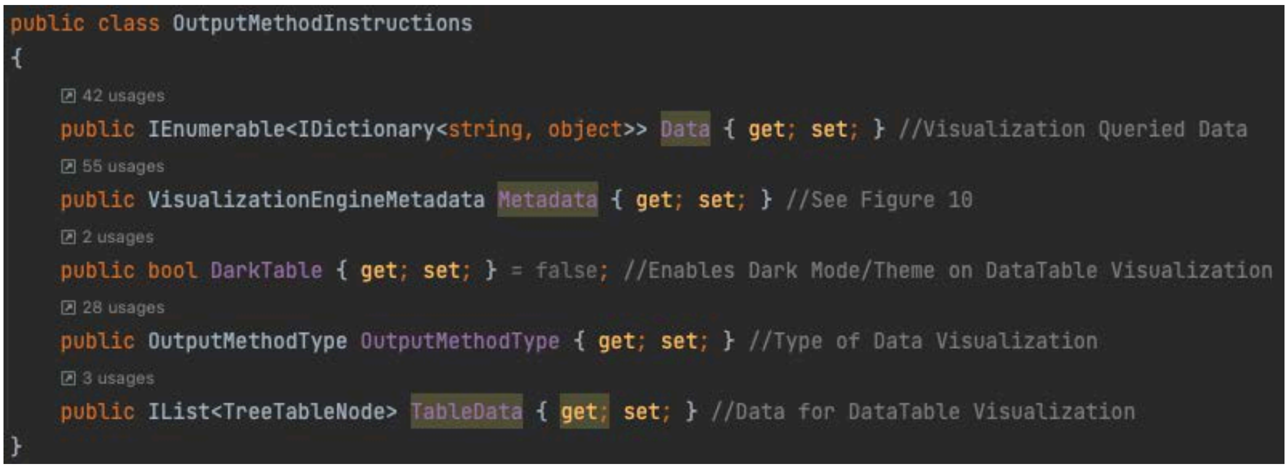


```
float darkDisable = selectedRules.Where(r :ChartElementAdaptationRule => r.Element == "DARK_MODE").Sum(r :ChartElementAdaptationRule => r.Disable);
float darkEnable = selectedRules.Where(r :ChartElementAdaptationRule => r.Element == "DARK_MODE").Sum(r :ChartElementAdaptationRule => r.Enable);
if (darkEnable > darkDisable)
{
    if (chartType == "Table")
    {
        instructions.DarkTable = true;
    }
    else
    {
        instructions.Metadata.Theme = ThemeType.DarkKelly;
    }
}
```

Figure 6 - Ranking Dark Mode Visualization Element Example Code

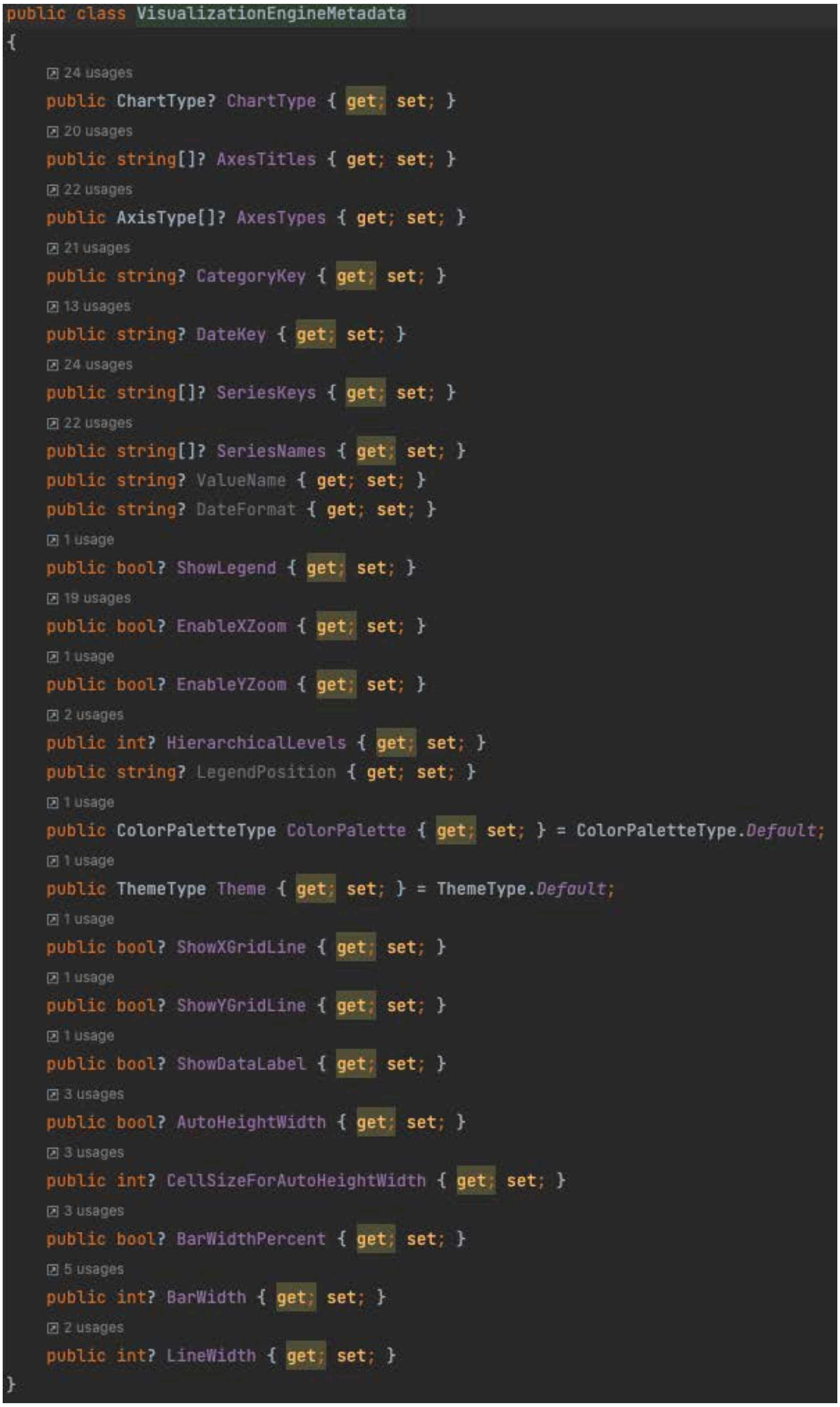
4 Adaptation Engine Outputs

The previous sections have presented the adaptation engine's inputs and the processing performed on those inputs. Specifically in Section 3.2 we have presented how an OutputMethodInstructions object was created for the best fit data visualization. Furthermore, in Section 3.3 we have presented how a OutputMethodInstructions object was further modified to include visual element adaptations for the best fit data visualization. Essentially, this OutputMethodInstructions object is the final object returned by the adaptation engine. Moreover, this is the object used by the visualization engine to render the final best fit data visualization on the user's browser. In this section, we provide the structure and properties of the OutputMethodInstructions object and the VisualizationEngineMetadata object with appropriate code comments regarding some properties for further illustrating how the adaptation engine dictates to the visualization engine on what is to be rendered.



```
public class OutputMethodInstructions
{
    42 usages
    public IEnumerable<IDictionary<string, object>> Data { get; set; } //Visualization Queried Data
    55 usages
    public VisualizationEngineMetadata Metadata { get; set; } //See Figure 10
    2 usages
    public bool DarkTable { get; set; } = false; //Enables Dark Mode/Theme on DataTable Visualization
    28 usages
    public OutputMethodType OutputMethodType { get; set; } //Type of Data Visualization
    3 usages
    public IList<TreeTableNode> TableData { get; set; } //Data for DataTable Visualization
}
```

Figure 7 - Partial OutputMethodInstructions Object Code



```
public class VisualizationEngineMetadata
{
    24 usages
    public ChartType? ChartType { get; set; }
    20 usages
    public string[]? AxesTitles { get; set; }
    22 usages
    public AxisType[]? AxesTypes { get; set; }
    21 usages
    public string? CategoryKey { get; set; }
    13 usages
    public string? DateKey { get; set; }
    24 usages
    public string[]? SeriesKeys { get; set; }
    22 usages
    public string[]? SeriesNames { get; set; }
    public string? ValueName { get; set; }
    public string? DateFormat { get; set; }
    1 usage
    public bool? ShowLegend { get; set; }
    19 usages
    public bool? EnableXZoom { get; set; }
    1 usage
    public bool? EnableYZoom { get; set; }
    2 usages
    public int? HierarchicalLevels { get; set; }
    public string? LegendPosition { get; set; }
    1 usage
    public ColorPaletteType ColorPalette { get; set; } = ColorPaletteType.Default;
    1 usage
    public ThemeType Theme { get; set; } = ThemeType.Default;
    1 usage
    public bool? ShowXGridLine { get; set; }
    1 usage
    public bool? ShowYGridLine { get; set; }
    1 usage
    public bool? ShowDataLabel { get; set; }
    3 usages
    public bool? AutoHeightWidth { get; set; }
    3 usages
    public int? CellSizeForAutoHeightWidth { get; set; }
    3 usages
    public bool? BarWidthPercent { get; set; }
    5 usages
    public int? BarWidth { get; set; }
    2 usages
    public int? LineWidth { get; set; }
}
```

Figure 8 – VisualizationEngineMetadata Object Code

5 Conclusions

This software deliverable provided a summary the overall development and architecture of the adaptation engine. Specifically, the deliverable presented the overall adaptation engine architecture in a diagram and then it followed a sequential approach to demonstrating the adaptation procedure in three distinct steps. More specifically, the set of different inputs to the adaptation engine were made clear to the reader. Next, the two core operations of the adaptation engine were discussed including the selection of the best fit data visualization and also the selection of visual element modifications that are to be applied on the best fit data visualization. Finally, the deliverable presented the data visualization instructions which are used as the main communication means from the adaptation engine to the visualization engine.