Modularization 1: Shared Data

Data is communicated between the components through shared storage

System KWIC

**global:** Characters, Index, AlphabetizedIndex

**module** Input

operation read: data lines from the input medium

operation store: data lines (as packed "Characters" in core)

/* e.g., Characters: packed array [1..10000] of char */

The KWIC Problem

HOW ARCHITECTURE WINS TECHNOLOGY WARS
ARCHITECTURE WINS TECHNOLOGY WARS HOW
WINS TECHNOLOGY WARS HOW ARCHITECTURE
TECHNOLOGY WARS HOW ARCHITECTURE WINS
WARS HOW ARCHITECTURE WINS TECHNOLOGY

THE ART OF SYSTEMS ARCHITECTING
ART OF SYSTEMS ARCHITECTING THE
OF SYSTEMS ARCHITECTING THE ART
SYSTEMS ARCHITECTING THE ART OF
ARCHITECTING THE ART OF SYSTEMS

ARCHITECTING THE ART OF SYSTEMS
ARCHITECTURE WINS TECHNOLOGY WARS HOW
ART OF SYSTEMS ARCHITECTING THE
HOW ARCHITECTURE WINS TECHNOLOGY WARS
OF SYSTEMS ARCHITECTING THE ART
SYSTEMS ARCHITECTING THE ART OF
TECHNOLOGY WARS HOW ARCHITECTURE WINS
THE ART OF SYSTEMS ARCHITECTING
WARS HOW ARCHITECTURE WINS TECHNOLOGY
WINS TECHNOLOGY WARS HOW ARCHITECTURE

Input

Circular Shift

Alphabetizer

Output
Modularization 1: Shared Data

Data is communicated between the components through shared storage

module Circular Shift

operation readChar: packed characters from "Characters"
operation prepare: the starting index of the line (e.g., Index.first) and the offset for each word from the starting position.

/* not a physical circular shift, "Output" does this */
/* one data structure choice:
   Index
   first: ^Characters
   offset: positive integer */

The KWIC Problem

How architecture wins technology wars

<table>
<thead>
<tr>
<th>Input</th>
<th>first</th>
<th>offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOW ARCHITECTURE WINS TECHNOLOGY WARS HOW ARCHITECTURE WINS TECHNOLOGY WARS HOW WINS TECHNOLOGY WARS HOW ARCHITECTURE TECHNOLOGY WARS HOW ARCHITECTURE WINS WARS HOW ARCHITECTURE WINS TECHNOLOGY</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>18</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Circular Shift</th>
<th>first</th>
<th>offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>THE ART OF SYSTEMS ARCHITECTING THE ART OF SYSTEMS ARCHITECTING THE ART OF SYSTEMS ARCHITECTING THE ART OF SYSTEMS ARCHITECTING</td>
<td>39</td>
<td>1</td>
</tr>
<tr>
<td>39</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>
Modularization 1: Shared Data

Data is communicated between the components through shared storage

module Alphabetizer
operation readChar: packed characters from "Characters"
operation readIndex: from "Index"
operation alphabetize: convert "index" to an "alphabetized index"
by listing the circular shifts alphabetically
/* as with "Circular Shift", no explicit representation */
/* sorting algorithm hidden */
/* as with "Circular Shift", local data structure hidden */

The KWIC Problem

Input

<table>
<thead>
<tr>
<th>HOW ARCHITECTURE WINS TECHNOLOGY WARS</th>
<th>first</th>
<th>offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARCHITECTURE WINS TECHNOLOGY WARS</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>TECHNOLOGY WARS HOW ARCHITECTURE</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>WAR HOW ARCHITECTURE WINS TECHNOLOGY</td>
<td>1</td>
<td>18</td>
</tr>
<tr>
<td>HOW ARCHITECTURE WINS TECHNOLOGY WARS</td>
<td>1</td>
<td>18</td>
</tr>
<tr>
<td>THE ART OF SYSTEMS ARCHITECTING</td>
<td>39</td>
<td>1</td>
</tr>
<tr>
<td>ART OF SYSTEMS ARCHITECTING THE ART</td>
<td>39</td>
<td>5</td>
</tr>
<tr>
<td>OF SYSTEMS ARCHITECTING THE ART</td>
<td>39</td>
<td>20</td>
</tr>
<tr>
<td>SYSTEMS ARCHITECTING THE ART OF</td>
<td>39</td>
<td>20</td>
</tr>
<tr>
<td>ARCHITECTING THE ART OF SYSTEMS</td>
<td>39</td>
<td>20</td>
</tr>
<tr>
<td>ARCHITECTURE WINS TECHNOLOGY WARS</td>
<td>39</td>
<td>20</td>
</tr>
<tr>
<td>HOW ART OF SYSTEMS ARCHITECTING THE</td>
<td>39</td>
<td>20</td>
</tr>
<tr>
<td>HOW ARCHITECTURE WINS TECHNOLOGY WARS</td>
<td>39</td>
<td>20</td>
</tr>
<tr>
<td>OF SYSTEMS ARCHITECTING THE ART</td>
<td>39</td>
<td>20</td>
</tr>
<tr>
<td>SYSTEMS ARCHITECTING THE ART OF</td>
<td>39</td>
<td>20</td>
</tr>
<tr>
<td>TECHNOLOGY WARS HOW ARCHITECTURE WINS</td>
<td>39</td>
<td>20</td>
</tr>
<tr>
<td>THE ART OF SYSTEMS ARCHITECTING</td>
<td>39</td>
<td>20</td>
</tr>
<tr>
<td>WARS HOW ARCHITECTURE WINS TECHNOLOGY</td>
<td>39</td>
<td>20</td>
</tr>
</tbody>
</table>

Circular Shift

Alphabetizer
Modularization 1: Shared Data

Data is communicated between the components through shared storage.

module Output
operation readChar: packed characters from “Characters”
operation readIndex: from “AlphabetizedIndex”
operation print: list all the circular shifts using “Characters” and “AlphabetizedIndex”

/* a candidate algorithm: */
loop i
  go to position (first-i, offset-i) and print everything before $;
  go to position (first-i, 1) and print everything before (first-i, offset-i )*/
pool

The KWIC Problem

HOW ARCHITECTURE WINS TECHNOLOGY WARS$THE ART OF SYSTEMS ARCHITECTING

Input

HOW ARCHITECTURE WINS TECHNOLOGY WARS
ARCHITECTURE WINS TECHNOLOGY WARS HOW
ARCHITECTURE WINS TECHNOLOGY WARS HOW
ARCHITECTURE WINS TECHNOLOGY WARS HOW
ARCHITECTURE WINS TECHNOLOGY WARS HOW
ARCHITECTURE WINS TECHNOLOGY WARS HOW

first  offset
1      1
1      5
1      18
39     1
39     5
39     20
39     20
39     5
Modularization 1: Shared Data

Data is communicated between the components through shared storage

module Master Control
" control the sequencing among the four modules "
operation main: " batch mode, original proposal by Parnas "
Input;
Circular Shift;
Alphabetizer;
Output
" incremental mode -> loop"
" concurrent mode?"

Lawrence Chung

The KWIC Problem

Non-Functional Requirements

modifiability -- changes in processing algorithms
- e.g., line shifting: one at a time as it is read or all after they are read or on demand when the alphabetization requires a new set of shifted lines
- e.g., batch alphabetizer vs. incremental alphabetizer

modifiability -- changes in data representation
- e.g., storing characters, words and lines (e.g., in 1-d array/2-d array/linkedList-array, compressed vs. uncompressed)

enhancibility -- additions of (enhancement to) system function
+ e.g., to eliminate noise words (where? -> Input -> Circular Shift -> Alphabetizer -> Output ->)
- e.g., "a", "an", "the", "and", "or", "in", "of", "with", "for" + "it", "you", "it", "they", ...) the user deletes lines from the original or shifted lines (creates blank entries in "index" -> Alphabetizer needs to change)

performance -- space and time
+ efficient data representation, as the same storage is shared
- fast, thanks to no copying of data

reusability -- to what extent can the components serve as reusable entities?
- low, due to dependence on shared data and format (e.g., Alphabetizer cannot assume full explicit representation by Circular Shift)

Lawrence Chung
Modularization 1: Shared Data

The architecture

* style: Shared Data
* component: Processes & Data (per individual descriptions)
* glue: Direct Memory Access, Subprogram Call, System I/O
* constraint: Data is communicated between the components through shared storage
* pattern:

```
+-----------------+   +-----------------+
| Subprogram Call |   | System I/O      |
+-----------------+   +-----------------+
| Direct Memory Access |
```

* rationale: (if selected, NFRs)
* some intuitive appeal, as distinct computational aspects are isolated in different modules
* shared data still abundant (although OO style becoming popular)