OPENARCH - ECLIPSE

SOEN 6741 - Advanced Software Architecture

Team A: Duy Thanh Phan, Neha Sanjay Deshmukh, Dhruvi Patel, Mutasimur Rahman

Concordia University – Gina Cody School of Engineering and Computer Science

1. INTRODUCTION

Eclipse is a powerful and versatile open-source Integrated Development Environment (IDE) that provides a comprehensive set of tools for software development across multiple programming languages and platforms.

2. DESIRABLE TRAITS

1. **Layered Architecture:**

Eclipse may employ a layered architecture, where different components and services are organized into layers, each responsible for specific functionalities.

2. **Separation of Concerns (SoC) Principle:**

Eclipse emphasizes the separation of concerns, dividing the software system into distinct modules or plugins.

3. **Dependency Injection and Inversion of Control:**

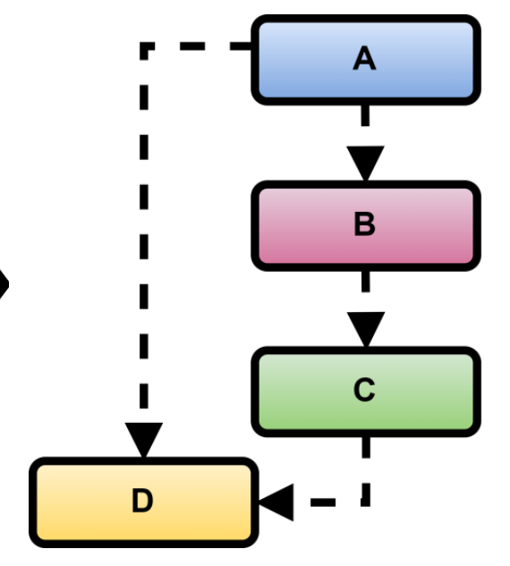
These patterns facilitate loose coupling, testability, and flexibility in managing component dependencies.

4. **Event-Driven Architecture:**

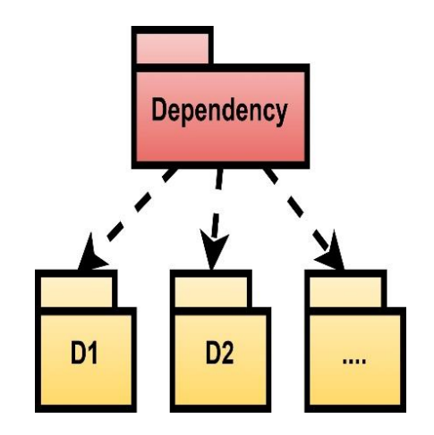
Eclipse follows an event-driven architecture, where various components and plugins can subscribe to and emit events.

5. **Acyclic Dependency Principle (ADP):**

In Eclipse, the dependencies between various components or modules are structured in such a way that no module depends on itself, directly or indirectly.



6. **Static Dependency Principle (SDP):**



Most components or modules in Eclipse are designed to handle a specific responsibility, which aids in maintaining and evolving the system.

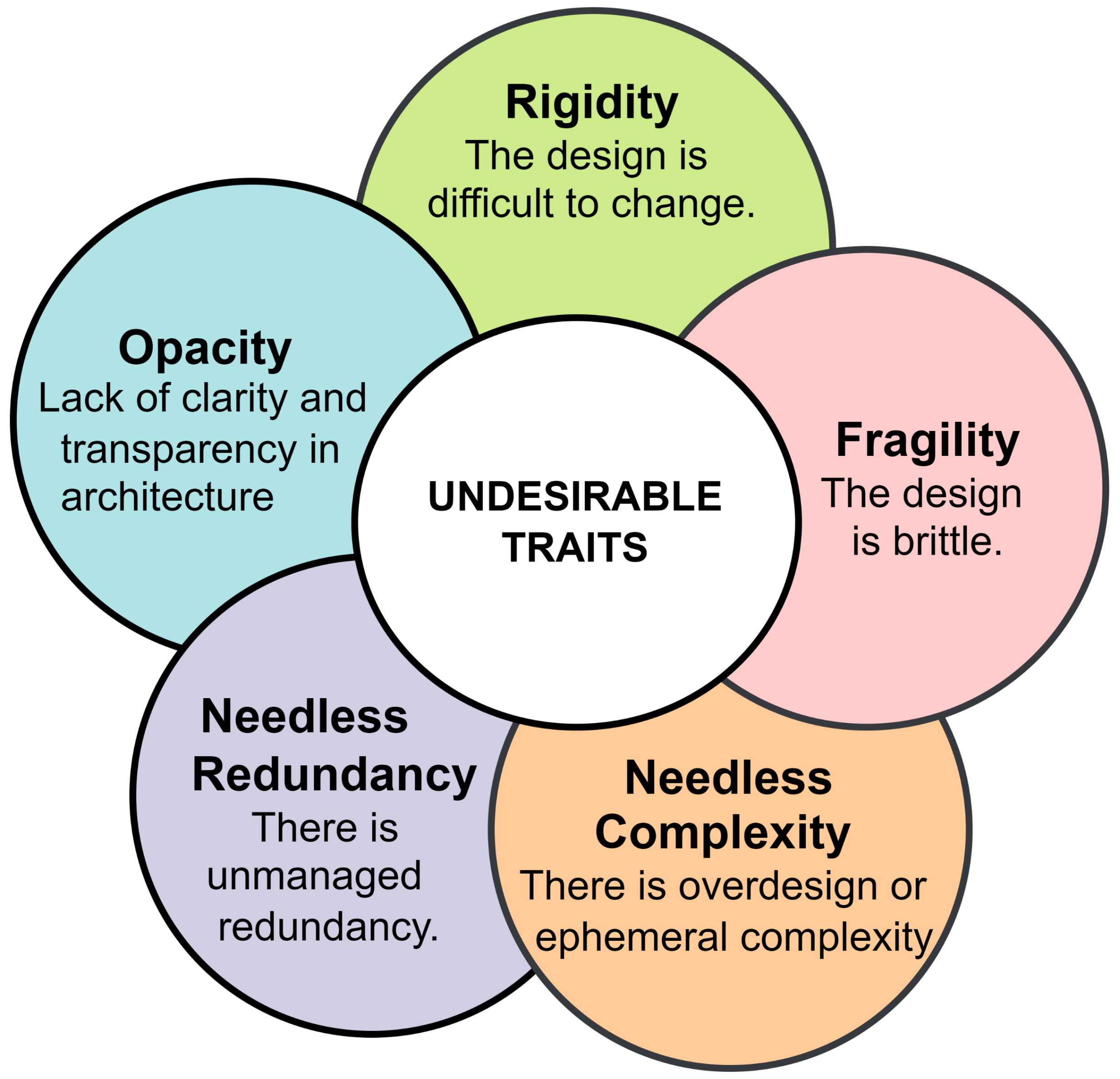
7. **Open-Closed Principle (OCP):**

This principle enable developers to extend functionality of Eclipse without modifying its existing codebase, ensuring stability, maintainability, and flexibility.

Existing Methods

New Methods

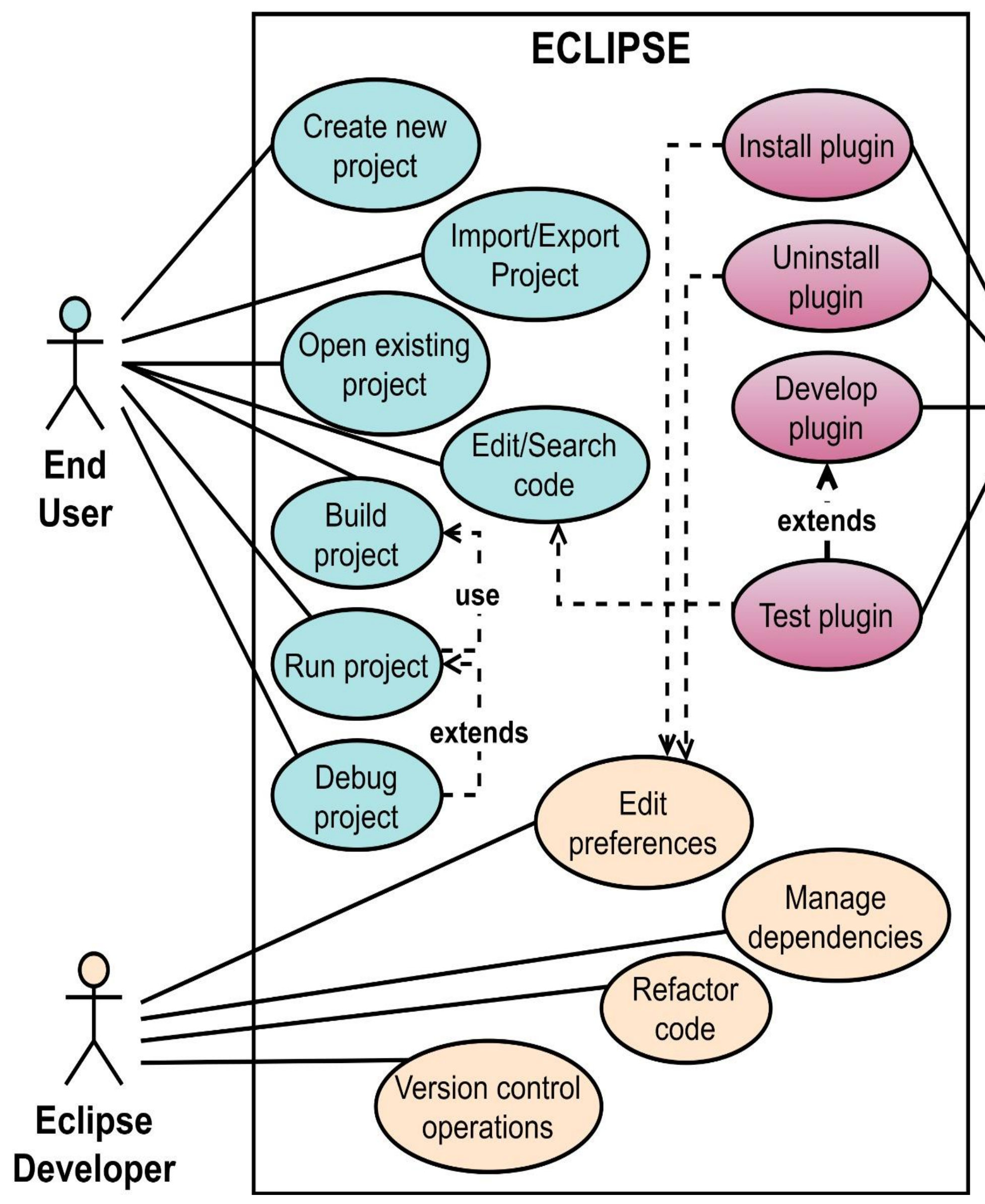
3. UNDESIRABLE TRAITS


UNDESIRABLE TRAITS
Rigidity
The design is difficult to change.
Fragility
The design is brittle.
Needless Complexity
There is overdesign or ephemeral complexity.
Needless Redundancy
There is unmanaged redundancy.
Opacity
Lack of clarity and transparency in architecture

3.1. USES OF UNDESIRABLE TRAITS

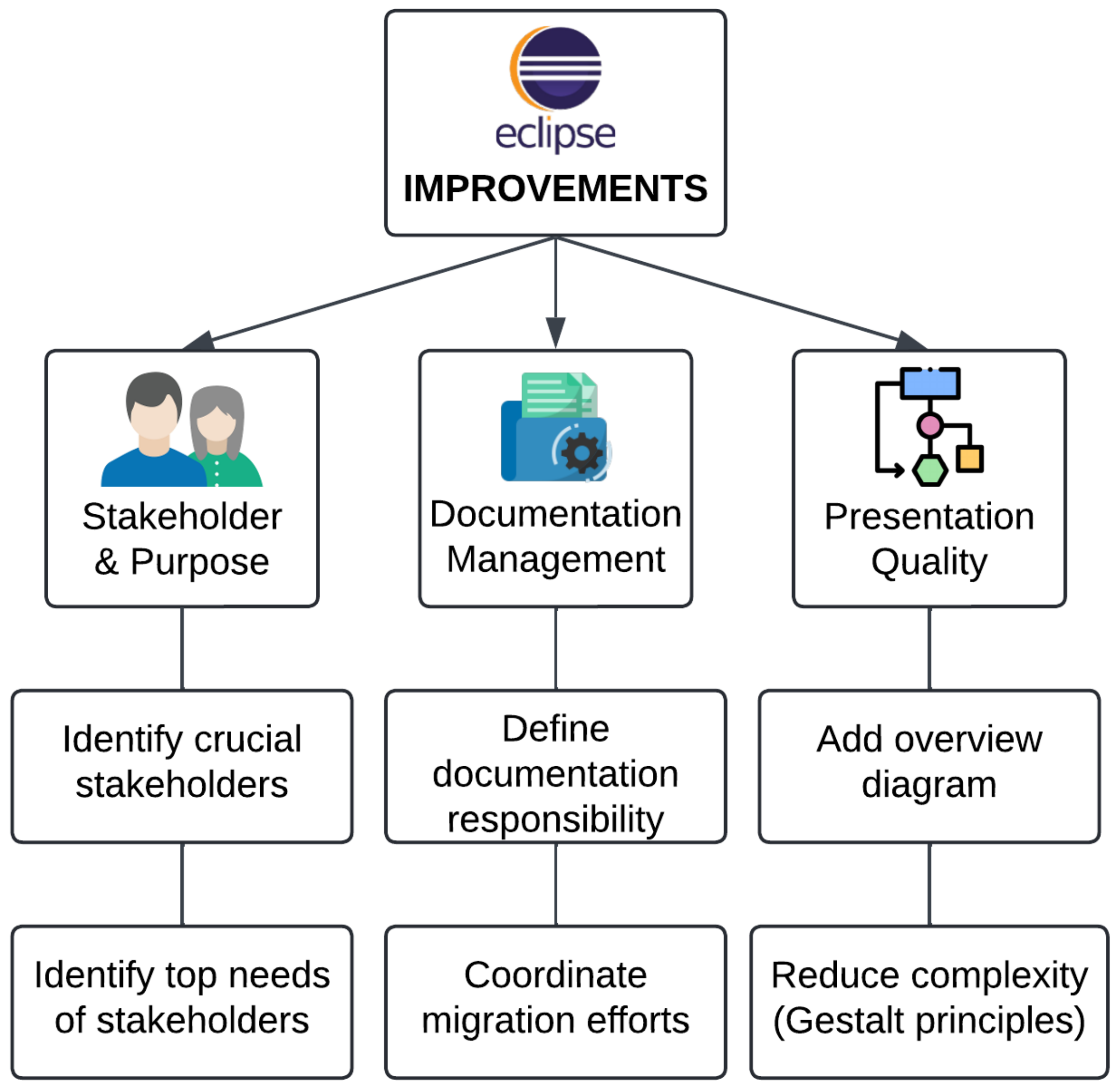
- IDENTIFICATION OF DESIGN ISSUES
- MAINTENANCE & EVOLUTION
- QUALITY ASSURANCE
- SCALABILITY
- COLLABORATION
- LEARNING & GROWTH

4. SOME ACTORS AND USE CASES



Actor	Role	Interactions
End User	Primary users such as Software Developers and QA Testers	**Project management:** Create, open, import/export projects **Code handling:** Edit and search **Development process:** Build, run, and debug projects
Plugin Developer	Specialized in developing and maintaining Eclipse plugins	**Plugin lifecycle:** Develop, test, install, and uninstall plugins
Eclipse IDE Developer	Maintains and develops the Eclipse IDE codebase	**Codebase management:** Develop and refactor code **System settings:** Manage version control and edit preferences

5. IMPROVEMENTS


eclipse
IMPROVEMENTS
Stakeholder & Purpose
Identify crucial stakeholders
Identify top needs of stakeholders
Documentation Management
Define documentation responsibility
Coordinate migration efforts
Presentation Quality
Add overview diagram
Reduce complexity (Gestalt principles)

6. LESSONS LEARNED

(1) Make sure to have concrete references for every piece of information in the presentation. **(2)** Ensure every presentation has a complete agenda, with an introduction and a conclusion. **(3)** Frequent communication and self-discipline are key to the project's success.

7. CONCLUSIONS

The Eclipse architecture embodies a robust, modular, and extensible framework that has revolutionized the development landscape. Its structure allows high flexibility, satisfying advanced use cases for different stakeholders. More architectural work is needed to enable maintenance and development in Eclipse with high satisfaction.

8. REFERENCES

[1] "Wiki shutdown plan · Wiki · Eclipse Foundation / Help Desk · GitLab," GitLab. Accessed: Jun. 09, 2024. [Online]. Available: https://gitlab.eclipse.org/eclipsefdn/helpdesk/-/wikis/Wiki-shutdown-plan

[2] Wermelinger, M., Yu, Y., Lozano, A. et al. Assessing architectural evolution: a case study. Empir Software Eng 16, 623–666 (2011). https://doi.org/10.1007/s10664-011-9164-x

[3] P. Kamthan, THE 'UNDESIRABLES' OF SOFTWARE ARCHITECTURE

[4] D. Todorovic, "Gestalt principles," Scholarpedia, vol. 3, no. 12, p. 5345, 2008, doi: 10.4249/scholarpedia.5345.

[5] "IBM Host On-Demand 15.0.0." Accessed: Jun. 01, 2024. [Online]. Available: https://www.ibm.com/docs/en/host-on-demand/15.0?topic=support-creating-host-demand-plug-ins.