On the Criteria To Be Used in Decomposing Systems into Modules

D.L. Parnas
Carnegie-Mellon University
Diciembre de 1972

David Lorge Parnas

- Doctorado en ingeniería eléctrica
- De los primeros en aplicar conceptos de la ingeniería al desarrollo de software
- Trabajó como profesor por muchos años
- Introdujo los conceptos cohesión y acoplamiento entre módulos
- Reconocimientos:
  - ACM "Best paper" award 1979:
    Designing Software for Ease of Extensionand Contraction
  - ICSE "Most influencial paper" award 1991,1995:
    Designing Software for Ease of Extensionand Contraction y
    The Modular Structure of Complex Systems

Frases de Parnas

What advice do you have for computer science/software engineering students?

... pay more attention to the fundamental ideas rather than the latest technology...

...However, what worries me about what I just said is that some people would think of Turing machines and Goedel's theorem as fundamentals. I think those things are fundamental but they are also nearly irrelevant. I think there are fundamental design principles, for example structured programming principles, the good ideas in "Object Oriented" programming, etc.

Introducción

Este paper discute sobre la modularización como un mecanismo para:

- Mejorar la flexibilidad
- Mejorar la comprensibilidad
- Reducir el tiempo de desarrollo

Se discutirá sobre el criterio para separar el sistema en módulos

Se presenta un problema de diseño y es resuelto con el método convencional (programación estructurada) y con el no-convencional.
Se comparan las ventajas y desventajas de ambos métodos.

Modularización

¿Qué es Modularizar?:

- La modularización es la asignación de responsabilidades a distintos módulos, con decisiones de diseño previas al desarrollo de los mismos.

Beneficios esperados:

- **Gerenciamiento**: Se espera que grupos separados puedan trabajar en diferentes módulos reduciendo el tiempo de desarrollo
- **Flexibilidad**: Se puede modificar un módulo sin necesidad de modificar otros
- **Comprensibilidad**:Es posible estudiar un módulo a la vez

Ejemplo KWIC Index Production System

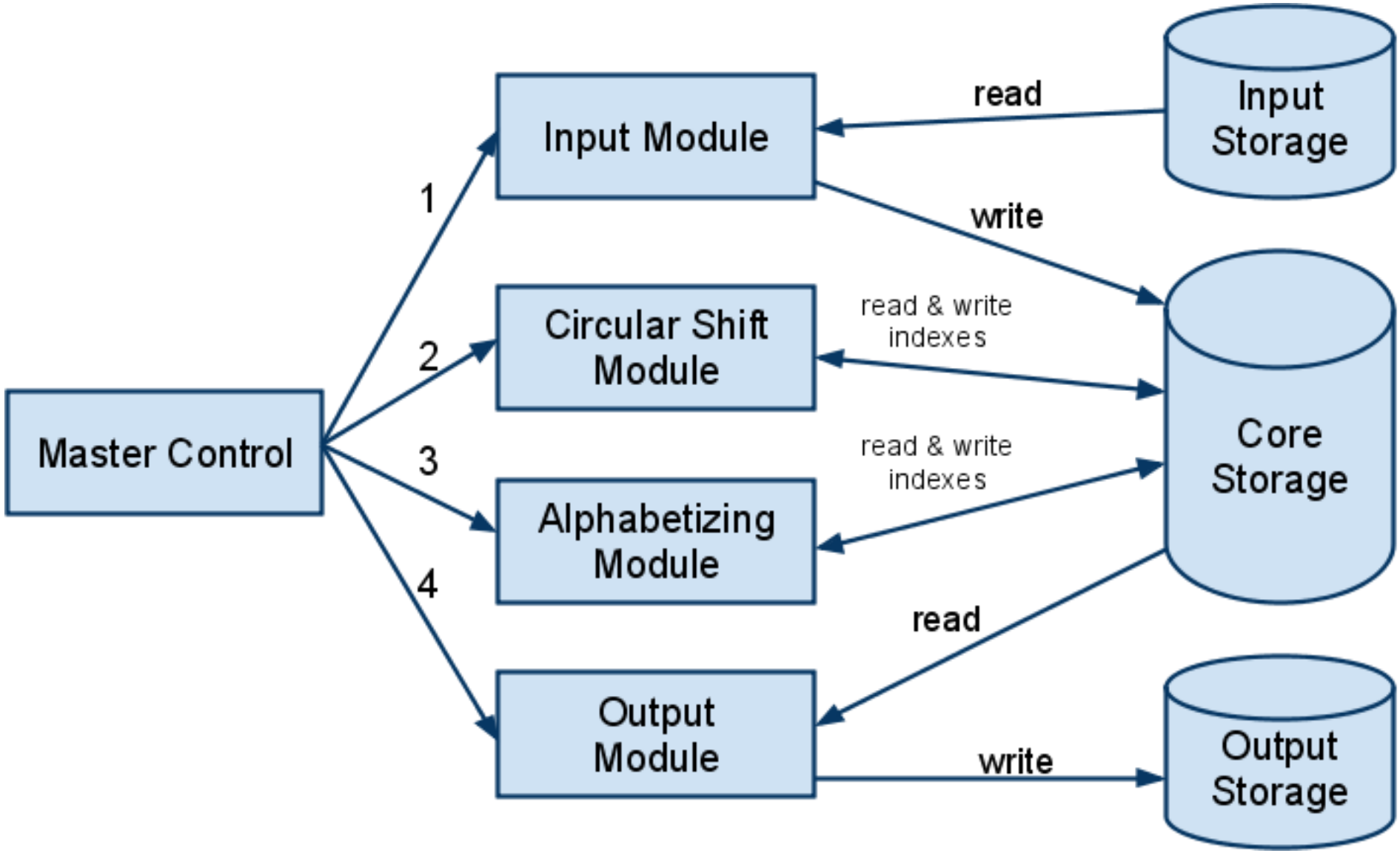
- KWIC significa Key Word in Context
- Un índice KWIC se forma ordenando alfabeticamente y alineando palabras
- Ejemplo:

Dado el texto: "KWIC is an acronym for Key Word In Context "

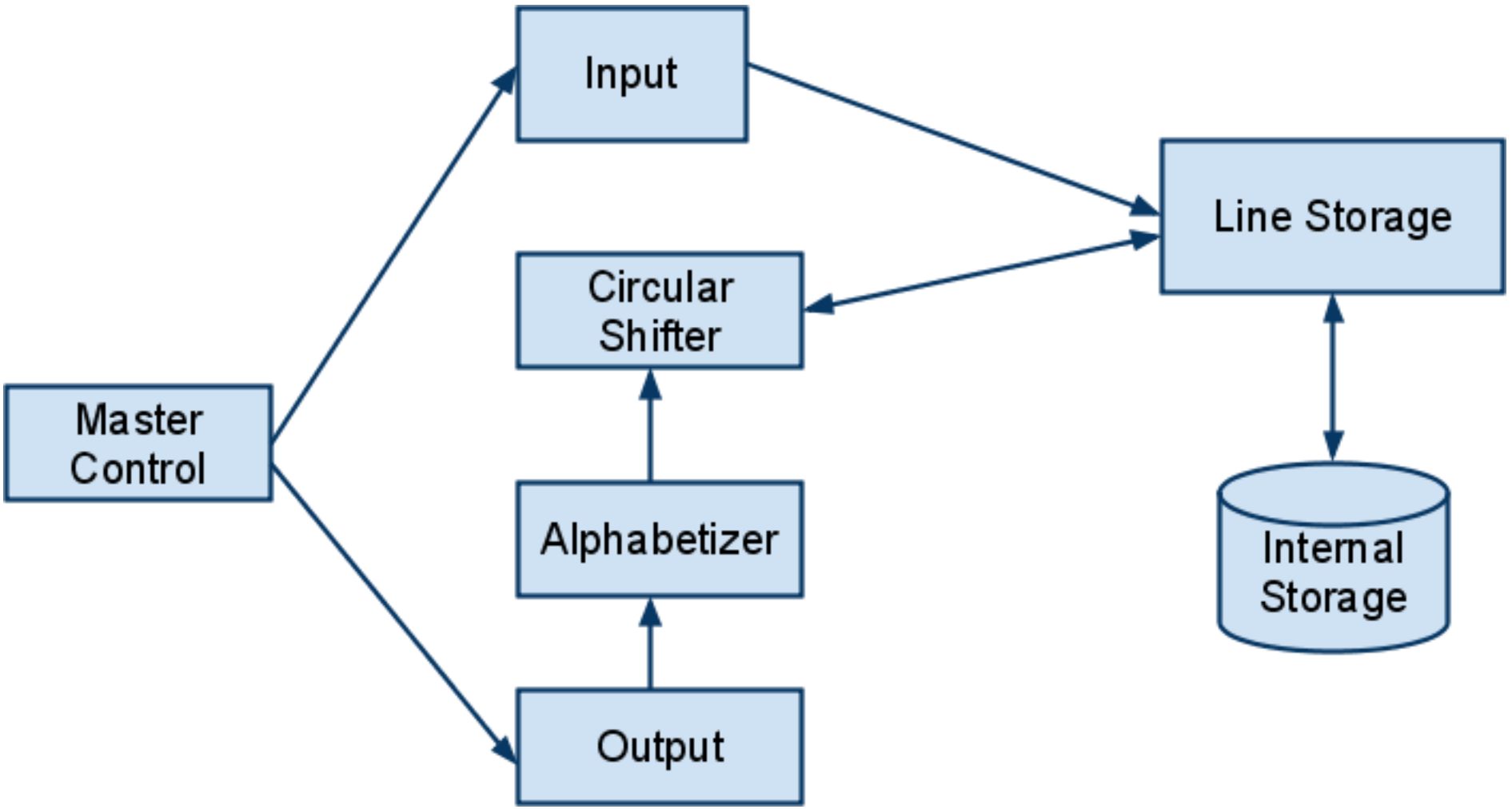
El indice KWIC sería:

... is an **acronym** for Key
... Word In **Context**
... acronym for **Key** Word In ...
KWIC is an acronym ...
... for Key **Word** In ...

Diseño sin Information Hiding



Diseño con Information Hiding



Comparación

- Ambos esquemas funcionarán.
- Tal vez ambos sea idénticos en la representación run-time (en cuanto al código de bajo nivel). Esta representación sólo sirve para ejecutar.
- Pero no son iguales en cuanto al código fuente. Que sirve para comprender, modificar y documentar.

Modificabilidad

Modificación	**Cambios Sin I.H**	**Cambios Con I.H**
Cambio del formato de la entrada	Módulo Input	Módulo Input
Guardar todas las líneas en el core vs. sólo algunas	Todos los módulos	Módulo Line Storage
Comprimir vs. no comprimir líneas en core	Todos los módulos	Módulo Line Storage
Hacer vs. no hacer índice para los circular shifts	Módulos Circular Shift, Alphabetizer y Output	Módulo Circular Shift
Alfabetizar la lista una vez vs. Alfabetización parcial / on demand	Módulos Alphabetizer y Output	Módulo Alphabetizer

Desarrollo Independiente

Sin Information Hiding:

- Los módulos se comunican a través de tablas
- El diseño del esquema de tablas es complejo y debe ser hecho antes de comenzar con el desarrollo de los módulos

Con Information Hiding:

- Se usan interfaces más abstractas
- El diseño de las interfaces entre módulos es más simple y por lo tanto el desarrollo de los mismos debería comenzar antes.

Comprensibilidad

Sin information Hiding:

- Comprender el módulo Output requiere comprender el Input, Alphabetizer y Circular Shift.
- Las estructuras de las tablas no tienen sentido si no se comprenden los módulos.
- El sistema sólo puede ser comprendido como un todo.

Con Information Hiding

- Cada módulo puede ser comprendido por separado.
- Utilizando nombres declarativos en los nombres de las interfaces se puede inferir su comportamiento

El Criterio

Sin Information Hiding

- Descomposición Flowchart
- Cada módulo es un paso de procesamiento

Con information Hiding

- No corresponden a pasos de procesamiento (ej: Line Storage)
- Cada módulo se caracteriza por su conocimiento sobre una decisión de diseño, la cual oculta a los demás

Eficiencia

Para la época la implementación utilizando information hiding podía ser considerablemente más costosa por el mayor switcheo de control entre módulos.

Parnas propone una solución al problema:
Tener una herramienta que ayude a compilar el código de manera tal que la separación de módulos no se encuentre en el código de máquina.

Estructura Jerárquica

Parnas dice que dos módulos están jerárquicamente relacionados si uno "usa" o "depende-de" el otro.

Considera que la existencia de una estructura jerárquica entre módulos es beneficiosa, ya que permite la reutilización de módulos.

La descomposición y la estructura jerárquica son propiedades independientes de un sistema.

Conclusiones

- Es incorrecto descomponeren módulos basandose en un flowchart.
- Propone comenzar con una lista de decisiones que pueden cambia.
- Cada módulo es diseñado para ocultar las decisiones enumeradas.
- El paper fue muy reelevante en su época, hoy en día son conceptos que parecerían ser naturales.