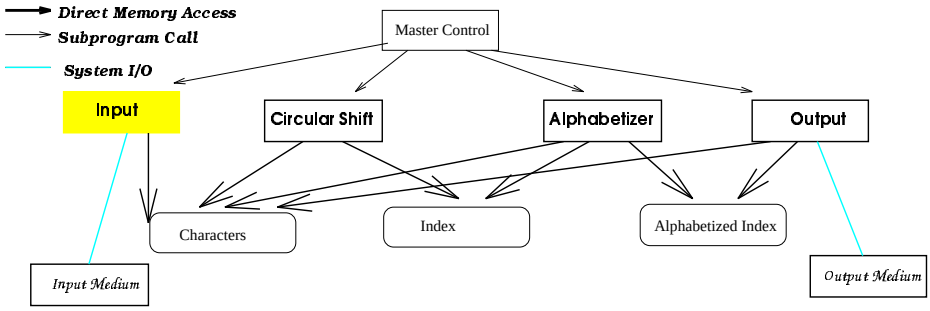
Modularization 1: Shared Data



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$THE ART OF SYSTEMS ARCHITECTING

Input

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

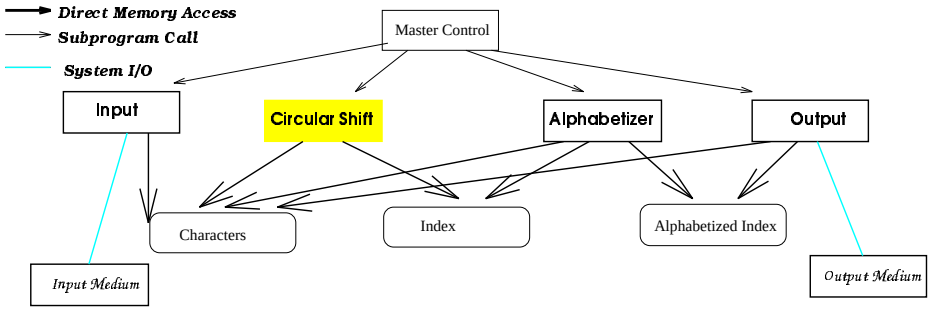
Circular Shift

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

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 */*

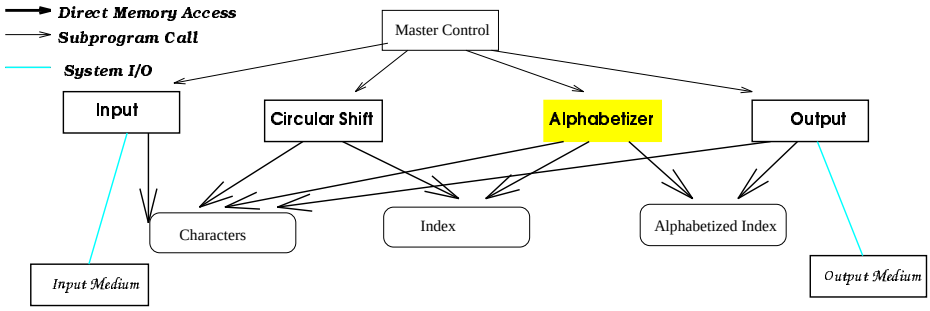
The KWIC Problem

HOW ARCHITECTURE WINS TECHNOLOGY WARS$THE ART OF SYSTEMS ARCHITECTING **Input**

	first	offset
HOW ARCHITECTURE WINS TECHNOLOGY WARS	1	1
ARCHITECTURE WINS TECHNOLOGY WARS HOW	1	5
WINS TECHNOLOGY WARS HOW ARCHITECTURE	1	18
TECHNOLOGY WARS HOW ARCHITECTURE WINS		
WARS HOW ARCHITECTURE WINS TECHNOLOGY		
THE ART OF SYSTEMS ARCHITECTING	39	1
ART OF SYSTEMS ARCHITECTING THE	39	5
OF SYSTEMS ARCHITECTING THE ART		
SYSTEMS ARCHITECTING THE ART OF		
ARCHITECTING THE ART OF SYSTEMS		

Circular Shift

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

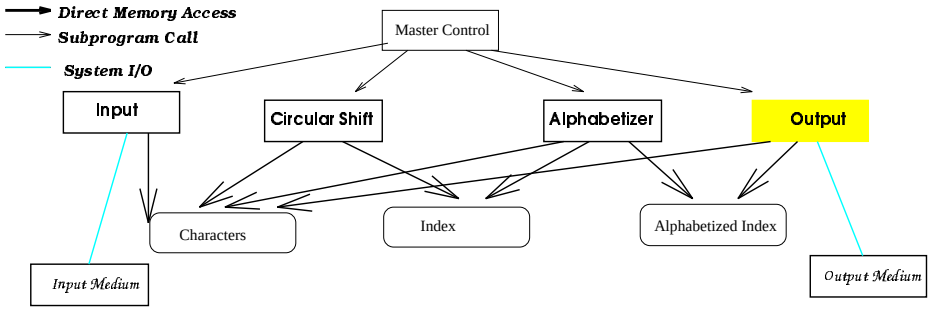
HOW ARCHITECTURE WINS TECHNOLOGY WARS$THE ART OF SYSTEMS ARCHITECTING **Input**

	first	offset
HOW ARCHITECTURE WINS TECHNOLOGY WARS	1	1
ARCHITECTURE WINS TECHNOLOGY WARS HOW	1	5
WINS TECHNOLOGY WARS HOW ARCHITECTURE	1	18
TECHNOLOGY WARS HOW ARCHITECTURE WINS		
WARS HOW ARCHITECTURE WINS TECHNOLOGY		
THE ART OF SYSTEMS ARCHITECTING	39	1
ART OF SYSTEMS ARCHITECTING THE	39	5
OF SYSTEMS ARCHITECTING THE ART		
SYSTEMS ARCHITECTING THE ART OF		
ARCHITECTING THE ART OF SYSTEMS	39	20
ARCHITECTING THE ART OF SYSTEMS	39	20
ARCHITECTURE WINS TECHNOLOGY WARS HOW	1	5
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		

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**

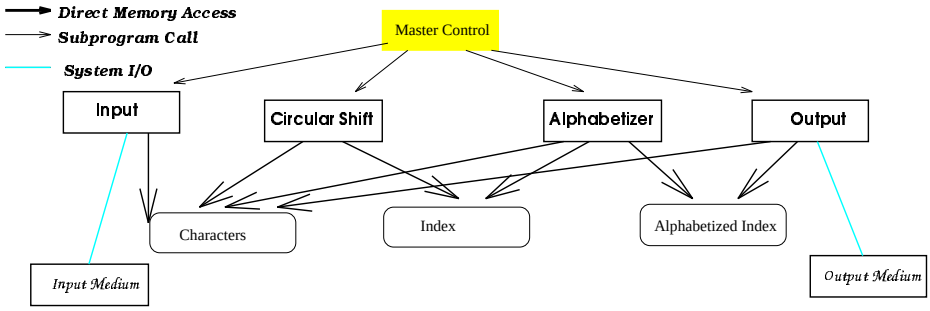
	first	offset
HOW ARCHITECTURE WINS TECHNOLOGY WARS	1	1
ARCHITECTURE WINS TECHNOLOGY WARS HOW	1	5
WINS TECHNOLOGY WARS HOW ARCHITECTURE	1	18
TECHNOLOGY WARS HOW ARCHITECTURE WINS		
WARS HOW ARCHITECTURE WINS TECHNOLOGY		
THE ART OF SYSTEMS ARCHITECTING	39	1
ART OF SYSTEMS ARCHITECTING THE	39	5
OF SYSTEMS ARCHITECTING THE ART		
SYSTEMS ARCHITECTING THE ART OF		
ARCHITECTING THE ART OF SYSTEMS	39	20
ARCHITECTING THE ART OF SYSTEMS	39	20
ARCHITECTURE WINS TECHNOLOGY WARS HOW	1	5
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		

Circular Shift

Alphabetizer

Output

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

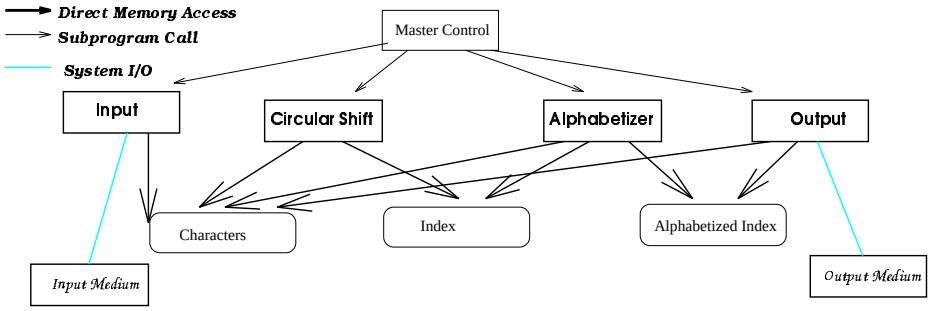
/ incremental mode -> loop*/*

/ concurrent mode?*/*

The KWIC Problem

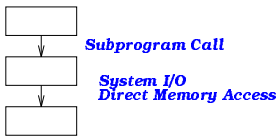
- *Non-Functional Requirements*
  - ***modifiability --- changes in processing algorithms*** -
    - e.g., line shifting: one at a time as it is read or ***-> affects alphabetizer***
      all after they are read or
      on demand when the alphabetization requires a new set of shifted lines
    - e.g., batch alphabetizer vs. incremental alphabetizer ***-> affects circular shift***
  - ***modifiability --- changes in data representation*** -
    - e.g., storing chacters, words and lines
      (e.g., in 1-d array/2-d array/linked-array, compressed vs. uncompressed)
      storing circular shifts explicitly or implicitly (as pairs of index and offset)
      core storage vs. secondary storage ***-> affects almost all modules***
  - ***enhanceability --- 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", "I", "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)

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:



- rationale: ***(if selected, NFRs)***

+ some intuitive appeal, as distinct computational aspects are isolated in different modules
o shared data still abundant (although OO style becoming popular)