# Errores comunes de Diseño en el TP2

Algoritmos y estructuras de datos II

12 de octubre de 2016

# Repaso de la teórica

## Documentación de un proyecto de software

- Especificación:
  Al especificar nos preocupamos por el qué y no por el cómo.
- Diseño completo:
  Al diseñar es cuando nos preocupamos del cómo.

El diseño es un proceso **iterativo**. Cambiamos de mundos, y lidiamos con el *contexto de uso* y *requerimientos de eficiencia*.

## Conceptos importantes de diseño en Algo2

- Abstracción (encapsular, ocultar información, generalizar)
- Modularización (división en subproblemas)

# Repaso: Abstracción
## Representación e interfaz

### Tareas de abstracción

- Definir la estructura, el Rep y el Abs.
- Ocultar información mediante la interfaz
- Generalizar mediante los parámetros formales
- Darle *sentido* a un módulo a través de su uso y no sus datos

### Cuidado!

- No romper el Rep (o tratar de mantenerlo) fuera del módulo
- Generalizar sólo cuando corresponda o convenga
- No acceder a la estructura de representación desde afuera

# Repaso: Modularización
## Servicios usados y exportados

**Uno de los pasos más importantes del método de diseño es:**

- Definir qué módulos usarán a qué otros módulos.
- Cada módulo dirá *qué hace* y *cuántos recursos necesita*, pero ocultará *cómo lo hace*.

**Cuidado!**

- **NO** deben quedar módulos sin diseñar
- Todo debe encajar como en un rompecabezas
  - Un módulo podrá ser cambiado por otro que "encaje", es decir, que sus servicios usados y exportados sean compatibles con los módulos de los cuales depende o dependen de él en el sistema.

# Abstracción y modularización

## Ventajas del ocultamiento, la abstracción y el encapsulamiento

- La implementación se puede cambiar y mejorar sin afectar su uso.
- Ayuda a modularizar.
- Facilita la comprensión.
- Favorece el reuso.
- Los módulos son más fáciles de entender.
- Y de programar.
- El sistema es más resistente a los cambios.

Aprender a elegir buenas descomposiciones no es fácil. Ese aprendizaje comienza ahora y continúa en Ingeniería del Software I.

Diapo sacada de la teórica de Fernando Schapanik :-)

# Errores comunes de diseño en el TP2

## Los tipos de errores se presentan por:

1. Romper encapsulamiento
2. Falta de abstracción
3. Problemas de modularización

## Algunos errores concretos

- "Hablar" de la estructura interna en la interfaz del mismo módulo o desde otro distinto
- Usar punteros en vez de iteradores
- Usar géneros de TADs o funciones de TADs no implementadas como algoritmos
- La interfaz depende de la representación interna
- Hay operaciones de módulos (algoritmos) en las Pre/Post
- Falta justificar el orden complejidad de los algoritmos
- Parte de la representación es diccA(K,conj) y luego hablar en la estructura interna del diccA sobre conj
- Usar en los algoritmos módulos sin decir que diseño tienen

# Checklist para el TP2

## Extracto de la "TP2 Checklist" link

- Los géneros que aparecen en las **operaciones** corresponden a **géneros de módulos**
- Los géneros que aparecen en las **estructuras** de representación corresponden a **géneros de módulos** o renombres de tuplas.
- Los géneros que aparecen en los **"se explica con"** corresponden a **géneros de TADs**
- En las Pre y Post los parámetros son tratados como los TADs con los que se explican.
- **NO** existen módulos sin un diseño documentado
  - Los módulos usados en los algoritmos y en la representación, ya sean de la cátedra o no, tienen un diseño y figuran en la sección servicios usados"
- **Todos** los algoritmos no triviales poseen una justificación de su complejidad
- Las operaciones auxiliares (privadas), si las hay, tienen Pre, Post, Complejidad (y su justificación), Aliasing y **no** figuran en la interfaz

# Algunas FAQ de cuatrimestres anteriores... I

- "Trie", "AVL", "Hash", ¿son buenos nombres para un módulo?
  *Depende de la interfaz. La pregunta clave es: ¿con qué TAD se explica la interfaz del módulo? Si la respuesta es algo como ?Conj? o ?Dicc?, probablemente no sea un buen nombre. Un módulo llamado ?Trie?, ?AVL? o ?Hash? puede tener sentido como abstracción intermedia (entre algo como Conj o Dicc y los punteros o arreglos). Pero no son buenos nombres, en el contexto de la materia, para módulos cuyas interfaces en realidad se explican con los TADs Diccionario, Conjunto, Multiconjunto, etc.*
- "DiccTrie", "ConjAVL", ¿son buenos nombres para un módulo?
  *Es deseable que los nombres de módulo no revelen la representación interna, a no ser que sean concebidos explícitamente con esa intención. Romper el encapsulamiento en los hechos (por ej: usar los nodos de un AVL desde el módulo de arriba, que no debería saber nada sobre esos nodos) es un error grave y frecuente. Por eso preferimos ponernos un poco puristas con los nombres de los módulos: porque, si ya desde el nombre, un módulo expone*

# Algunas FAQ de cuatrimestres anteriores... II

*públicamente su estructura de representación, eso suele ser síntoma de que hay algo importante que no están entendiendo. No siempre es fácil hallar un buen nombre para un módulo. Cada grupo debería preguntarse qué nombre(s) son más adecuados para usar en su TP2, enfrentarse con este problema y resolverlo de algún modo. Ante la duda, lo mejor es que justifiquen brevemente qué alternativas consideraron y por qué eligieron la que eligieron. Nos importa más la justificación que el nombre concreto elegido.*

- La complejidad de eliminar un elemento cualquiera de un Heap, puede llegar a ser $O(\log n)$?
  *No, la complejidad de eliminar un elemento arbitrario, en peor caso, es O(n) porque hay que buscarlo primero O(n) y luego restablecer el invariante del Heap O(log n), por ende, todo es O(n).*

# Algunas FAQ de cuatrimestres anteriores... III

- Tengo Puntero(algo) por todos lados en mi estructura. Algo anda mal, no?
  *Quizás necesitan utilizar un iterador de diccionario o conjunto y/o además analizar si realmente lo necesitan. Muchas cosas se pueden resolver sin iteradores dentro de la estructura, con excepción de las funciones que tienen que devolver iteradores. Igualmente, para un Trie, ABB, AVL van a necesitar punteros sí o sí.*
- Cuánto cuesta crear un arreglo dimensionable o estático de tamaño N?
  *Cuesta O(N).*
- Puedo usar un módulo "Arreglo" como el TAD del apunte de TADs básicos?
  *Si bien el TAD Arreglo Dimensionable está en el apunte de TADs básicos, el módulo homónimo NO está en el apunte de módulos básicos. Esto es porque es un módulo especial, que viene "gratis de fábrica". En particular, como vimos en clase, es un tipo primitivo pero NO se comporta como un tipo básico (en particular, no se*

# Algunas FAQ de cuatrimestres anteriores... IV

*puede copiar un arreglo en O(1), así que por defecto se usan referencias). Son provistos por nuestro lenguaje de diseño de forma análoga a como C++ provee arreglos (estáticos y dinámicos). Para más detalles ver el apéndice A.3 del apunte de diseño. Ver también el módulo Vector de los módulos básicos.*

- Los arreglos dimensionables tienen alguna función que permita obtener la longitud?
  *Por lo que figura en el apunte no y deberían mantener una variable aparte con el tamaño. Los arreglos dimensionables son provistos por nuestro lenguaje de diseño de forma análoga a como C++ provee arreglos (estáticos y dinámicos). A efectos del TP pueden asumir que tienen la función longitud en O(1). Para más detalles ver el apéndice A.3 del apunte de diseño. Ver también el módulo Vector de los módulos básicos.*

# Algunas FAQ de cuatrimestres anteriores... V

- Tengo un Conjunto(tupla$< A, B >$) y siempre busco por la primera componente A de la tupla para ver la segunda B. Esta bien esto?
  *Esto podría estar enmascarando la necesidad de utilizar un diccionario(clave,significado), vean si es este el caso.*
- Se pueden desestimar los costos de eliminación de elementos?, con lo cual se pueden ignorar en el cálculo de complejidades?
  *No se puede desestimar el costo de buscar un elemento. Sólo de borrarlo cuando se lo encontró. Agrego lo siguiente: tampoco se puede desestimar el costo de restablecer invariantes de estructura DESPUÉS de eliminar un elemento. Ojo.*
- El costo de agregar atrás en el modulo vector de la cátedra es O(1) en peor caso?
  *Para el TP, si se necesita calcular en peor caso agregar es O(n), que es cuando el se llena el arreglo dimensionable de su representación interna. Sin embargo, se puede justificar que agregar para este módulo es O(1) en peor caso para una secuencia de n llamadas a la misma operación. Esto entra dentro del análisis amortizado de la*

# Algunas FAQ de cuatrimestres anteriores... VI

*complejidad. Verificar que si se usa este módulo se cumpla lo pedido por el enunciado.*

- Tengo un iterador que solo elimina, puedo no implementar siguiente del iterador?
  *Si es el caso de un iterador complejo sobre árboles, pueden implementar solamente las funciones que necesiten.*

- Hace falta hacer las pre y post de las algoritmos auxiliares que uso dentro de mi representación? Tengo que hacer un nuevo TAD para esto?
  *Todos los algoritmos deberían tener su pre y post. También, recuerden que pueden utilizar lógica de primer orden en las pre y post. Ahora, si el algoritmo involucra punteros y especificar su pre o post es muy complejo, pueden dejarlas expresadas en palabras. Por otro lado, si no pueden especificar la pre o post debido a que no hay ninguna función de los tads para ello, quizás deberían considerar extender el TAD en cuestión mediante otras operaciones o definir uno nuevo. Pensar bien si esto es necesario.*

# Algunas FAQ de cuatrimestres anteriores... VII

- Se puede suponer que tenemos una operación que verifica sin un nat es módulo 2, en O(1)?
  *Pueden asumir en el contexto de la materia que las operaciones algebraicas sobre naturales son O(1) y están ya definidas. Definirlas por uds. mismos es un buen ejercicio también!*

- Diseñé mi propio diccionario, tengo que implementar todas la funciones exportadas del TAD?
  *No, solo las correspondientes a los generadores y observadores más las que necesiten en su TP.*

- ¿Cómo se escribe un Rep sobre una estructura que usa punteros?
  *Cuando hay punteros involucrados, pueden optar entre describir sus propiedades formalmente o hacerlo en castellano (lo más preciso que puedan). Como se sientan más cómodos. Cuando la estructura usa punteros de manera no trivial (como por ejemplo un árbol!), suele ser complicado expresar todo lo que hay que decir (por ejemplo cosas como "no hay ciclos"). Hacerlo formalmente a veces puede resultarles bastante difícil. Hacerlo en castellano parece más fácil?*

# Algunas FAQ de cuatrimestres anteriores... VIII

*pero hacerlo bien en castellano no es realmente tanto más fácil. :-)*
*Para hacerlo formalmente pueden suponer un TAD Puntero(alfa) razonable, con la constante NULL, operaciones como *, y las convenciones usuales (por ejemplo, $p \rightarrow f$ se lee (*p).f).*

# Recursos de LaTeX para el informe

- Latex del apunte de módulos básicos: link
- Pseudo-códigos y fuentes en Latex:
  `http://www.ctan.org/pkg/algorithm2e`
- Entorno algorithm de LaTex:
  `http://www.ctan.org/pkg/algorithms`
- Estilo del Cormen: link1 link2
- Código fuente: `http://www.ctan.org/pkg/listings`

# Fin

## A tener en cuenta...

- Entrega electrónica: miércoles 25 de mayo hasta las 22hs.
- Entrega impresa: viernes 27 de mayo 17hs.
- Ver normativa de entrega: link
- Verificar toda la "TP2 Checklist" antes de la entrega: link
- Leer apunte de diseño: link
  - En particular el Apéndice B "Documentación de diseño"

# Fin

Suerte con la entrega!