

A Growing Search Engine Based on a J2EE Platform

Introduction

The purpose of this project is to develop a web search engine, Info-miner using KWIC+, which shall be an extension to the KWIC software system, we implemented as part of the project I. The software architecture of this project will be implemented using an Implicit Invocation architecture style instead of an OO style on a J2EE platform.

Requirement Specification

Functional Requirements

Info-miner system shall accept a list of key words and return URLs whose descriptions conations any of the given keywords. No noise words shall be given as part of the list of keywords. Info-miner shall use another software system, KWIC+, which was originally developed in project I, in order to efficiently maintain a database of URLs and the corresponding descriptions. KIWC+ shall accept an ordered set of lines, where each line consists of two parts: 1) the URL part (e.g., <http://www.utdallas.edu>) and 2) the description part of an ordered set of alphanumeric words such as the description of <http://www.utdallas.edu> is education institution. Each word is an ordered set of characters excluding any punctuation symbols or control characters. The description part of any line shall be “circularly shifted” by repeatedly removing the first word and appending it at the end of the line. The KWIC+ index system shall output a list all circular shifts of the description parts of all lines in alphabetical orders, together with their corresponding URLs. No line in the output list starts with any noise word such as “a”, “the”, and “of”. KWIC+ shall allow for tow modes of operation: i) for building an initial KWIC index; and ii) for growing the indices with later additions.

Info-miner shall have the following functions:

1. Case sensitive search: The system shall store the input as given and retrieve the input

2. Hyperlink enforcement: when the user clicks on the URL retrieved as the result of a query, the system shall automatically take the user to the web site that matches the URL;
3. Specifying OR/AND/NOT search;
4. Multiple search engine: to run concurrently;
5. Deletion of out of date URL: an corresponding description from the database;
6. Listing of the query result in alphabetically ascending order;
7. Setting the number of results to show per page, and navigation between pages;

Non-functional Requirements

The Info-miner system shall be used for a web search engine and be modeled with implicit invocation architecture style. It must be implemented by usage of Java applet.

- The Info-miner system shall be easily understandable and easy to use.
- The system shall be modifiable. For example, line shifting can be performed on each line as it is read from the input medium, or on all lines after they are read from the input medium. The alphabetic shifts can be done in batch or modified to incremental alphabetizer.
- The Info-miner system shall be enhance able. For example, it allows the system to be enhanced by Boolean operators or full text search.
- The system shall be reusable with good performance.
- The system must be portable, responsive, and adaptable.
- The system must also be user-friendly. The Info-miner system shall provide a user interface for user to enter text lines. The KWIC+ index system shall use an output medium to display all circular shifted lines in descending alphabetical order in a way such that user can read it conveniently.

Architecture Design

In this project we will use implicit invocation architecture style to implement the Info-miner search engine. Implicit Invocation is an event-driven architectural style. The main idea is that instead of invoking a procedure directly, component can announce one or more events. Components can also be associated in an event by involving a related procedure. Once the event is announced, the system itself invokes all of the procedures that have been associated with that event. Thus an event announcement “implicitly” causes the invocation of the procedures, which may reside in other modules. However, Java doesn’t completely support the implementation of the implicit invocation architecture. We have used the Java Event model to implement the implicit invocation design. The Implicit Invocation architecture design for Info-miner is pictorially presented in Figure 1.

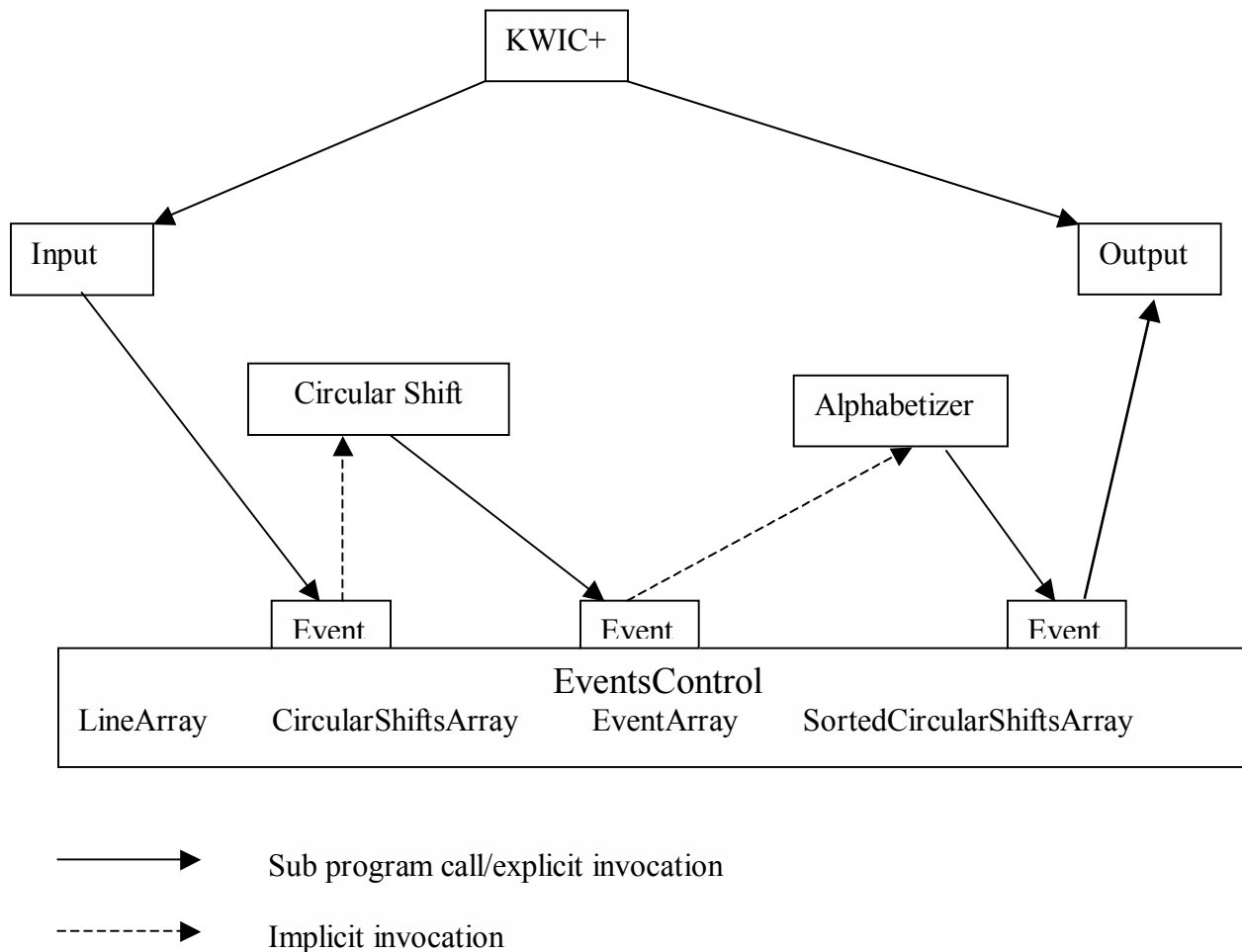


Figure 1: Implicit invocation architecture style

Components: Objects

The KWIC+ System contains the components. The modules other than KWIC+ and Search Engine are helper classes to the KWIC+ module.

1. Module KWIC+

This module makes procedure calls to all other modules that are included in the KWIC+ system. It provides the following interface for the Microminer: -

2. Module Input

This module has the following operations:

- Start – to obtain user input
- addListener – adds listener to it

3. Module Circular Shift

This module generates Circular shifts of the user input strings and creates indices of the words in the lines.

4. Module Noise Eliminator

This module eliminates the circular shifts generated by CircularShift, which start with a noise word. The following noise words will be detected by the system: {"and ", "an ", "or ", "the ", "at ", "for ", "is ", "it ", "on ", "if ", "be ", "to ", "did ", "are ", "a ", "so ", "what ", "than ", "that ", "this "}. It is part of the Circular Shift Module.

5. Module Alphabetizer

This module generates alphabetized shifts of the circular shifted descriptions obtained by accessing circularshifts array.

6. Module Output

This module is an Enterprise Java Bean, which stores the processed input in the database

- MakeCloudscapeConnection – establishes connection with the database
- storeOutput - This method stores the output to a string array.
- getOutput - This method returns a copy of the output.
- ConstructCSSString – constructs the output strings by accessing lineArray

7. Module Search Engine

This module searches the Alphabetized circular shifts produced by the KWIC+ system. When the search button is clicked, it searches the memory by matching the keywords input by the user and displays the original descriptor and the webaddress

- MakeCloudscapeConnection – Establishes database connection.
- performSearch - This internal method performs binary search on the search array to search for the matching keywords by accessing noOfIndices. printWebAddress - This internal method prints the result of the search in the text area by using getAlphaIndices of Alphabetizer module of KWICplus system.

- setSearchText

Following are the standard methods present in an Enterprise bean.

- ejbCreate
- ejbRemove
- ejbActivate

Glue

- Implicit Invocation: Each module in the system does not know about the other module, but invoked the other modules by the process of events, which is the implicit invocation.
- Subprogram Call: The procedure call is made through acquiring an instance of one module and invokes the corresponding methods/procedures.
- System I/O: This is Input/Output from outside medium.

Constraints

Computations are invoked implicitly as data is modified, based on active data model.

Pattern

Implicit procedure calls are made through the invocation of events.

J2EE 4 Tier client-style Architecture

The Info-miner web search engine also use the J2EE 4 Tier client-server style of architecture for our system. Figure 2 shows the 4 Tier architecture design.

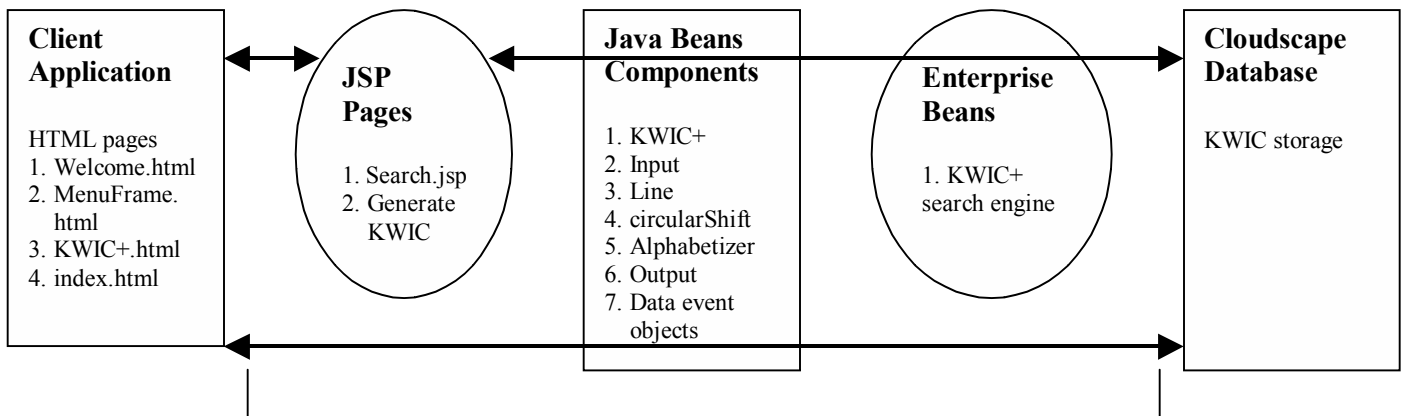


Figure 2: J2EE Server

Advantages and Disadvantages

Implicit Invocation architecture has the features from both shared data style and O-O styles. It has the advantages of supporting functional enhancements to the system for example the additional modules can be attached to the system by registering them and invoking them based on data-changing events. Furthermore the data is accessed abstractly, which insulates computations from changes during data representation.

However, there is a major disadvantage of implicit invocation, which it can be difficult to control the processing order of the implicitly invoked modules. This makes it difficult to follow the particular event. Another main drawback of this style is that its implementation will use more space than other style decompositions due to the data-driven of implicit invocation.

OO Architecture Style

- Data is encapsulated in each component. Each module provides an interface for other components to access the data by invoking that interface (Information hiding).
- Individual modules can be changed without affecting others. This feature will allow the flexibility of algorithm change if necessary.
- The addition of functionality is relatively easy in ADT architecture design. We can expand the system in the future enhancements.

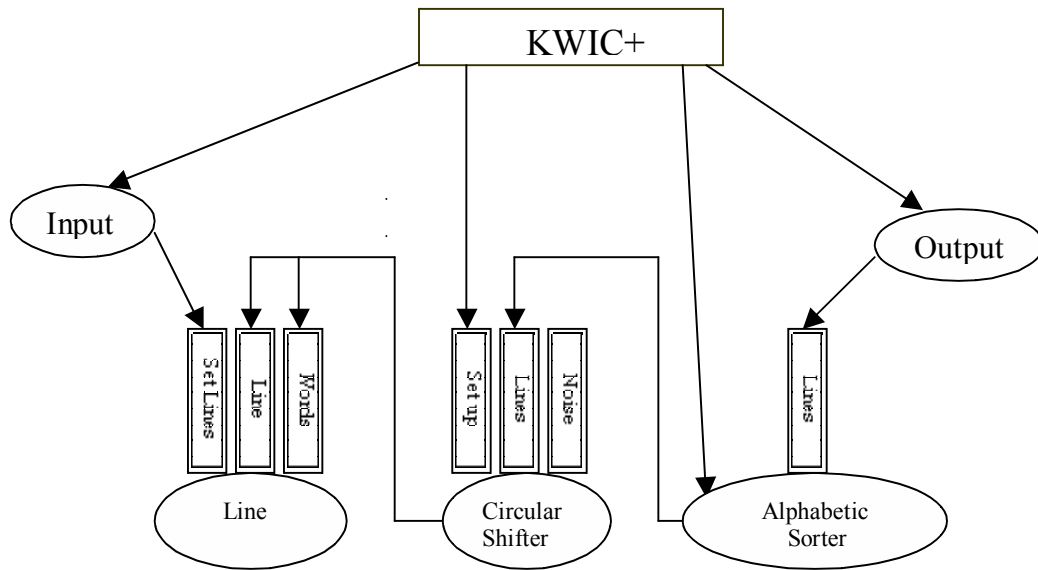
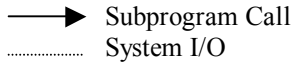


Fig. 1: Architecture Design based on OO Style

Components

a. Input:

- Reads data from the input medium;
- Captures the event when the submit button is clicked;
- Stores data lines by calling methods of Line Storage.

b. Lines:

- Used to store the ordered set of words
- Provides the real world abstraction for a set of strings
- Used by the input module to store the input lines

c. Circular Shifter:

- Gets the input lines from the lines storage

- Generates circularly shifted lines and use it to display to the user
- Stores the circular shifts

d. Sort:

- Reads the circularly shifted lines and internally sorts and generates the alphabetized circular shifts
- Stores the sorted strings to the output.

e. Output:

- Reads the output and Displays it in the screen

f. KWIC:

- This module sequences operations among all other modules.

Glue

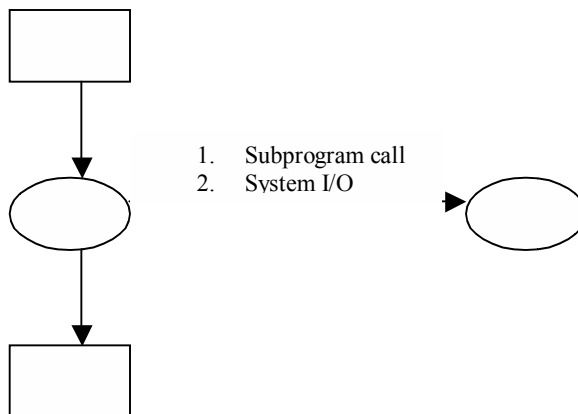
- Subprogram call: The subprogram call is invoked through the instantiation of an object and introduces the related methods and procedures.
- System I/O: The system I/O is both the input and output media.

Constraints

In ADT architecture design, the information hiding limits the data access only through the corresponding interface. This limitation results in the information duplication and space consumption.

Patterns

Each module can call the procedures in other module through interface during the running time.



The relation and map between OO style and implicit invocation style

Module	OO Style	Implicit Invocation
Input	Explicitly invoked by KWICplus	Explicitly invoked by KWICplus
CircularShift	Explicitly invoked by Line	Implicitly invoked by Input
NoiseEliminator	Explicitly invoked by CircularShift	Implicitly invoked by CircularShift
Alphabetizer	Explicitly invoked by NoiseEliminator	Implicitly invoked by NoiseEliminator
Output	Explicitly invoked by Alphabetizer	Implicitly invoked by Alphabetizer. It is an Enterprise Java Bean, which stores the processed user input into database.
KWIC+	Explicitly invoked by Microminer. This module controls the sequence of execution of the KWIC+ system constituted by the above modules.	Explicitly invokes Input and Output modules and it is not responsible for controlling the sequence. It is an Enterprise Bean responsible for processing user input.
SearchEngine	Explicitly invoked by Microminer. This module searches the Alphabetized circular shifts produced by the KWIC+ system. The system searches the descriptors entered by the user at that moment.	It is an Enterprise Java Bean, which searches for the url and descriptors.
Microminer	This is the Master controller which invokes the KWIC+ and the SearchEngine modules.	There is no Microminer module here.