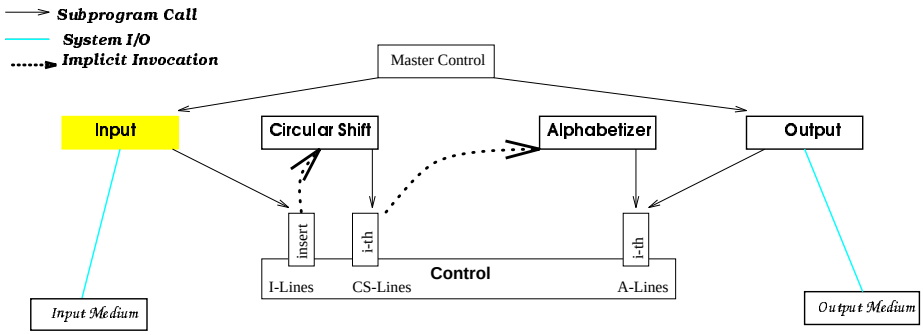
Modularization 3: Implicit Invocation



Component integration based on shared data
But,
the interface to the data is more abstract than in Shared Data (like in ADT)
i.e., storage formats are not exposed to computing modules,
but data are accessed abstractly (e.g., as a list or a set)
computations are invoked implicitly as data is modified, based on active data model

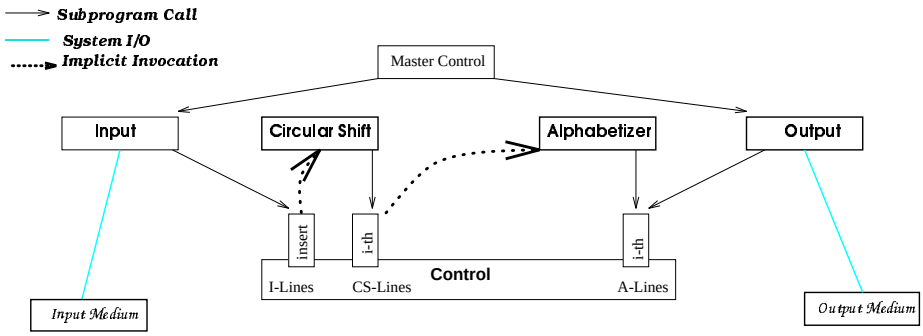
System KWIC

module Input

operation read: data lines from the input medium
*/through operation insert, a new line to the line buffer, "I-Lines" */*

/* -> implicitly invokes (triggers) */

Modularization 3: Implicit Invocation



module Circular Shift
/ through operation i-th, read I-lines and produce shifted lines and store them in a separate abstract line buffer, CS-lines */*

/* -> implicitly invokes (triggers) */

module Alphabetizer
/ through operation (2nd) i-th, read CS-lines, produce alphabetized lines and store them in a separate abstract line buffer, A-lines */*

module Output */* access A-lines and print out */*

module Master Control */* explicitly invokes Input and Output */*

Question: How does Master Control know when to invoke Output?

The KWIC Problem

Non-Functional Requirements

- ***modifiability --- changes in processing algorithms*** *+*

  e.g., line shifting: one at a time as it is read or *+ not affecting others*
  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*** *+ data access is abstract*

  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 *+ not affecting others*

- ***enhanceability --- additions of (enhancement to) system function*** *+*

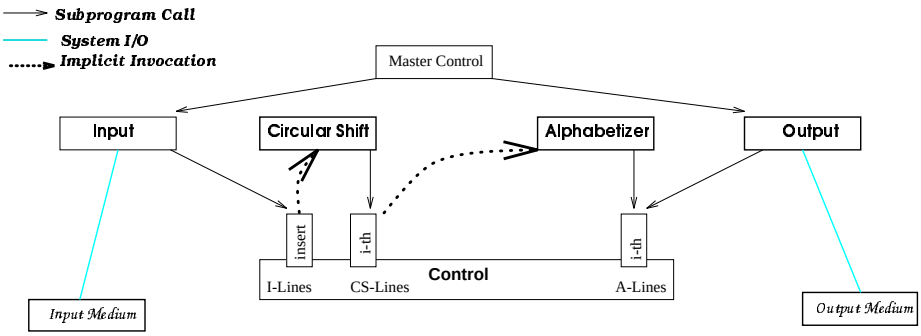
  e.g., to eliminate noise words *++ additions are invisible to other modules (registration only)*
  (e.g., "a", "an", "the", "and", "or", "in", "of", "with", "for", "I", "you", "it", "they", ...)
  the user deletes lines from the original or shifted lines

- ***performance --- space and time***
  + tend to use more space than previous ones for natural representations
  - can be inefficient due to triggering (detection can be costly)

- ***reusability --- to what extent can the components serve as reusable entities?***
  + high, implicitly invoked modules rely only on the existence of certain externally triggered events

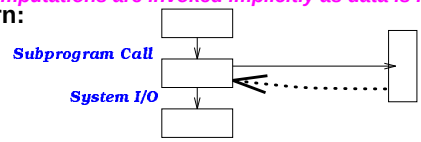
Lawrence Chung

Modularization 3: Implicit Invocation



The architecture

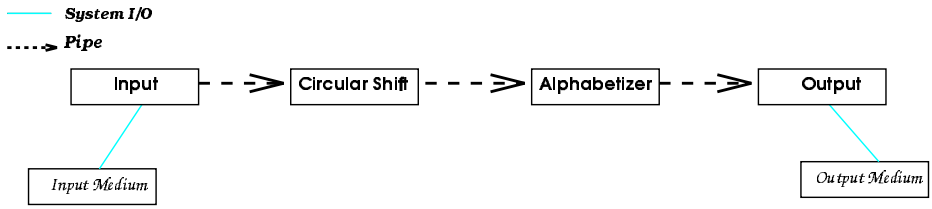
- style: *Implicit Invocation*
- component: *Processes & Data (Repository)* *(per individual descriptions)*
- glue: *Subprogram Call* *System I/O* *Implicit Invocation*
- constraint:
  computations are invoked implicitly as data is modified, based on active data model
- pattern:



- rationale: *(if selected, NFRs)*
  - can be difficult to control the order of processing

Lawrence Chung

Modularization 4: Pipe and Filter



Each filter processes the input data and produces output data
Each filter can run whenever it has data on which to compute
Data sharing between filters is strictly limited to that transmitted on pipes

System KWIC
- ***filter Input***
- ***filter Circular Shift***
- ***filter Alphabetizer***
- ***filter Output***

The KWIC Problem

- *Non-Functional Requirements*
  - ***modifiability --- changes in processing algorithms*** *+*
    - e.g., line shifting: one at a time as it is read or *+ process independence*
      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 chacters, words and lines *+ process independence*
      (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
  - ***enhanceability --- additions of (enhancement to) system function***
    - e.g., to eliminate noise words *+ by inserting filters at the appropriate points*
      (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
      - difficult to support an interactive system
  - ***performance --- space and time***
    - inefficient use of space, as each filter must copy all input data to its output port
    - inefficient, due to data replication (if not concurrent processing)
  - ***reusability --- to what extent can the components serve as reusable entities?***
    + high, thanks to process independence
  - *intuitive flow of processing*

Summary

- *Tradeoff analysis*

		Shared Data	*ADT*	*Implicit Invocation*	*Pipe & Filter*
modifiability	[algorithm]	-	+	+	++
	[data rep.]	--	+	++-	++
enhanceability	[add function]	+-	++-	++	+-
performance	space	++	+(?)	-	--(?)
	response time	+	+	-	-(?)
reusability		--	+	+	++
intutitiveness		-	+	+ -(?)	+

- *Prioritize*
  - *e.g., enhanceability of highest priority*
- *Other NFRs*
  - *e.g., intuitiveness*
- *More Scenario Analysis*
  - *e.g., Include + Omit*

Reading Assignment

Sections 4.1 (Shared Information Systems) &

4.2 (Database Integration)

Section 3.2 (Instrumentation Software)