|  |
| --- |
| **SOEN 6471**  **ADVANCED SOFTWARE ARCHITECTURE**  **Summer 2023**  Deliverable 1  OPENARCH – Eclipse IDE  Declaration  We, the members of the team, have read and understood the Fairness Protocol and the Communal Work Protocol, and agree to abide by the policies therein, without any exception, under any circumstances, whatsoever.  **TEAM J**  A K M Saifun Nabi  Hema Reddy Muppidi  Karansinh Matroja  Mahek Akshaykumar Master  Piyush Singla |

**Table of Contents**

1. Introduction …………………………………………………………….. 3
   1. Uniquely Defining Characteristics …………………………………. 3
   2. Salient Software Activities …………………………………………. 4
2. Model for Context of Use for Eclipse IDE ……....................................... 4

2.1 Key Entities …………………………………………………………. 4

1. UML Class Diagram …………………………………………………….. 6
   1. Class Diagram ……………………………………………………….. 6
   2. Class Descriptions and Relationships ……………………………….. 6
2. Contributions Table ……………………………………………………… 8
3. References ………………………………………………………………... 8

6 GitHub ………………………………………………………………..….. 9

1. **Introduction**

Eclipse is a highly popular open-source Integrated Development Environment (IDE) known for its extensive features, flexibility, and large community support. It offers a customizable interface, project management tools, version control integration, and a plugin architecture that allows developers to extend and tailor its functionality. Eclipse's primary focus is on Java programming, providing automatic code completion, refactoring, debugging, and other essential tools for efficient software development [1].

* 1. **Uniquely Defining Characteristics**

One of the most defining characteristics of Eclipse is its plugin-based architecture. This modular design enables developers to enhance and customize the IDE by adding plugins that provide additional features, tools, and language support. This extensibility allows developers to adapt Eclipse to their specific needs and work seamlessly with different programming languages and frameworks [2].

In addition to its plugin architecture, Eclipse stands out for its wide-ranging language and domain support. It offers robust tooling and support for languages such as Java, C/C++, Python, PHP, JavaScript, and more. Furthermore, Eclipse provides specialized IDEs and frameworks tailored to specific domains, such as Eclipse for Java Developers, Eclipse for PHP Developers, and Eclipse for Android Development. This comprehensive language and domain support make Eclipse a versatile choice for developers working across various technology stacks [3].

Eclipse integrates a comprehensive set of development tools that cover different stages of the software development lifecycle. It includes powerful code editing features, sophisticated debugging capabilities, testing frameworks, profiling tools, and integration with popular version control systems. The IDE's seamless integration of these tools enhances productivity and streamlines the development process [1].

Another distinguishing characteristic of Eclipse is its cross-platform compatibility. It is designed to run on multiple operating systems, including Windows, macOS, and Linux, ensuring that developers can work with Eclipse on their preferred platform without sacrificing functionality or user experience [4].

* 1. **Salient Software Activities**

Eclipse boasts a thriving ecosystem supported by a large and active community of developers. The Eclipse Marketplace serves as a central hub for discovering and installing a wide range of plugins, frameworks, and libraries contributed by the community [5]. This ecosystem offers developers an extensive selection of resources and extensions to enhance their Eclipse experience and meet their specific development needs.

Furthermore, Eclipse fosters collaboration and community engagement. It provides infrastructure for collaborative development, including version control system integration, issue tracking, and code review tools. The Eclipse Foundation, a not-for-profit organization, oversees the governance and evolution of Eclipse, encouraging community contributions and promoting innovation within the ecosystem [5].

In conclusion, Eclipse's uniquely defining characteristics, including its plugin-based architecture, extensive language support, integrated development tools, cross-platform compatibility, vibrant ecosystem, and community engagement, make it a preferred choice for developers worldwide. These qualities contribute to its widespread adoption and position it as a leading IDE in the software development industry.

1. **Model for context of use for Eclipse IDE:**

The Eclipse Integrated Development Environment (IDE) is a powerful and widely used tool for software development. It contains a base workspace and an extensible plug-in system for customizing the environment. It provides a range of features and capabilities that make it suitable for various contexts and purposes.

**2.1** **Following are the key entities within this context:**

* **Users:** Eclipse IDE serves a wide spectrum of users, including teams of software professionals, individuals working on their own projects, and students. It provides a user-friendly setting with features that may be adjusted to suit various programming needs and ability levels.
* **Projects:** Users of the Eclipse IDE can create, manage, and work on numerous projects at once. Users can import current projects into Eclipse or start fresh projects from scratch. Each project has essential configuration files, source code files, and development resources.
* **Package Explorer:** A key component of the Eclipse IDE that offers a hierarchical view of the project's structure is the Package Explorer. It provides a tree-like organization of files and folders and the package hierarchy for Java projects. Developers can view and edit files within their projects by navigating Package Explorer.
* **Files (Java, XML) :** Java and XML files are among the many file types supported by the Eclipse IDE. While XML files are frequently used for configuration or data storage, Java files are the source code produced in the Java programming language. For these file types, Eclipse offers syntax highlighting, code completion, and error checking to aid developers in writing and managing code effectively.

Java development is well supported by the Eclipse IDE, which is well renowned for this. For Java projects specifically, it offers features like code auto-completion, debugging tools, and project management capabilities. For Java programming, a lot of programmers utilize the Eclipse IDE as their primary environment.

* **Activities:** The software development lifecycle is facilitated by the use of the Eclipse IDE for a variety of tasks. Users may work with version control tools like Git, rework code, manage dependencies, and create, edit, and delete source code files. In order to make it simple for users to develop and package their projects, Eclipse also interfaces with build automation technologies like Apache Maven or Gradle.
* **Console:** Users of the Eclipse IDE can inspect the output of their apps or engage in interactive debugging on the console. During programme execution, the console shows the system output, error warnings, and other pertinent data. It is especially helpful for troubleshooting, logging, and comprehending the behaviour of applications during runtime.
* **Server:** Users may deploy and test their apps straight from the Eclipse IDE thanks to connectivity with a number of application servers. Users can deploy projects to the server for execution, manage server instances, and modify server parameters. The development and testing process is streamlined by this functionality, particularly for web and corporate applications.
* **Debug and Compile:** An effective debugging environment is offered by Eclipse IDE for finding and correcting software issues. Users have the ability to set breakpoints, examine variables, walk through code, and watch the program's flow during execution. Eclipse also provides project compilation and building, ensuring that the code is converted into executable form and producing the required

artefacts for deployment.

1. **Stakeholder model of Open Arch:**

**3.1 Diagram:**

A picture containing text, diagram, plan, parallel

Description automatically generated

*Figure 3.1.1: Stakeholder model of Eclipse IDE [UML Class Diagram]*

**3.2 Below are the descriptions of each class of the diagram along with their relationship:**

* **Eclipse IDE:** This is the main class of the diagram. It has workspace, activeWorkspace, activeEditors, searchFile attributes. All of the attributes are of List type. This class has also got methods like openWorkspace, closeWorkspace, saveWorkspace, switchWorkspace, openEditor, closeEditor, searchFile. It has an aggregation relationship with Workspace class which is represented by the attribute workspaces. An Eclipse IDE can have list of workspaces. It also has composition relationship with the Search and Editor class. An IDE can have multiple active editors and list of files can be searched. These relationships are represented by the activeEditor and searchFile attributes respectively.
* **Workspace:** An Eclipse IDE can have many workspace. It has got a list of projects with separate naming. It has methods to get the name of the workspace and also the list of projects under each workspace. It has got aggregation relationship with the Project class which is represented by the attribute projects.
* **Project:** A workspace can have multiple projects. It has got attributes like projectName, location of the source code of the project and build configuration. It has got methods getName() to retrieve the name of the project, getSourceLocation() to get the list of folders for the project files and getBuildConfiguration() for the runtime configuration of the project. It has aggregation relationship with the Folder class which is represented by the attribute sourceLocation and composition relationship with the BuildConfiguration class which is represented by the buildConfiguration attribute.
* **Folder:** All the project related files are kept in a folder. There are two attributes of this class which are name and files. getName() method is used to retrieve the name of the folder and getFiles() method is implemented to get the list of files under that folder. It has got aggregation relationship with the File class which is denoted by the files attribute.
* **Editor:** It depicts the main editor panel in an Eclipse IDE. It’s main two main attributes are file and content. getFile() are used to open the file, getContent() method is used to retrieve the content of that file, setContent() method is used to set the content and finally save() method are implemented to save the file in that editor. It has got association relationship with the File class which is delineated by the file attribute.
* **File:** It’s the basic file method to retrieve and save the file in an editor. Name and content are two attributes of this class. getName(), getContent() and setContent() are used for various implementation of the File. It has got association relationship with the Editor class which is represented by the file attribute.
* **Search:** This class is used to search the files in an EclipseIDE by the text. searchFile and listOfFiles are two of it’s attribute. validateText() method is used to check the correctness of the text used for searching and searchByFile() method returns the list of files after search. It has got aggregation relationship with the File class which is represented by the listOfFiles attribute.
* **BuildConfiguration:** This is the configuration class for each project. In EclipseIDE each class may have different configurations. It has key, value pair mapping to implement the configuration for various options.

1. **Table of Contributions:**

|  |  |
| --- | --- |
| **Team J** | **Contributions** |
| 1. **Piyush Singla** | * Researched and reached to a choice on the OPENARCH model. * Compiled pertinent data regarding Eclipse Architecture. * Worked on UML Class Diagram with Saifun * Documented Problem 3 with the team-collaboration |
| 1. **Mahek Akshaykumar Master** | * Researched and reached to a choice on the OPENARCH model. * Compiled pertinent data regarding Eclipse Architecture. * Worked on Introduction and Description * Documented Problem 1 and helped in editing the final version with the team-collaboration. |
| 1. **Karansinh Matroja** | * Researched and reached to a choice on the OPENARCH model. * Compiled pertinent data regarding Eclipse Architecture. * Worked on the Model for Context of Use with Hema * Documented Problem 2 with the team-collaboration |
| 1. **Hema Reddy Muppidi** | * Researched and reached to a choice on the OPENARCH model. * Compiled pertinent data regarding Eclipse Architecture. * Worked on the Model for Context of Use along with Karan * Documented Problem 2 with the team-collaboration |
| 1. **A K M Saifun Nabi** | * Researched and reached to a choice on the OPENARCH model. * Compiled pertinent data regarding Eclipse Architecture. * Worked on UML Class Diagram with Piyush * Documented Problem 3 and helped in editing the final version with the team-collaboration. |

1. **References:**

[1] “Best IDE Software - A List of the Top 10 - KeyCDN,” *KeyCDN*. <https://www.keycdn.com/blog/best-ide>

[2] “Notes on the Eclipse Plug-in Architecture,” *Notes on the Eclipse Plug-in Architecture*. <https://www.eclipse.org/articles/Article-Plug-in-architecture/plugin_architecture.html>

[3] M. Soni, “Eclipse IDE: An Overview,” *Open Source For You*, May 08, 2023. <https://www.opensourceforu.com/2023/05/eclipse-ide-an-overview/>

[4] B. Refi, “14 Java IDEs and Text Editors – Complete Guide,” *Bluebird*, Apr. 08, 2023. <https://bluebird-europe.com/java-ides-and-text-editors/>

[5] T. Team, “14 Useful (and Free) Plugins for Eclipse IDE,” *The Official Tabnine Blog*, Feb. 06, 2021. <https://www.tabnine.com/blog/plugins-for-eclipse/>

[6] [https://eclipseide.org](https://eclipseide.org/)

**6. Github Repo Link :**

<https://github.com/singlapiyush1/SOEN_6471_GroupJ_Project>