CS6362 Software Architecture and Design

SPRING 2013

Project Phase ii- interim

cyberminer Search Engine SYSTEM

ARCHITECTURE SEPECIFICATION

Done by:

Abhishek Dey Das

Aishwarya Muthukumaran,

Brian Boyle

Clay Houllion

David Riviera

Gowrav Deivasigamani

Harshita Vissa

Naveen Kumar Devaraj

Pooja Dhondge

Sanjitha Srinivasan

Shiva Malavika Ganesh

Sripadmanabhan Subramanian

Sudharsan Lakshminarasimhan

Web URL: <http://sadteam.abhis.ws/>

Table of Contents

1. Architecture styles 3

1.1 Pipe and Filter 3

1.1.1 Components (Pipes and Filters) and their interactions 3

1.1.2 Pros 4

1.1.3 Cons 4

1.2 Abstract data type 5

1.2.1 Introduction 5

1.2.2 Modules 7

1.2.3 Pros 8

1.2.4 Cons 8

2. Rationale 8

Table of figures

Figure 1 - Pipe and Filter architecture 3

Figure 2 - ADT architeture 6

**1. Architecture Styles**

The following architecture style were considered before a decision was made regarding the architecture for the KWIC system.

* Pipe and Filter.
* Main program/sub routine with shared data.
* Abstract Data Type.
* Implicit Invocation.

Amongst the above choices, a thourough analysis was conducted for Pipe and Filter architecture and Abstract Data Type architecture since these architectures are the most suited for this project.

**1.1 Pipe and Filter**

Input Medium

Alphabetizer

Circular Shift

Input

Output Medium

Output

Filters : Input

Circular Shift

Alphabetizer

Output

**Figure 1 - Pipe and Filter architecture**

**1.1.1 Components and their interactions**

· The KWIC system can be represented as a sequence of processing filters.

1. Reading the input.

2. Making circular shifts.

3. Alphabetize.

4. Display the output.

·

Every filter processes the incoming data and transfers the output to the next filter.

These filters can be implemented as modules as described below.

· Filter 1: Input

o This filter reads the sentences from the input medium.

o Processes the data to remove the special characters and extra spaces and generates new sentences.

o It transfers the new sentences to circular shift.

· Filter 2: Circular shift

o This filters reads the output from the Input Filter.

o Performs the circular shift.

o Transfers the shifted sentences to Alphabetizer.

· Filter 3: Alphabetizer

o Reads the circularly shifted sentences.

o Sorts the sentences in the alphabetical order

· Filter 4: Output

o This module is called after alphabetization

o It displays the output to the output medium.

**1.1.2 Pros**

* It maintains the flow of processing.
* It supports reuse since each filter can function in isolation.
* New filters can be added for changing requirements.
* Easy to implement modifications since change of the functionality of one filter will not affect the other filters.

**1.1.3 Cons**

* Hard to implement for interactive systems.
* The problem of space inefficiency exists.
* Every filter must copy all data for processing.

**1.2 Abstract data type**

**1.2.1 Introduction**

In ADT architecture design, the system can be viewed as a collection of ADT objects. Data structures and their associated operations are encapsulated in an abstract data type (ADT) or object. The components of a system are instances of an ADT and they interact through procedure (or method) calls. An object is responsible for preserving the integrity of its data structures and also these data structures are hidden from other objects.

MasterControl

Splitlines()

Sort()

ShiftLine()

Getshift()

Alphabetizer

Circular Shift

Output Medium

Input Medium

**Figure 2 - ADT architecture**

**1.2.2 Modules**

There are totally six modules in this architecture, each corresponding to a specific function. We create an object to each of these, which perform their corresponding functions.

**Module1: MasterControl**

This module takes procedure calls to all other modules. This is the main class, which controls the overall system. This class creates an object for each of the other classes and calls their respective method to process the data.

The methods used by this module are:

* Init

All the applet user interface components are initialized and loaded

* ActionPerformed

This method corresponds to the button press events

**Module 2: InputManager**

This module is responsible to read the input from the input device, process it to remove special characters and multiple new lines and spaces, and then store this data in the StoreSentences module so that the processed data is available for the CircShift module.

The methods used by this module are:

* splitlines()

It splits the lines based on the delimiter.

**Module 3: CircularShift**

This module is responsible for shifting the lines in a circular manner, by appending the words one at a time.

The methods used by this module are:

* shiftLine(Vector input)

Constructs the circular shift and put the results lines into a new vector which can be used by “Alphabetizer” to reconstruct the alpha shifts of the lines

* getshift()

Returns the vector which contains circular shift lines

**Module 4: Alphabetizer**

This module sorts the circularly shifted lines in alphabetical order.

The methods used by this module are:

* sort()

This method sorts the lines based on the input order specified in the requirements.

**Module 5: OutputManager**

This module displays the alphabetically shifted lines onto the output text in the applet screen.

**1.2.3 Pros**

* The data is hidden in each module. So any changes to the data can be easily accommodated as long as the object using that data maintains the same input and output formats.
* Encourages decomposition of a problem into a number of interacting components/agents.
* Flexible and open to changes. It can accommodate new functionalities without the need to change the existing modules since each module is designed to access other modules only through well-defined interfaces.
* Supports the reusability of existing classes and objects.
* This design is easier to maintain as all modules interact with each other through well defined interface functions or procedures.

**1.2.4 Cons**

* For an object to interact with another, it must know its identity.
* When the methods of an object change, so must all other objects that use this object
* The overall performance is degraded as there are lot of operations involved in data copying and reconstruction.
* Poor response time because the data needs to be re-constructed in different modules.
* It is difficult to control the exact ordering of modules called internally by these objects.

**2. Rationale**

The following table shows the advantages and disadvantages of both the architectures.

|  |  |  |
| --- | --- | --- |
|  | Pipe and Filter | ADT/ OO |
| Change in algorithm | + + | + |
| Change data representation | + + | + |
| Add function | + | + |
| Space | - | + |
| Time | - | + |
| Reusability | + + | + |
| Intuitiveness | + + | + |

As described above abstract data type has advantages in all the non functional requirements. Whereas pipe and filter does not perform well and has poor time and space complexities when compared to abstract data type method.

As we have to design a system which could be modified and enhanced later for the up coming new requirements, therefore KWIC must be developed very modular and should have data security features. Taking these things under consideration ADT architecture is best suited. If required to build an entirely new system, we could still reuse the existing classes of the KWIC.

For the above mentioned reasons we decided to implement the project using ADT/OO architecture.

**CyberMiner Architecture:**

KWIC Indexing System

Web Crawler

Database

Search Logic

User Interface

1. **User Interface:**

The home page is Java Serverl Pages and is rendered as HTML page. The page has a form consisting of input textbox and a submit button. When the user starts typing keyword in the input textbox, the form gets submitted. The form is submitted to another jsp page, which displays the search results. The search results also has the pagination feature, so that the user could navigate through the search results.

1. **Search Logic:**

The java classes have the methods which queries the database for getting the search results. The methods present are search method and autofill method. The search method returns an array list and then that is given to the user interface. The auto fill method gives the predicted text as output, which then is displayed in the suggestion in the textbox.

1. **KWIC Indexing System:**

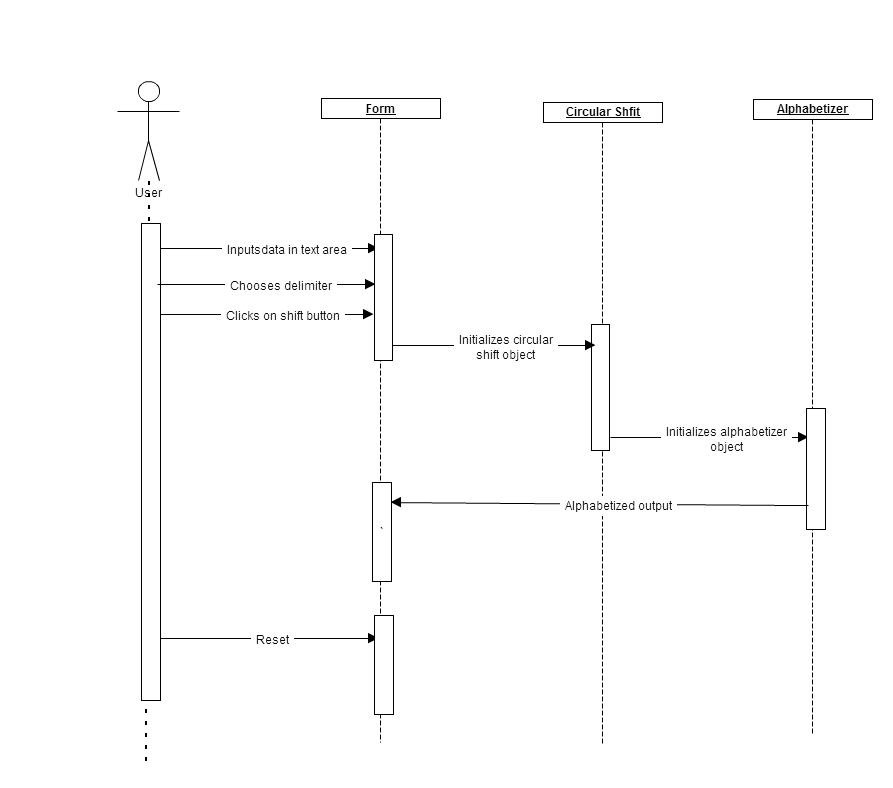
KWIC system shall accept an ordered set of lines and it would output a list of circular shifts of descriptor parts of all lines in ascending alphabetical order.

1. **Web Crawler:**

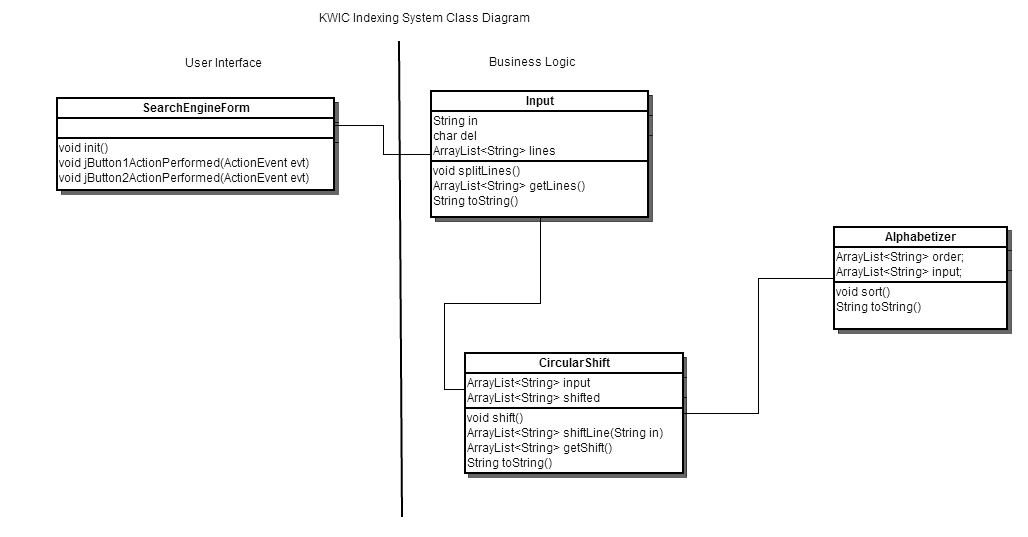
Web crawler crawls the pages of the website and gets the url and description of the pages visite and they are then indexed by KWIC Indexing system.

**3. UML Diagrams for KWIC Indexing System:**

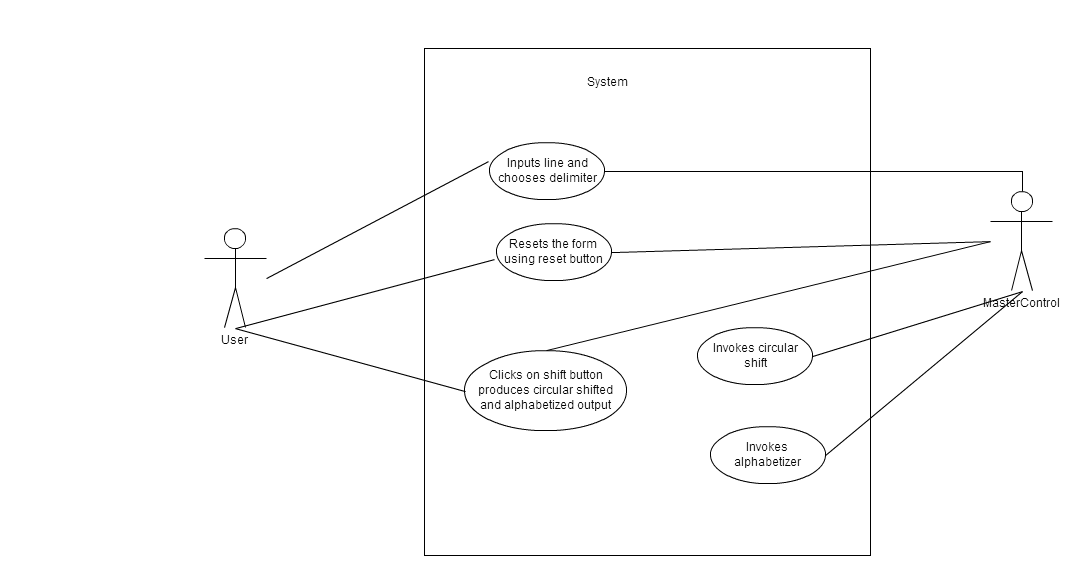
1. **System Sequence Diagram:-**

****

1. **Class Diagram:-**

****

1. **Use Case Diagram:-**

****

**References:**

* Lecture Notes
* Feedback from professor and TA
* Gliffy UML – Online UML Editor