Modular Decomposition Issues

Design Alternatives & Selection Criteria
The KWIC Problem
Architectural Alternatives
Summary

Design Alternatives & Selection Criteria

Your architecture is as good as (or as poor as) your design decisions!

➔ (potentially huge) design space

➔ How do we select a "good" solution?

basis for tradeoff analysis & design rationale
The KWIC Problem

 Functional Requirements

The KWIC (Key Word in Context) index system accepts an ordered set of lines, (where)
each line is an ordered set of words, and
each word is an ordered set of characters.

Any line may be “circularly shifted”
by repeatedly
removing the first word and
appending it at the end of the line.

The KWIC index system outputs a listing of all circular shifts of all lines in alphabetical order.
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/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
  (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

- reusability -- to what extent can the components serve as reusable entities?

Architectural Alternatives (to be considered)

- Direct Memory Access
  - Subprogram Call
  - System I/O

- Implicit Invocation

- Pipe

Lawrence Chung