**Software Architecture and Design**

**Names: Tommie Walker**

**Esther Nyarangi**

**Group Assignment 2**

**The Process Architecture**

The implementation and development of this program and corresponding architecture was obtained through a breakdown of requirements, then an assessment of necessities. After figuring what we had an obligation to deliver, we needed to figure out tools or objects we needed to implement to get to a solid resolution.

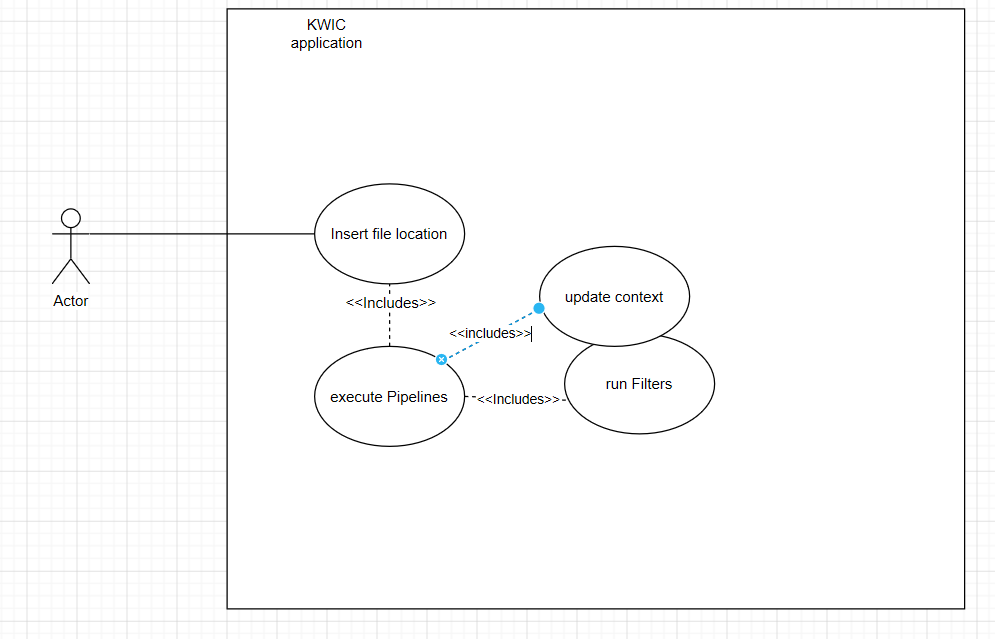
The assignment was meant to be done as a group and work split evenly. However, to get to a solid end product, Tommie had to formulate and implement the architecture used in this assignment, while Esther was to complete the graphs and charts. Upon the last meeting we will conduct a final review. The final review resulted in Tommie assisting Esther reconstruct and alter graphs. We communicated with each other through text messages, WhatsApp, and Zoom video calls. Code has been notated and made accessible through GitHub. Progress reports were done twice at least week while in production.

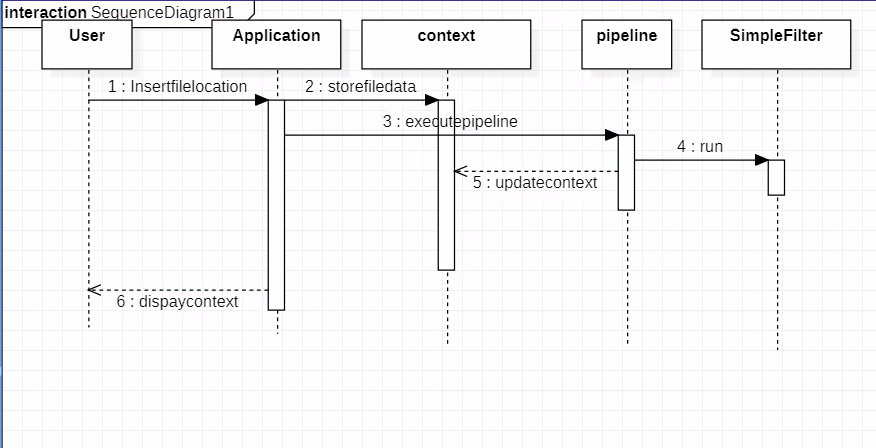
**The 4+1 view model**

It concentrates on four main views: logical, process, development and physical. This model was used to implement our various implementations.

**The Requirement Specification:**

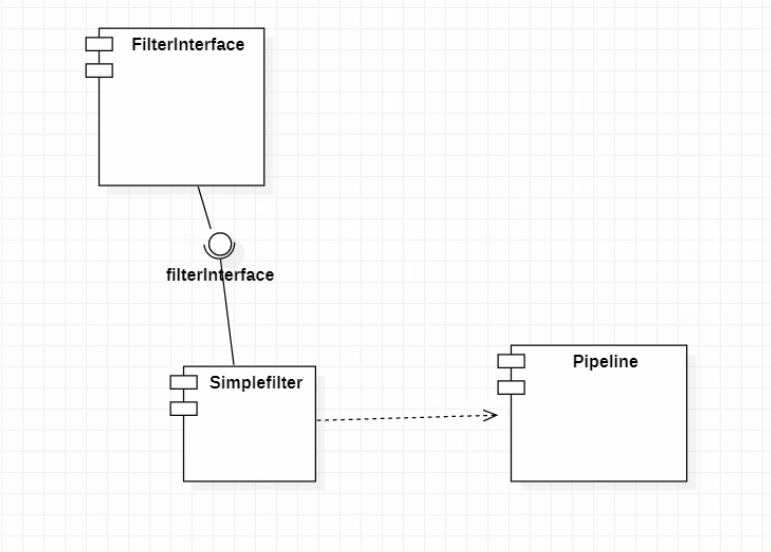
We incorporated the Use Case diagram in implementing this design to show the functional requirements of the system. We also used the sequence diagram to show the process view respectively.



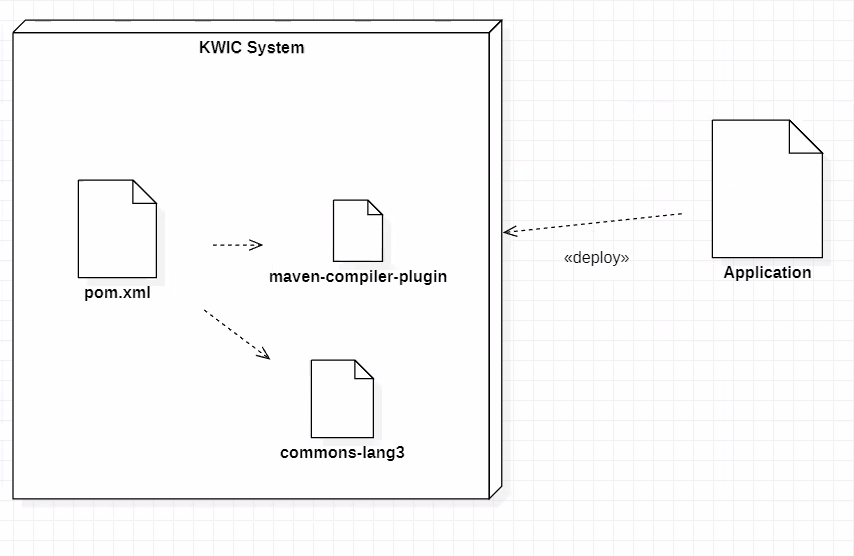


**Architecture Specification:**

The architecture pattern used in this application was a combination of Object-Oriented design pattern, and Shared data design pattern. To display this pictorially, a component and deployment diagram can be found below. The advantages of using this architecture include but not limited to information hiding by the use of encapsulation. This allowed for the creation of unique filters while only accessing that generic encapsulating object. Also, by using object-oriented architecture, we can specify exactly what information we want to use for the corresponding object rather than making a copy of an object containing fields and information that would not be needed. The shared data architecture allowed for each object to have access to original on the ready.



Deployment diagram:



**OO and Shared Data Architecture**

A combination of the two architectures was used to ensure better results. The shared data provides good performance to the system and ensures that there is no duplication of data. The OO design ensures that each module provides an interface whereby other components can access the data only by invoking that interface.

Components: processes, data and objects.

Connectors: Direct memory access, subprogram call, system I/O

**Advantages**

Performance is very efficient due to shared data and fast due to no copy.

Better reusability due to use of OO architecture.

Enhancement of system functions.

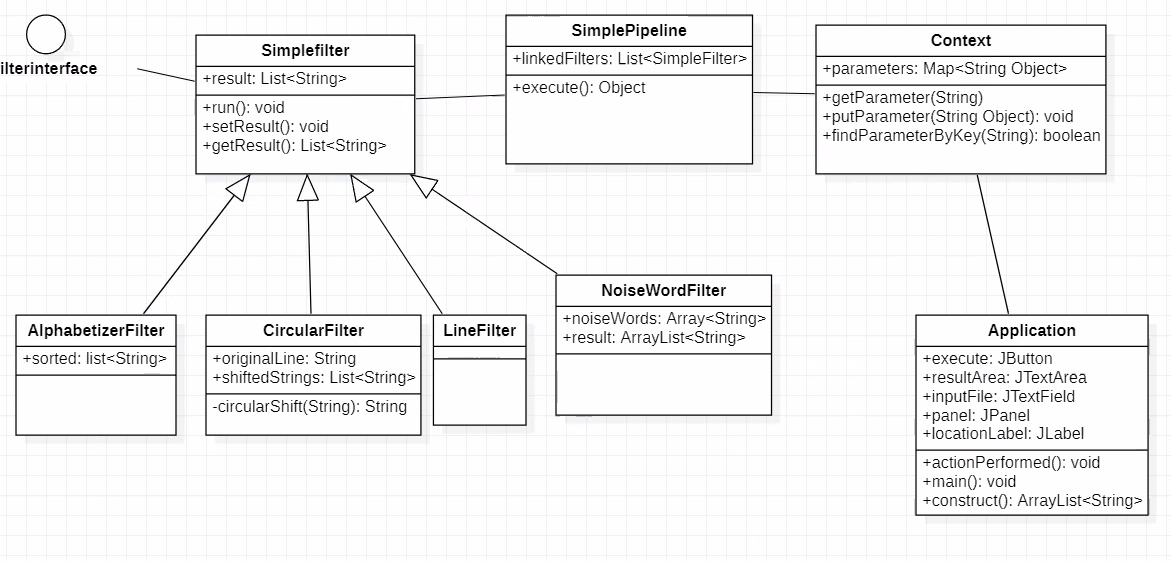
**Disadvantages**

Require users to follow a protocol for correct use of initialization and error handling.

Depends critically on single thread of control

**Design Implementation:**

We used the class diagram to elaborate this.



**Implementation Specification:**

The development view of our software system represents the software development configuration aspects of the software system. Our architecture addressed concerns such as reusability and ease of development.

**Performance comparison:**

The performance of our system is faster on average by 3 ms. This is observable when running the programming. Upon file location submission a timer is started and stopped once the results are returned to the front end. Switching between git branches, the oo branch implementing the new architecture for this assignment is faster then the time compiled under the master branch which uses the old pipe and filter architecture.

Master average compile time: 25 ms.

Oo average compile time: 22 ms.