THE COMPLETER FRAMEWORK FOR ASYNCHRONOUS OPERATIONS

SAAD BIN KHALID (20K-0161)
BILAL AHMED KHAN (20K-0183)
MUHAMMAD AHMED AHSAN (20K-0343)

```
mod = modifier_ob.modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("modifiers.mos("mod
               mod_mirror_object = mirror_co
               Atlon -- "MIRROR_X":
              mod.uso x - True
              mod.use_y = False
                 mod.use z - False
                 Fation - "MIRROR Y"
                      mod.use x = False
                     mod.use y - True
                      mod.uso z - False
                      wtion -- "HIRROR Z":
                        mod.use_x = False
                   mod.use y - False
                      mod_use z = True
                      tion at the end add back the desels
                     b.select-1
                       *.scene.objects.active = modifier at
                      ected" + str(modifier ob)) # modifies
                           ob.select = 0
              context.selected_objects[0]
                        objects[one.name].select = 1
                  Please select exactly two objects,
                 OPERATOR CLASSES -----
                  mirror to the selected object****
                    *.mirror_mirror_x*
```

INTRODUCTION

- Asynchronous operations are crucial in modern software.
- Existing methods (callbacks, promises) often lead to complexity and errors.
- The Completer Framework aims to simplify asynchronous development.

PROBLEM STATEMENT

- Managing asynchronous tasks, coordination, and error handling can be very complex.
- Existing approaches are not well-suited for intricate scenarios.



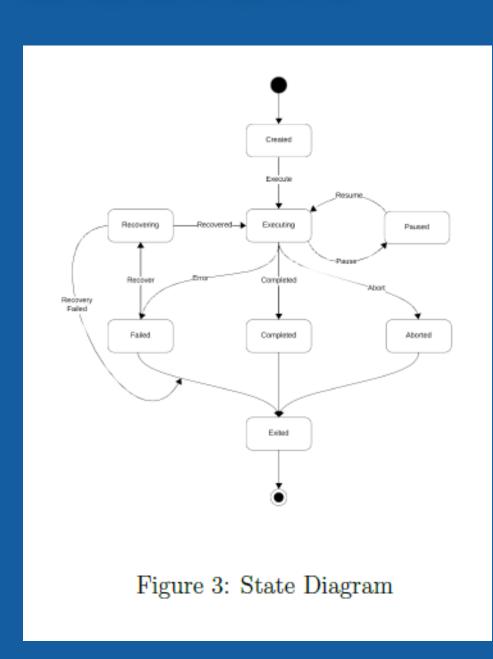
RESEARCH QUESTIONS

- Can we create a unified framework to streamline asynchronous development?
- How do we design the framework to be robust and errorresistant?
- How can we ensure adaptability across different platforms and environments?

RESEARCH OBJECTIVES

- Create "The Completer Framework"
- Simplify development compared to existing methods
- Incorporate robust error handling
- Design for cross-platform adaptability

THE COMPLETER FRAMEWORK: DESIGN



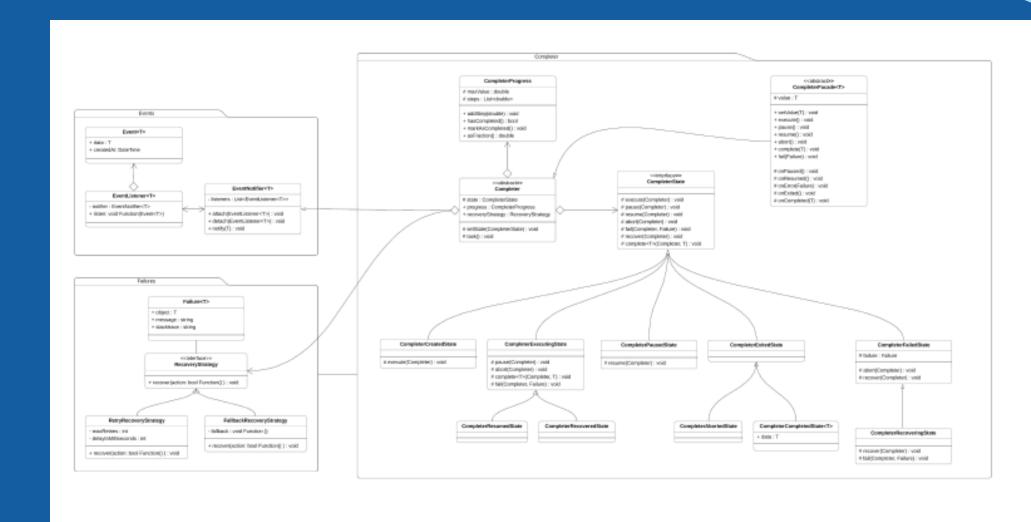


Figure 4: Class Diagram

THE COMPLETER FRAMEWORK: DESIGN PATTERNS USED

- STATE DESIGN PATTERN
- OBSERVER PATTERN
- COMMAND PATTERN
- FACADE PATTERN

CASE STUDIES

• E-commerce Order Processing

 The Completer Framework can be integrated into the order processing system to manage the asynchronous tasks associated with order fulfillment. Each order's lifecycle can be represented by different states within the framework

Data Pipeline for Analytics Platform

 The Completer Framework serves as the backbone of the data pipeline, managingthe asynchronous tasks involved in data ingestion, processing, transformation, and analysis. Each stage of the data pipeline is represented as a state within the framework.

CONCLUSION

- The Completer Framework leverages design patterns for effective asynchronous task management
- The framework simplifies development, enhances robustness, and provides adaptability
- It has the potential to streamline the development of modern software applications