Note Taking Environment for Visual Analytics Systems
Technical Description

This document describes technical aspects of the Note Taking Environment (NTE) divided in the following sections:

1. Introduction and Technical Background
2. NTE UI and Widget Description
3. API Description

1 Introduction and Technical Background

The foundation of our approach is a note-taking environment (NTE, java component) pluggable to any VA system. The NTE interface allows the analyst to create a knowledge graph that is composed of different widget types. The NTE is capable to run as stand alone tool but is more effective when it is tightly integrated with a VA system. Therefore it provides an Application Programming Interface (API), allowing developers to integrate their system and the NTE. The API allows VA tools to send bookmarks (visualizations) and additional information to the NTE. Additional information may be defined by the developer and depends on the analysis case at hand. These could be textual annotations, or measures, such as uncertainty information, data-, or visual measures that might be visualized in the NTEs knowledge graph. In addition to the visualization and context information, the NTE API is capable to handle a function implementation that will be called when a bookmark is clicked in the NTE (callback function). This allows a developer to define and set the current state/configuration of the bookmarked visualization (in order to "jump back" to this particular analysis state any times). In this way, the NTE API is completely independent from any application specifics of the VA system. The NoteTakingService also offers a logging interface (similar to common used logger systems in JAVA, e.g., log4j-http://logging.apache.org/log4j/2.x/) and allows for logging on different levels low/operational and high/task) and action types (according to the task typology described by Brehmer and Munzner [BM13]). The integration architecture is shown in Figure 1 on the left.

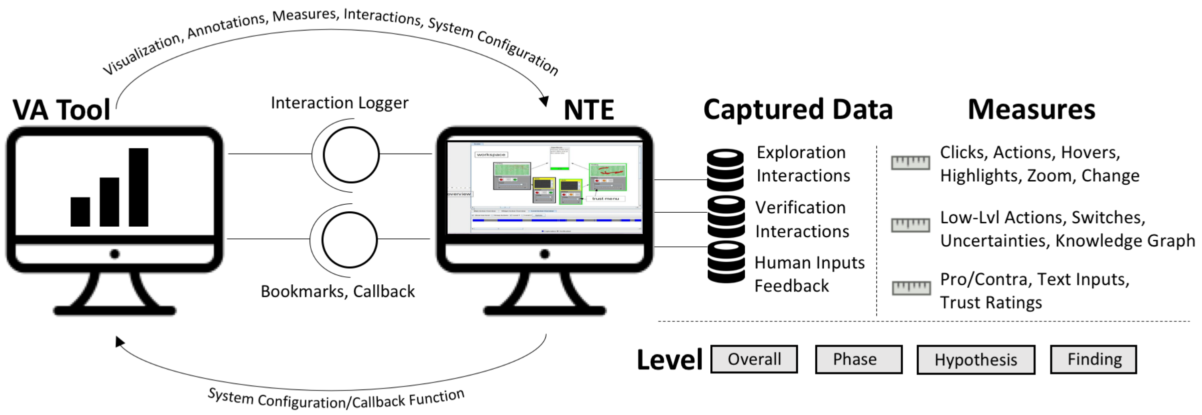

Visualization, Annotations, Measures, Interactions, System Configuration
VA Tool
Interaction Logger
Bookmarks, Callback
NTE
System Configuration/Callback Function
Captured Data
Exploration Interactions
Verification Interactions
Human Inputs Feedback
Measures
Clicks, Actions, Hovers, Highlights, Zoom, Change
Low-Lvl Actions, Switches, Uncertainties, Knowledge Graph
Pro/Contra, Text Inputs, Trust Ratings
Level
Overall
Phase
Hypothesis
Finding

Figure 1: Our capturing approach: A note taking environment is plugged to a VA system. This integration enables us to capture analytical processes on different levels. Additionally, the NTE enables analysts to build a knowledge graph and to take further human inputs, such as trust ratings. Finally, measures are derived as a basis for analyzing human factors in VA processes.

The knowledge graph elements and the interaction measures and further human inputs are captured and stored in data files that can be analyzed further. For example, for each widget, the annotation, interactions and trust rating is stored. Additionally, for the user study, a global trust rating was captured and stored, after each task.

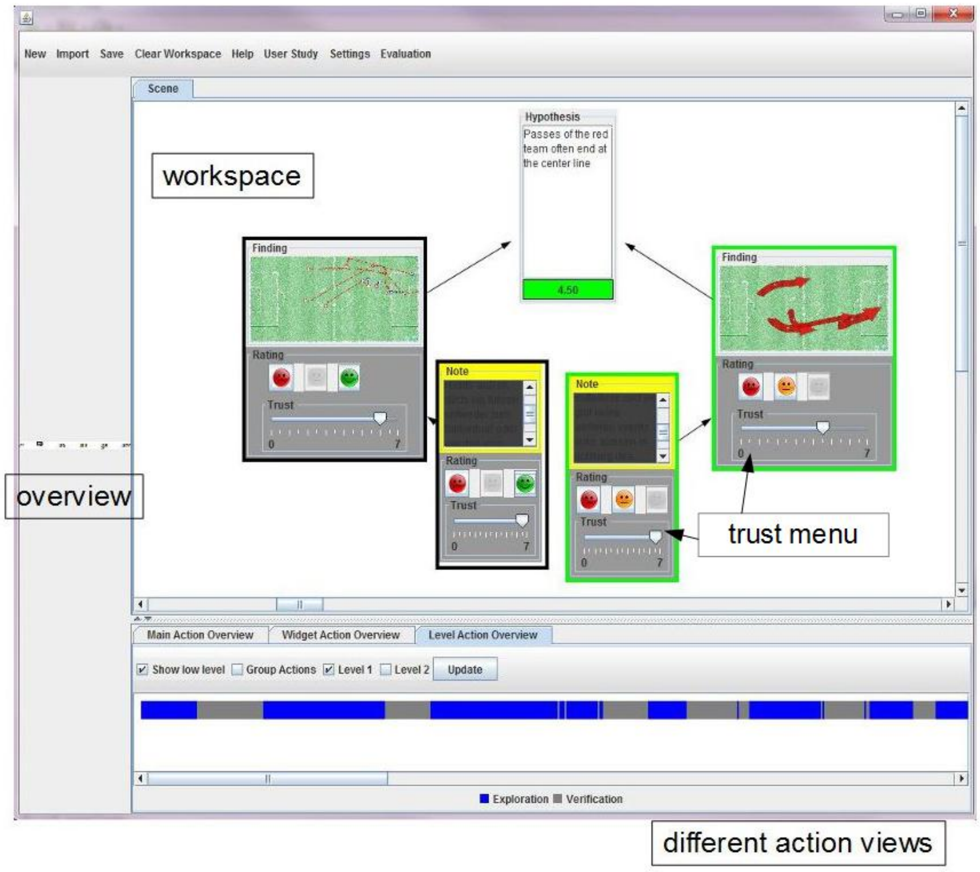


Figure 2: The NTE user interface composed of 3 main panels (workspace, interaction view, and overview).

2 NTE UI and Widget Description

Figure 2 shows the NTE user interface. It contains three main windows: 1) The workspace where analyst are building a knowledge graph, 2) The interaction visualizations in the bottom panel and 3) an overview component.

2.1 Menu Items

The menu bar on top offers several options/operations:

New: This menu provides options to create different widget types (Hypothesis, Notes, Insights).

Import: 1-File) An image file can be imported to the workspace (as a Finding widget). 2-XML) This allows to load previously stored knowledge graphs.

Save: Saves the current knowledge graph (the user has to enter a file name and choose a location).

Clear Workspace: Removes all items from all panels.

Help: Here different descriptions of the UI can be shown.

User Study: This option prepared our user study by adding the different hypothesis for task 1-6.

Settings: Capturing interactions can be turned off/on. Furthermore, the ration between uncertainty and trust can be changed (under development).

Evaluation: This option provides another pop up panel that shows the captured measures per widget or hypothesis for post analysis evaluation (under development).

2.2 Widgets

Creating, connecting and removing widgets: Widgets can be created by using the menu bar. Another possibility is to right click in the workspace. A context menu is shown providing the ability to chose a widget type that will be created. Similarly, right click on widgets will bring up another context menu for widget operations, such as deleting widgets. Furthermore, when a widget is hovered, it can be resized and repositioned.

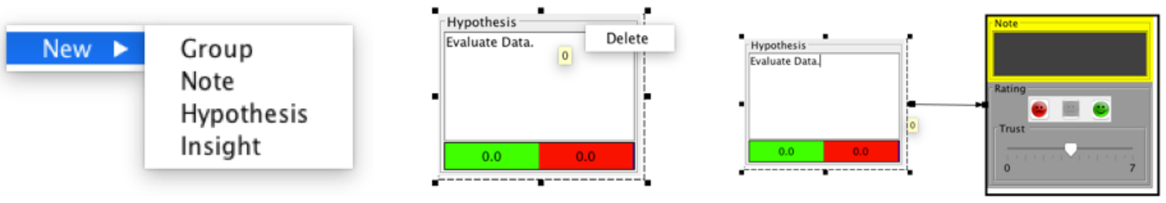


Figure 3: Widget operations.

Hypothesis: A hypothesis widget holds a text area where hypotheses, tasks or questions can be described. In addition, a trust bar is shown. It is composed of green and red bars, depending on the amount of positive/negative findings and their trust ratings.

Finding: Each finding widget is composed of an image (that has to be imported from hard disc or external VA system by using the API). In addition, a user can declare a widget as verifying, falsifying or neutral. Furthermore, a trust slider is present (in all widgets except hypothesis).

Note/Insight: These widgets are simple text widgtes for externalizing thoughts/comments (notes) or interpretations (insights).

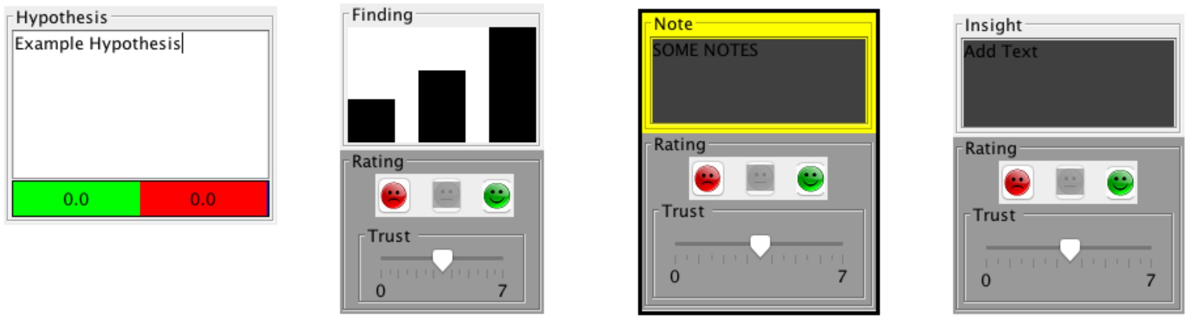


Figure 4: Different widget types (Hypothesis, Finding, Note, Insight)

Trust Slider : Each widget (except the hypothesis widget) contains a trust slider that allows the analyst to express their trust in it. The trust value is in range 1 to 7.

Pro/Contra/Neutral : Additionally, the user declares a widget as verifying, falsifying or neutral using the face icon-buttons. These information will be propagated to the hypothesis widget and visualized (bar).

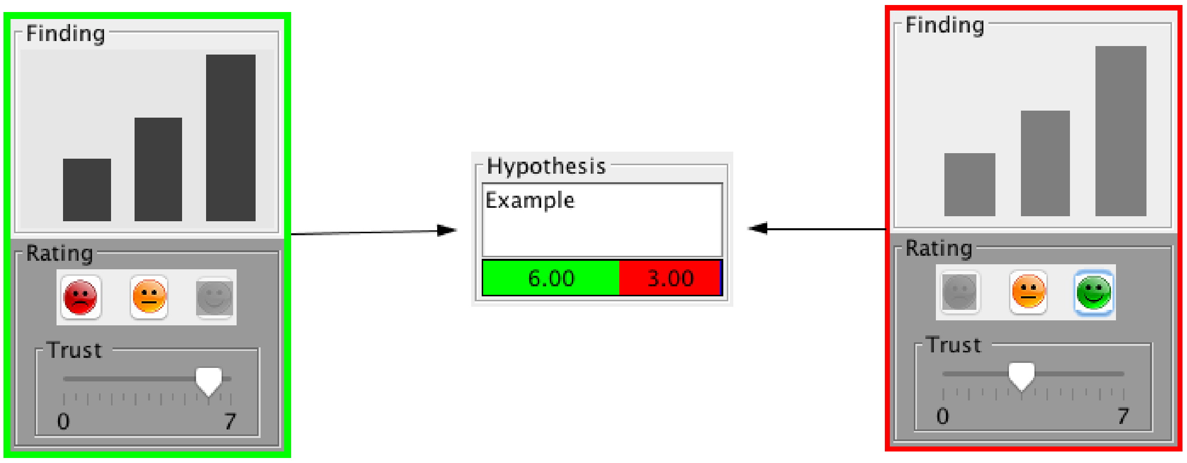


Figure 5: Trust ratings for pro and contra evidences will be aggregated for each hypothesis.

2.3 Interaction Views

The captured interaction data can be shown in different ways. Therefore, the interaction view offers different tabs:

Main Action Overview: The main action overview shows all the captured interactions. Grouped/similar actions are shown among each other. Further, the rectangle border encodes the action context (data, visualization, model) and the interaction level/phase is encoded as the rectangle background color (exploration-dark gray, verification-light gray).

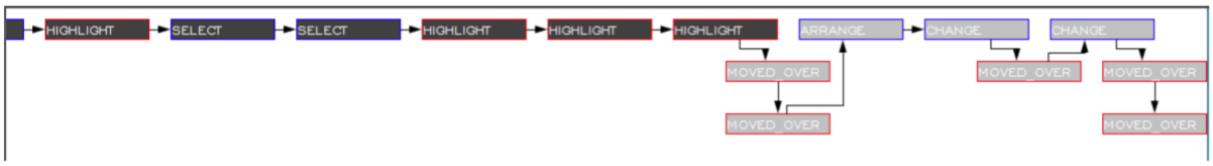


Figure 6: Main action overview

Widget Action Overview: The widget action overview shows the same information as the main action overview for a single selected widget. E.g., when finding is clicked, all exploration and verification interactions related to it are shown.

Level Action Overview: To create an overview of all the interactions (the main widget view only shows the details of current interactions) the level action overview provides a simplified and aggregated visualization. First, the interaction phases are visualized as bars. Blue bars denote the amount of exploration interactions (in a VA system) whereas gray bars denote the amount of interactions in the NTE. This view enables us to analyze and detect tool switches. In addition, more detailed rectangles for different interaction types can be shown underneath.

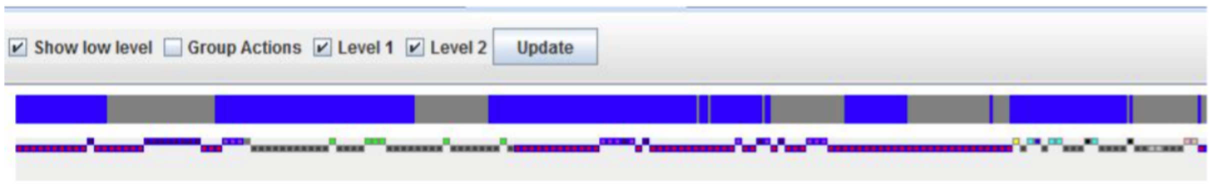


Figure 7: Level Action Overview, showing different phases (top bar) and detailed interaction (bottom rectangles).

3 API Description

The NTE is available as jar file and maven artifact. Please contact the authors (dominik.sacha@uni-konstanz.de) if you are interested in using it. In the long term, we plan to release the source code once we arrived at a stable version. However, the basic functionality is ready to be used as a research prototype.

3.1 Integrate the Note Taking Environment

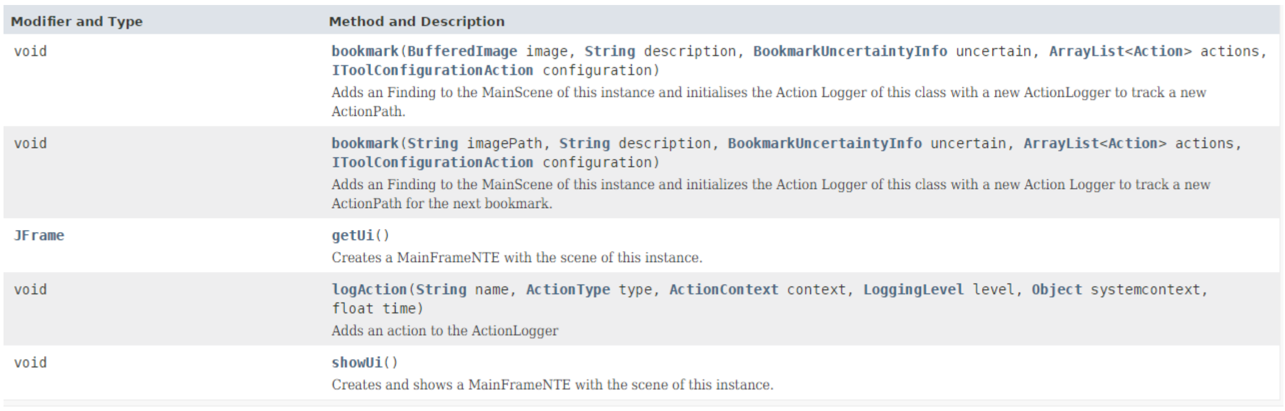
Through the Interface NoteTakingService the Note Taking Environment can be included in external system. If the external System is based on Maven the Note Taking Environment could be loaded to a Maven Repository and used as dependency with the following Maven Coordinates.

```xml
<dependency>
    <groupId>com.nt</groupId>
    <artifactId>NoteTakingEnvironment</artifactId>
    <version>1.0-SNAPSHOT</version>
</dependency>
```

If the external system is not based on Maven the Note Taking Environment can be included as a jar file.

3.2 Use the Note Taking Environment

3.2.1 NoteTakingService:



Modifier and Type	Method and Description
void	`bookmark(BufferedImage image, String description, BookmarkUncertaintyInfo uncertain, ArrayList<Action> actions, IToolConfigurationAction configuration)` Adds an Finding to the MainScene of this instance and initialises the Action Logger of this class with a new ActionLogger to track a new ActionPath.
void	`bookmark(String imagePath, String description, BookmarkUncertaintyInfo uncertain, ArrayList<Action> actions, IToolConfigurationAction configuration)` Adds an Finding to the MainScene of this instance and initializes the Action Logger of this class with a new Action Logger to track a new ActionPath for the next bookmark.
JFrame	`getUi()` Creates a MainFrameNTE with the scene of this instance.
void	`logAction(String name, ActionType type, ActionContext context, LoggingLevel level, Object systemcontext, float time)` Adds an action to the ActionLogger
void	`showUi()` Creates and shows a MainFrameNTE with the scene of this instance.

Figure 8: The NoteTakingService provides the shown functions.

3.2.2 Parameter description:

Function	Parameter	DataTyp	Description	null
bookmark	image	BufferedImage	BufferedImage of the visualization you want to bookmark	no
	description	String	Description of the bookmarked visualization.	yes
	uncertaintyValue	BookmarkUncertaintyInfo	See Section 'Uncertainty'	yes
	actions	ArrayList<Action>	List of actions you want to add to your bookmark.If its null the action list of the NTE is used	yes
	imagePath	String	Path to a visualization you want to bookmark	no
	configuration	IToolConfigurationAction	this object provides the possibility to add a funtion to a visualization. It will be performed if the user double-clicks on the image	yes
logAction	name	String	Name of the action	yes
	type	ActionType	Type of the action. Enums are defined in the NTE	no
	level	ActionLevel	LoggingLevel of the action Enums are defined in the NTE	yes
	context	Object	Object describing the current state of your system	yes
	timestamp	int	Timestamp of the action	yes

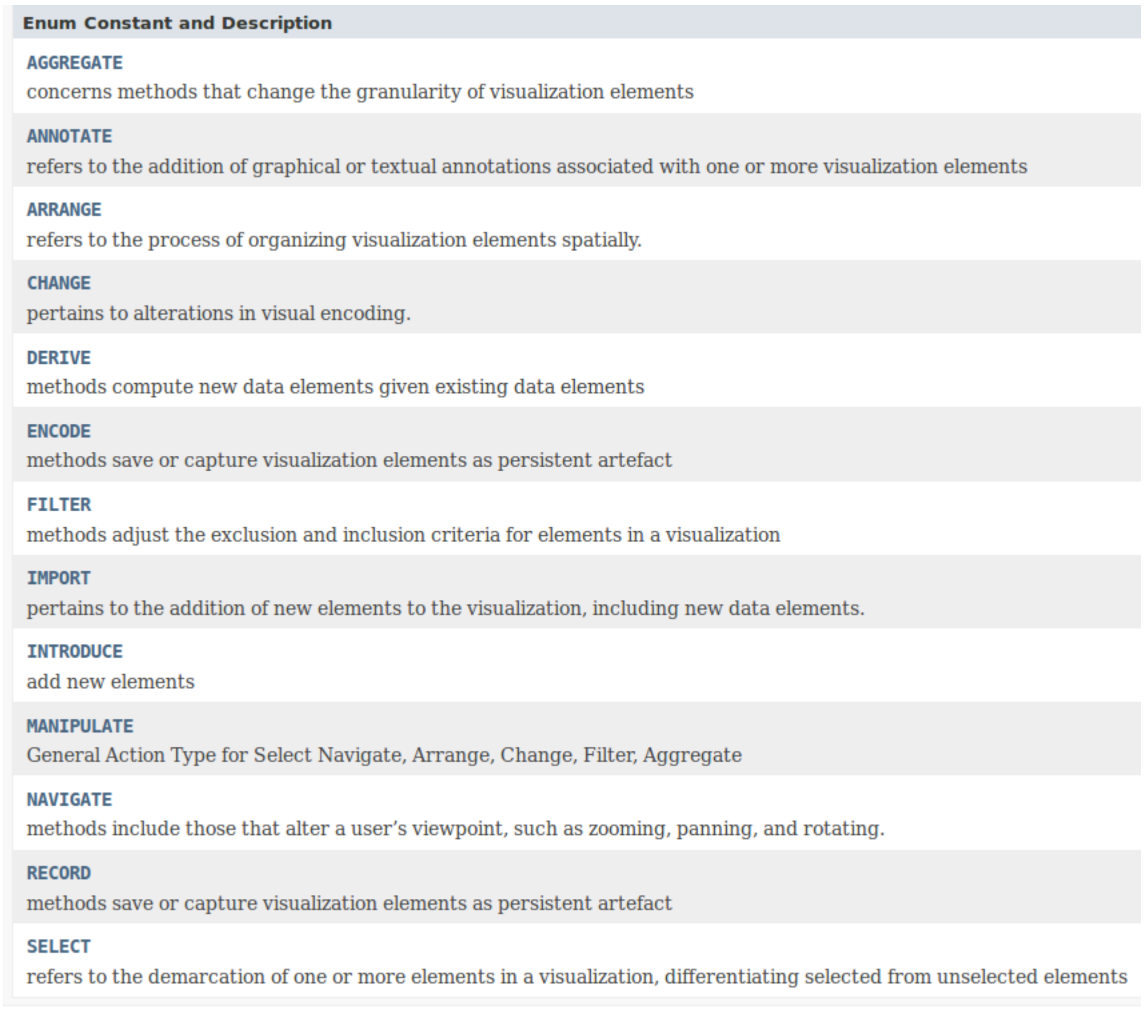
Action Logger:
Each Note Taking Service as an Action Logger, which manages the logged actions. If a user preferes to use another Action Logger, the actions can be added to a finding over the bookmark action. If the actionList of the bookmark action is null, the actionList of the Action Logger of the used instance is taken.
As the NoteTaking Service is implemented as Singelton the first step you need to do if you want to use a function of the interface is to get an instance.

```
        NoteTakingService service = NoteTakingService.getInstance();
```

3.3 Actions

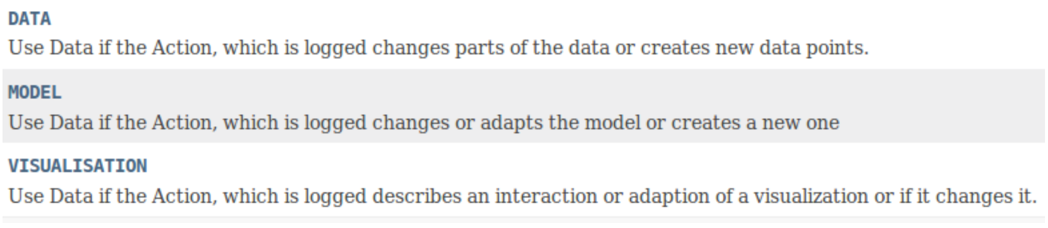
An action has a name, an ActionType, an ActionContext, a LoggingLevel, an Object describing the Systemstate and a timestamp. The name can be chosen by the user of the NTE, it is used to show the action in an Action View.

3.3.1 ActionType



Enum Constant and Description
AGGREGATE concerns methods that change the granularity of visualization elements
ANNOTATE refers to the addition of graphical or textual annotations associated with one or more visualization elements
ARRANGE refers to the process of organizing visualization elements spatially.
CHANGE pertains to alterations in visual encoding.
DERIVE methods compute new data elements given existing data elements
ENCODE methods save or capture visualization elements as persistent artefact
FILTER methods adjust the exclusion and inclusion criteria for elements in a visualization
IMPORT pertains to the addition of new elements to the visualization, including new data elements.
INTRODUCE add new elements
MANIPULATE General Action Type for Select Navigate, Arrange, Change, Filter, Aggregate
NAVIGATE methods include those that alter a user's viewpoint, such as zooming, panning, and rotating.
RECORD methods save or capture visualization elements as persistent artefact
SELECT refers to the demarcation of one or more elements in a visualization, differentiating selected from unselected elements

3.3.2 ActionContext



DATA Use Data if the Action, which is logged changes parts of the data or creates new data points.
MODEL Use Data if the Action, which is logged changes or adapts the model or creates a new one
VISUALISATION Use Data if the Action, which is logged describes an interaction or adaption of a visualization or if it changes it.

3.3.3 Logging levels:

are defined in the Note Taking Environment as well. Possible Levels are: EXPLORATION and VERIFICATION. For logging in the Visual Analytics system use EXPLORATION.

3.4 IToolConfiguration

This object provides to possibility to implement a method that will be in the analysis system, if the user double clicks on a finding in the Note Taking Environment. This method could for example recover the system state of a certain visualization.

3.5 Use NoteTakingService:

```
service.logAction("Select Player",ActionType.SELECT,
ActionContext.DATA, LoggingLevel.EXPLORATION, null ,timestamp);
```

3.5.1 Bookmark:

A bookmark can be made with either the image path or a BufferedImage. In the NoteTakingEnvironment a finding with the image and the given parameters will appear.

```
String imagePath = "D:/ExamplePath";
BufferedImage image =  ImageIO.read(new File(imagePath));
// create a BookmarkUncertaintyInfo Object and fill it
 with the uncertainty values for your bookmark
BookmarkUncertaintyInfo uncertain =
new BookmarkUncertaintyInfo(1,null,"add");
service.bookmark(imagePath, "description", uncertain, null );
service.bookmark(image, "description", uncertain, null );
```

3.5.2 User Interface:

To work with the UI of the Note Taking Environment the following to functions provide the possibility to get the UI or to show it:

```
service.getUI()
service.showUI();
```

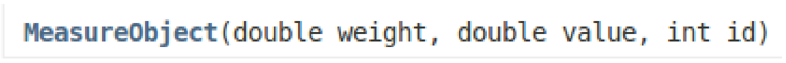
3.5.3 Callback Function:

To implement the possibility that users can perform a certain method with a double click on a visualization, implement this method in the perform() method of an IToolConfiguration implementation.

3.6 Uncertainty Information:

If your System works with one or more uncertainty calculations they can be added as additional information. Therefor, a BookmarkUncertaintyInfo object should be created.

3.6.1 MeasureObject:



```
MeasureObject(double weight, double value, int id)
```

This Class describes an Uncertainty value your program has calculated. Use this class if your program calculates more than one uncertainty value for one bookmark (e.g. one value for one data point). You can add a weight to the value for calculation of the actual value out of all uncertainty values.

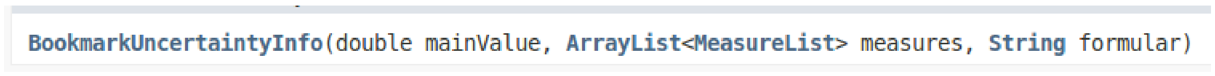
3.6.2 MeasureList:

```
MeasureList(String description, double mainValue, List<MeasureObject> measures, String formular)
```

This class contains a list of all uncertainty values for one bookmark, belonging to one type, like for example **data uncertainty**. The type can be chosen individually.

3.6.3 BookmarkUncertaintyInfo:

This is the main object for uncertainty information. Here you can declare the main value for your bookmark. If you have more than one uncertainty type this can be handled in this class as well, just add one MeasureList for each type.



```
BookmarkUncertaintyInfo(double mainValue, ArrayList<MeasureList> measures, String formular)
```

Parameter:

Name	Type	Description	Example
mainValue	double	Uncertainty Value calculated for this bookmark.	1
measures	ArrayList<MeasureList>	Contains Different MeasureLists with values for this Bookmark	×
formular	String	Formular with which the main value is calculated.	average, aggregate, sum, product

If your system calculates just one value for a bookmark create a BookmarkUncertaintyInfo object, add this value as mainValue and set the other parameters null.