Estilos arquitectónicos

Solución 1: Funcional

- Se descompone el problema de acuerdo con las funciones básicas: entrada, *shift*, ordenar, salida.
- Un programa principal coordina el flujo de control llamando a cada una de las subrutinas.
- Los datos se comunican entre las subrutinas mediante almacenamiento compartido utilizan−do un protocolo simple de lecto−escritura.

Key word in context

- Cambios posibles al sistema:
  - El algoritmo de procesamiento: por ejemplo los movimientos circulares pueden hacerse al leer cada línea, una vez que se las leyó a todas, etc.
  - La representación de los datos
  - Las funciones del sistema: hacerlo interactivo, eliminar movimientos circulares de preposiciones
  - Desempeño: tiempo y espacio
  - Reuso de los componentes

Solución 1: Funcional (2)

- Ventajas
  - Eficiencia en la representación y acceso a los datos
  - La solución es atractivamente intuitiva
- Desventajas
  - Si se cambia la representación de los datos hay que modificar casi todos los módulos
  - Ídem para un cambio del algoritmo o de la funcionalidad
  - Los módulos no son particularmente reusables.

Solución 2: OO o TAD

- La descomposición es igual a la de la solución anterior
- Ahora los datos no son directamente accedidos por los módulos
- En su lugar cada módulo provee una interfaz que permite a otros componentes acceder a los datos sólo a través de la invocación de subprogramas en esa interfaz.

Solución 3: Tubos y filtros

- Se descompone el problema en cuatro filtros: entrada, *shift*, ordenar y salida.
- Cada filtro procesa los datos y los pasa al siguiente en la línea.
- Control distribuido: cada filtro puede ejecutar siempre que tenga datos con los que trabajar.
- Los datos que se comparten se limitan estrictamente a los que viajan por los tubos.

Solución 2: OO o TAD (2)

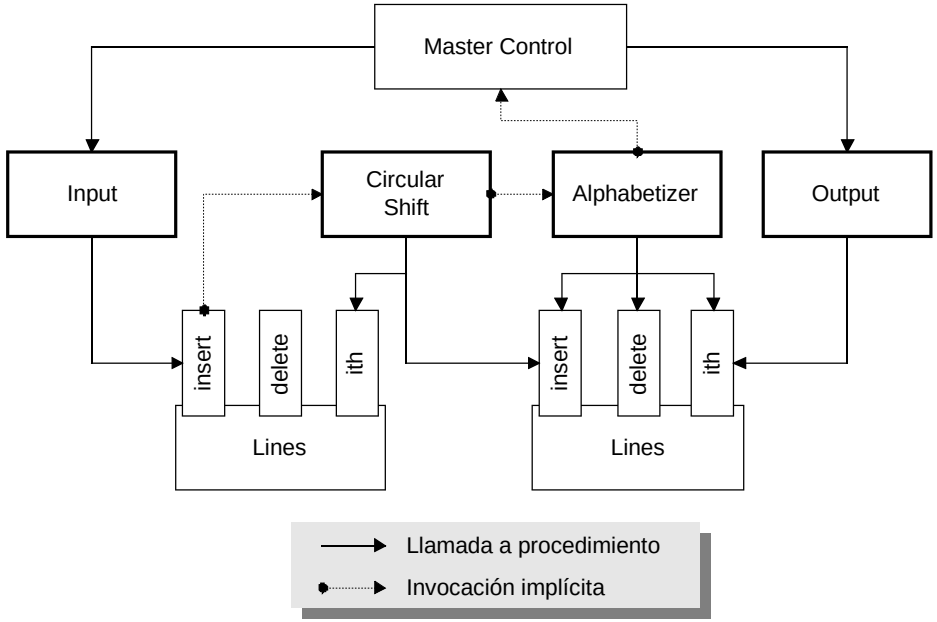
- Ventajas
  - Los algoritmos y los datos pueden cambiarse en cada módulo sin afectar a los otros.
  - Los módulos son más reusables debido a que hacen menos suposiciones sobre los otros.
- Desventajas
  - No acepta la inclusión de ciertas funciones
    - Se modifican módulos existentes haciéndolos más complejos, o se agregan nuevos módulos empeorando el desempeño.
  - No permite alterar el algoritmo general de procesamiento

Solución 3: Tubos y filtros (2)

- Ventajas
  - Mantiene el flujo de procesamiento intuitivo
  - Los filtros son altamente reusables
  - Se pueden agregar nuevas funciones
  - Se puede modificar pues los filtros son independientes
- Desventajas
  - No es posible tener un sistema interactivo
    - Para borrar una línea requeriría que los filtros compartan un almacén de datos compartido
  - Uso de espacio ineficiente: cada filtro debe copiar todos los datos a sus puertos de salida

Solución 4: Invocación implícita

- Los componentes se integran en base a datos compartidos pero con una interfaz más abstracta.
- Además, las subrutinas son invocadas implícitamente a medida que los datos son modificados.
  - Al añadir una línea se envía un evento al módulo CircularShift el cual comienza a hacer los *shifts* en otro repositorio compartido con interfaz abstracta.
- Los módulos tienen asignada la misma funcionalidad que en las otras soluciones.



Solución 4: Invocación... (3)

- Ventajas
  - Nuevos módulos pueden agregarse al registrarlos para ser invocados a medida que cambian los datos
  - La representación de los datos puede ser alterada
  - Reuso: los módulos sólo esperan eventos externos
- Desventajas
  - El orden de proceso es difícil de controlar
  - Tiende a utilizar más espacio debido a que las invocaciones son guiadas por los datos

Comparaciones

- Se consideran los cambios propuestos al comienzo y se comparan las 4 soluciones con respecto a cada cambio.

	Funcional	***TADs***	***Tubos y Filtros***	***Invocación Implícita***
Algoritmo	–	–	+	+
Datos	–	+	–	–
Funciones	+	–	+	+
Desempeño	+	+	–	–
Reuso	–	+	–	+

Definición y clasificación de estilos

Pueden determinarse mediante	Proveen en general
un conjunto de tipos componente	vocabulario de diseño
una disposición topológica de los componentes que indique sus relaciones en tiempo de ejecución	reglas de diseño o restricciones que fijan las composiciones permitidas
un conjunto de restricciones semánticas	interpretación semántica de manera tal que las composiciones permitidas tienen significado formal
un conjunto de conectores	análisis que pueden hacerse sobre sistemas desarrollados en el estilo en cuestión

¿Cómo clasificar los estilos?

Criterios de clasificación: interacciones de control y datos entre componentes

- Las clases de componentes y conectores que son usadas en el estilo
- ¿Cómo se comparte, aloja y transfiere el control entre los componentes?
- ¿Cómo se comunican datos en el sistema?
- ¿Cómo interactúan el control y los datos?
- El tipo de razonamiento permitido

Aspectos del control: cómo el control pasa de un componente a otro y cómo estos trabajan juntos por momentos.
Topología, dirección
Sincronización
Binding time

Aspectos de los datos: cómo los datos se mueven por el sistema
Topología, flujo de datos
Continuidad
Modo: pasaje, broadcast, ...
Binding time

Aspectos de la interacción control/datos:
Isomorfismo entre las topologías
Direcciones de movimiento relativas

Clasificación de estilos

Flujo de datos
Batch
Tubos y filtros
Control de procesos

Llamada a procedimiento
Programa principal y subrutina
Sistemas OO
Estratos jerárquicos

Eventos y Procesos
Procesos comunicantes
Invocación implícita

Sistemas de información
Bases de datos
Blackboard systems
Cliente−Servidor

Componentes

- Componente: entidades computacionales activas que cumplen tareas por medio de cómputos internos y comunicaciones externas.
- La relación entre un componente y su entorno se define a través de un conjunto de puntos de interacción llamados puertos.
- Los puertos pueden representar colecciones de procedimientos, conjuntos de eventos, protocolos, etc.

Conectores

- Conectores: definen la interacción entre componentes. Cada uno provee una forma para que varios puertos establezcan contacto y define lógicamente el protocolo de interacción.
- Cada conector posee una interfaz que consiste un un conjunto de *roles*.
- Cada rol define el comportamiento esperado de cada uno de los participantes en una interacción.

Llamada a procedimiento no alcanza

- El estado del arte de las notaciones y técnicas para describir diseños basados en llamada a procedimiento no tienen la suficiente riqueza como para describir muchos sistemas.
- Uno de los problemas fundamentales es que el conector (llamada a procedimiento) no es una entidad de primera clase; es decir, no está permitido describir interacciones que usen otros conectores.

Llamada a procedimiento no alcanza (2)

- Otro problema del DOO (el estilo de diseño basado en llamada a procedimiento más usado, estudiado y desarrollado en la actualidad) es que los componentes presentan una única interfaz a todos los otros componentes.
- Estos dos problemas tornan innecesariamente complicada la descripción de arquitecturas usando las notaciones clásicas de DOO como UML.

Llamada a procedimiento no alcanza (3)

- Nuevamente usaremos KWIC para ver que de existir otros conectores sería posible introducir ciertos cambios en el diseño de KWIC que el estilo OO no soporta adecuadamente.
- Queremos agregar la posibilidad de que los *shifts* que comienzan con ciertas palabras sean omitidos (y no sean indexados).

Omitir *shifts* en KWIC

- Filtrar en Output: directa pero ineficiente, hay que hacer y ordenar todos los *shifts*.
- Agregar un nuevo módulo de filtrado que sea llamado después de CircularShift y antes de Alphabetizer: menos ineficiente que la anterior.
- Alterar el algoritmo de cssetup de CircularShift de manera tal que filtre las líneas prohibidas: la decisión de omitir una línea no debería ser responsabilidad de este módulo.

Omitir *shifts* en KWIC (2)

- ¿Reescribir cssetup? ¿Modificarla? ¿Es eso lo que se persigue?
- La principal desventaja es que se destruye el criterio de modularización al incluir responsabilidades en el TAD que poco tienen que ver con su comportamiento esencial.
- El problema no está en que Parnas haya diseñado mal, sino en que no disponía de mejores conectores.