**DATAVILIJTM**

**Software Design Description**

****

**Author:** Neil Opena

Professaur Inc. TM

March 2018

Version 1.0

**Abstract:** This document serves as the software design document for **DataViLiJ** (Data Visualization Library in Java), a desktop application that will allow users to select an algorithm (from a set of standard AI algorithms) and dynamically show the user what changes, and how.

**Based on the IEEE Std 830TM-1998 (R2009) document format**

© 2018 Professaur Inc, which is a made-up company. This document is fictitious and was created for the completion of a Software Design Description assignment for the course CSE 219 (Software Design and Development) at SUNY Stony Brook University.

*No part of this publication may be reproduced in any form, in an electronic retrieval system or otherwise, without the prior written permission of the publisher.*

**1. Table of Contents**

Software Design Description 1

Table of Contents2

1. Introduction3
   1. Purpose3
   2. Scope 3
   3. Definitions, acronyms, and abbreviations3
   4. References4
   5. Overview4
2. Package-level Design Viewpoint4
   1. Software OverviewFIXME
   2. Java API UsageFIXME
   3. Java API Usage descriptionsFIXME
3. Class-level Design ViewpointFIXME
4. Method-level Design ViewpointFIXME
5. File/Data structures and formatsFIXME
6. Supporting InformationFIXME

**1. Introduction**

Given the increasing importance of data-driven artificial intelligence (AI) in many aspects of computer science, visualizing how AI algorithms work is becoming increasingly important. Java is among the most important programming languages used to implement these algorithms, but it lacks standard data visualization libraries (unlike some other languages such as Python). Moreover, all existing libraries are meant to show us the final output of the data science algorithms. They are not designed for visualizing the changes that happen *while* the algorithms are running and updating the data. In other words, the visualization libraries do not help us see *how* these algorithms learn from the data.

**DataViLiJ** (Data Visualization Library in Java) will be a desktop application that will allow users to select an algorithm (from a set of standard AI algorithms) and dynamically show the user what changes, and how.

**1.1 Purpose**

The purpose of this document serves as the blueprint for how the DataViLiJ application will be constructed. It specifies the packages, classes, methods, variables, relationships, and interactions necessary to create the fully functioning data visualization desktop application by containing various UML class diagrams. UML Sequence diagrams represent the interactions of the classes for each scenario of a specified use case.

**1.2 Intended Audience**

The intended audience for this Software Design Description is the development team, including the instructor, Professor Eugene Stark, the teaching assistants of the course, and myself, the primary software design and developer.

**1.3 Product Scope**

The goal of this project is for students and beginning professionals in AI to have a visual understanding of the inner workings of the fundamental algorithms. AI is a vast field, and this project is limited to the visualization of two types of algorithms that “learn” from data. These two types are called **clustering** and **classification**. The design and development of these algorithms is outside the scope of the project, and the assumption is that such algorithms will already be developed independently, and their output will comply with the data format specified in this document. DataViLiJ serves simply as a visualization tool for how those algorithms work. Both clustering and classification are, in theory, not limited to a fixed number of labels for the data, but this project will be limited to at most four labels for clustering algorithms, and exactly two labels for classification algorithms. Further, the design and development of this project will also assume that the data is 2-dimensional. As such, 3D visualization is currently beyond the scope of DataViLiJ.

As for the GUI interactions, touch screen capabilities are not within the scope of this application.

**1.3 Definitions, acronyms, and abbreviations**

1. **Algorithm:** In this document, the term ‘algorithm’ will be used to denote an AI algorithm that can “learn” from some data and assign each data point a label.
2. **Clustering:** A type of AI algorithm that learns to assign labels to instances based purely on the spatial distribution of the data points.
3. **Classification:** A type of AI algorithm that learns to assign new labels to instances based on how older instances were labeled. These algorithms calculate geometric objects that divide the x-y plane into parts. E.g., if the geometric object is a circle, the two parts are the *inside* and the *outside* of that circle; if the geometric object is a straight-line, then again, there two parts, one on each side of the line.
4. **Framework:** An abstraction in which software providing generic functionality for a broad and common need can be selectively refined by additional user-written code, thus enabling the development of specific applications, or even additional frameworks. In an object-oriented environment, a framework consists of interfaces and abstract and concrete classes.
5. **Graphical User Interface (GUI):** An interface that allows users to interact with the application through visual indicators and controls. A GUI has a less intense learning curve for the user, compared to text-based command line interfaces. Typical controls and indicators include buttons, menus, check boxes, dialogs, etc.
6. **IEEE:** Institute of Electrical and Electronic Engineers, is a professional association founded in 1963. Its objectives are the educational and technical advancement of electrical and electronic engineering, telecommunications, computer engineering and allied disciplines.
7. **Instance:** A 2-dimensional data point comprising a *x*-value and a *y*-value. An instance always has a name, which serves as its unique identifier, but it may be labeled or unlabeled.
8. **Software Design Description(SDD):** A written description of a software product, that a software designer writes in order to give a software development team overall guidance to the architecture of the project. This document, for example, is a SDD.
9. **Software Requirements Specification(SRS):** A description of a software system to be developed. It lays out functional and non-functional requirements and may include a set of use cases that describe user interactions that the software must provide.
10. **Unified Modeling Language (UML):** A general-purpose, developmental modeling language to provide a standard way to visualize the design of a system.
11. **Use Case Diagram:** A UML format that represents the user’s interaction with the system and shows the relationship between the user and the different *use cases* in which the user is involved.
12. **User:** Someone who interacts with the DataViLiJ application via its GUI.
13. **User Interface (UI):** See *Graphical User Interface (GUI).*

**1.4 References**

1. IEEE Software Engineering Standards Committee. “IEEE Standard for Information Technology – Systems Design – Software Design Descriptions.” In *IEEE STD 1016-2009,* pp. 1-35, July 20, 2009
2. Bannerjee, Ritwik. *DataViLiJ TM Software Requirements Specification* Professaur Inc., 2018

**1.5 Overview**

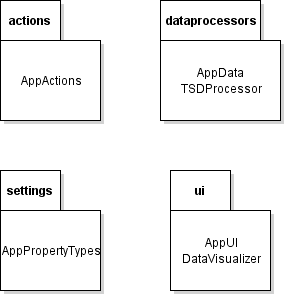
This Software Design Description (SDD) will includes design components that use UML to specify how to build the appropriate technologies for the operational capabilities of DataViLiJ and its UI functionalities and aesthetics, as described in the DataViLiJ Software Requirements Specification. Section 1 of this document includes the Introduction to the document itself and the References used. Section 2 provides the Package-level Design Viewpoint, specifying the packages and frameworks to be designed. Section 3 provides the Class-level Design Viewpoint, using UML Class Diagrams to specify how the classes should be constructed. Section 4 provides the Method-level Design Viewpoint, describing how methods will interact with one another. Section 5 provides deployment information like file and data structures and formats to use. Section 6 provides all the Supporting Information. All the UML Diagrams in this Software Design Description were created using the Violet UML Modeling tool.

**2. Package-level Design Viewpoint**

This design involves the construction of the DataViLiJ application. The Java API by Oracle is heavily relied upon as well the JavaX.XML API. The following sub sections describe how the components of the applications are to be constructed, including how the Java API will be used to build them.

**2.1 DataViLiJ Software Overview**

The DataViLiJ desktop application will be designed with the assistance of the XMLUtilities and ViLiJ frameworks. Fig 2.1. displays the components of the application with classes contained in packages.



**Fig 2.1.** DataViLiJ Package Overview (FIXME add inner classes maybe)

**2.2 Java API Usage**

The DataViLiJ application will be programmed using Java, therefore the Java API will be used extensively, the classes of which are specified in Fig 2.2.

**2.3 Java API Usage descriptions**