Project - Pattern

Presented by:

Daniel Libardo Diaz Gonzalez - *dandiazgo@unal.edu.co*

Andrés Felipe León Sánchez - *anleonsa@unal.edu.co*

Alejandro Medina Rojas - *alemedinaro@unal.edu.co*

Angel David Ruiz Barbosa - *aruizba@unal.edu.co*

Professor:

Oscar Eduaro Alvarez Rodriguez

oalvarezr@unal.edu.co

Friday 11th of July

Universidad Nacional de Colombia

Facultad de Ingeniería

Ingeniería de Sistemas y Computación

2025

CONTENT

1. Design Patterns

1.1. Component based architecture: The component based architecture is a software design pattern that focuses on modularity and reusability. It does the aforementioned by building a set of components independent from one another, which interact with each other via a system that normally resembles an interface. These components are designed to be reused across different projects, and they can be replaced or redesigned without affecting its counterparts' behaviour. The communication between the elements can be synchronous or asynchronous

1.2. Observer: The observer design pattern defines a one-directional relation between objects. When an object changes state, all of its observers are notified, which triggers an event on the notified object. This design ensures that all the components are properly interconnected and synchronized.

2. Implementation

2.1. Component based architecture: We implement the component based architecture by working with the Qt classes, specially in the development of the GUI. Qt works with a Meta-Object system in which all of the objects inherit the properties of QtObject that allows them to perform the necessary communications between objects as well as object trees for managing dependencies. This makes possible the modularity mentioned before, by permitting each component to work on its own, meaning they implement their own logic and functionality; while any action, like and the occurrence of an event or the modification of a related component is managed by the Meta-Object system and not by each individual component.

2.2. Observer: Qt's Meta-Object systems also come with techniques called signals and slots, which make use of the observer design pattern. This tool allows the connection between one dependent object and the object to which it is dependent, these are structured as the son and parent node in the object tree respectively. This connection permits emitting signals between them, that later triggers an event, also called by Qt as a slot,

3. Justification

3.1. Component based architecture: The usage of a component based architecture for the development of an IDE not only is a good choice but basically a necessity, as the roots of this design pattern resembles the way in which an IDE is built, that at the end of the day is a complex system with different features such as the editor space, the toolbar, menus and other tools that work separately but interconnects with one another. These components also

need to be maintained, tested and scaled on their own, an objective that can easily be achieved by working with the component based architecture that Qt offers.

3.2. Observer: An IDE is a software with a lot of tools, so it would consume a lot of memory and CPU capacity to follow the traditional polling method, in which a system is constantly checking for an update in the current status of an object. Also, this feature facilitates even further the capacity of the modularity provided by the component based architecture, as it enables the creation of new components and their connection to the already existing ones without refactoring the previous ones.