Introducción al Diseño de Sistemas

por Fernando Dodino

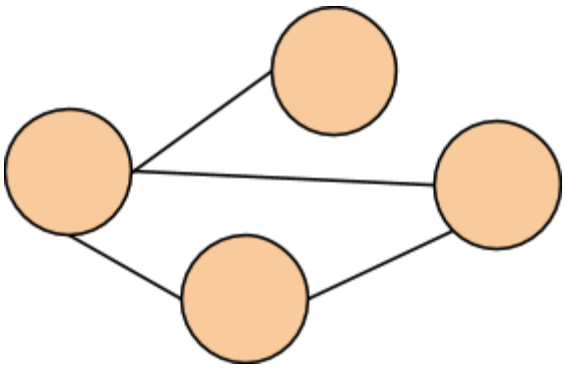
Versión 1.1

Marzo 2014

Índice

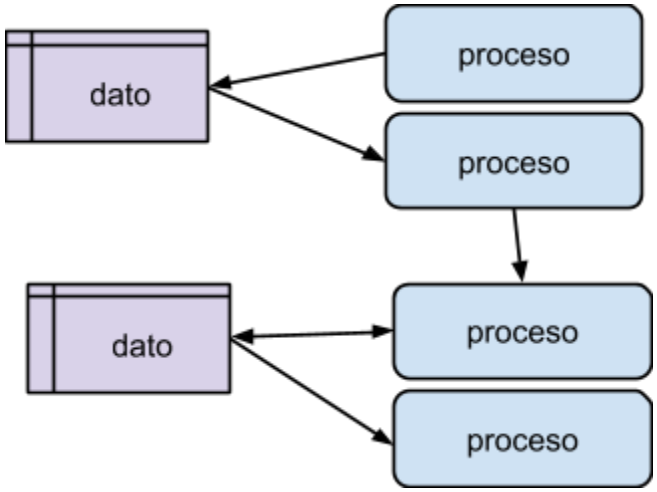
1 ¿Qué es diseñar?

- Buscamos abrir el juego, nada más, ver qué significa para cada uno: "pensar una solución", "resolver un problema", "decidir".
- Detenerse a pensar un poco qué es diseñar nos ayuda a entender el scope de la materia: lo que es esencial y lo que es tangencial.
- Tiramos finalmente una definición para orientar lo que buscamos: "**diseñar es tomar decisiones sobre los componentes de un sistema**", en particular
    - encontrar cuáles son esos componentes
    - qué responsabilidad tiene cada componente (qué va a hacer, qué no puede resolver, qué va a necesitar que otro le resuelva)
    - cómo se relaciona cada componente con los demás para formar un sistema
    - repasamos la definición de un sistema como "conjunto de partes que se relacionan para lograr un objetivo común".
- Y separando el pizarrón en cuatro partes, vemos que:
    - en el paradigma de objetos, los **componentes** son... *objetos* (a bajo nivel, marcamos los componentes con círculos), las **responsabilidades** están definidas por la *interfaz* del objeto. Recordamos interfaz vs. implementación; interfaz es lo que le puedo pedir a un objeto que haga (mensaje), la implementación es cómo lo termina resolviendo (método). **¿Cómo se relacionan los objetos?** Claro, enviándose mensajes. Pero para eso se tienen que conocer. ¿Y cómo los conozco? Mediante una *referencia* temporal (variable local, parámetro) o más permanente (variables de instancia o de clase, más allá de las variables globales que no nos interesa tratar).



    - en el paradigma estructurado tradicional, sí ese de C y Pascal, los **componentes** son *datos* y *funciones, procedimientos o procesos (los marcamos con cuadrados)*, a cada función o procedimiento le asignamos una **responsabilidad** (qué objetivo/s cumple/n) también demarcada por la interfaz. Los datos son estructuras más bien pasivas, que esperan ser afectadas por los procesos. ¿Cómo es la **relación** entre datos y funciones? Las funciones usan

estructuras de datos y también delegan responsabilidades a otras funciones (principalmente mediante "calls").

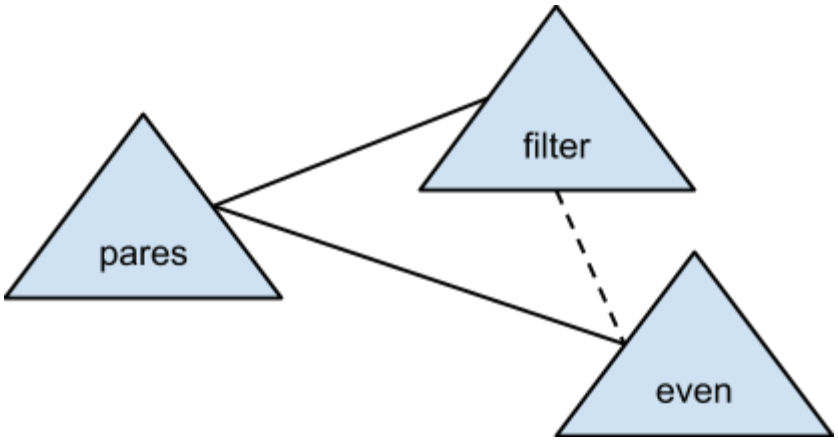


- en el paradigma funcional tenemos una distribución muy similar al diseño en el paradigma imperativo: los **componentes** son datos y *funciones* (los marcamos con triángulos). Cada *función* tiene una **responsabilidad.** ¿Se acuerdan cómo se **relacionan** las funciones entre sí? Preguntamos al curso a ver si sale: *orden superior* y *composición* son las formas más comunes.

Pasando en limpio:

```
pares ns = filter even ns     | ns -> dato; filter, even -> funciones
o
pares = filter even
```

muestra que pares está definido en función de filter + even. Por supuesto, también hay acoplamiento entre even y filter aunque el grado de dependencia es mucho más débil.



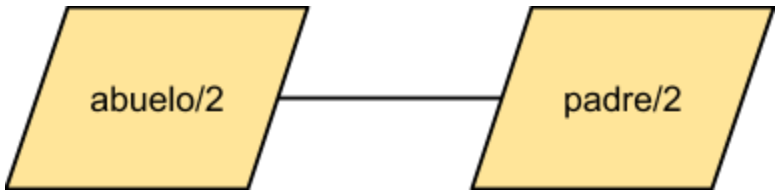
Nota: en este gráfico no incluimos adrede los componentes de datos.

- y por último en el paradigma lógico tenemos *predicados e individuos* como **componentes** (los marcamos con un... paralelogramo). Claramente cada *predicado/cláusula* tiene una **responsabilidad**, y las relaciones entre predicados se da mediante el orden superior y la utilización de predicados como antecedentes de reglas.

Para ponerlo en limpio:

```
abuelo(Abuelo, Nieto):-padre(Abuelo, Padre), padre(Padre, Nieto).
```

marca una relación entre el predicado `abuelo/2` y `padre/2`.



Aquí tampoco graficamos los componentes de datos.

2 Diseño de sistemas y diseño de software

Preguntamos a la clase: ¿cómo se relacionan entre sí el diseño de sistemas y el diseño de software? ¿qué abarca cada concepto?

- El diseño de software está englobado en el diseño de sistemas, que tiene características más generales. Podemos diseñar:
    - el sistema electoral que determina el ganador de una elección, un sistema de inscripción a la facultad, un sistema que entrega documentos de identidad a los ciudadanos,
    - sistemas de control: ascensores, puentes, cámaras de vigilancia, dispensers, brazos robóticos, etc.
    - y el software que puede hacer de soporte tecnológico a casi todos los sistemas que mencionamos anteriormente.

El diseño de software es particularmente interesante por sus características no tangibles. *Por eso la mayoría de los ejemplos de las clases siguientes van a estar enfocados en resolver necesidades de productos de software.*

¿Qué dicen los libros a todo esto?

- Edward Yourdon diferencia los sistemas automatizados, los manuales y los que tienen un mix de componentes manuales y automatizados. También enumera una serie de

razones por los que no debieran automatizarse algunos sistemas de procesamiento de información: costo, capacitación a usuarios finales, políticas, etc. [1][1]

- Ian Sommerville dice: "La ingeniería de sistemas se refiere a todos los aspectos del desarrollo y de la evolución de sistemas complejos donde el software desempeña un papel principal. Por lo tanto la ingeniería de sistemas comprende el desarrollo de hardware, políticas y procesos de diseño y distribución de sistemas, así como la ingeniería de software." [2]

3 Relaciones del diseño con otros conceptos

3.1 Diseño y tecnología

Hay algunas ideas instaladas tanto en el mercado laboral como en el ámbito académico, las vamos a anotar en el pizarrón:

- "Se puede diseñar la funcionalidad sin conocer la tecnología"
- "Si estoy diseñando y tiro una línea de código, dejé de diseñar"
- "Diseñar implica saber UML / saber Enterprise Architect / hacer cajitas".
- “Se puede programar sin necesidad de diseñar”

Nuestro pensamiento es que

- cuando no bajamos a detalle,
- cuando ignoramos cuestiones esenciales de la tecnología (arquitectura, paradigmas y lenguajes a desarrollar, etc.),
- cuando nos queremos abstraer de la programación y la consideramos una tarea menor, "operativa", donde no se piensa

estamos "generando una deuda", porque si diseñar es tomar decisiones, descartar decisiones técnicas produce un diseño pobre. Y decimos que diseñar no es hacer diagramas, hacer diagramas es documentar las decisiones que tomé en otro momento. Si no hice el diagrama no quiere decir que no diseñé, en todo caso quiere decir que no está documentado, esa es otra discusión.

Por otra parte, cuando pensamos que el diseño implica detallar cada una de las decisiones, posiblemente nos estemos obligando a tomar esas decisiones en forma prematura, donde no siempre tenemos toda la información o el contexto para definir apropiadamente los componentes. Más adelante trabajaremos en este aspecto.

Tiramos un ejemplo en clase: el profesor le pide a un alumno que construya una escalera de madera. ¿Qué preguntas se hace el profesor?

1. qué cualidades quiero que esa escalera tenga (que sea estable, que sea durable, que

[1] Capítulo 2: "La naturaleza de los sistemas", Edward Yourdon, Análisis Estructurado Moderno, Editorial Pearson Addison-Wesley

[2] Ian Sommerville, *Ingeniería del Software 7a.edición*, Editorial Pearson Addison-Wesley, cap.1 pág.7

se construya rápidamente, qué altura debe tener, cuántos escalones)
2. qué materiales debo utilizar (definido por la decisión anterior) y cómo voy a especificarlo
3. al pensar alternativas aparecen restricciones (*constraints*): de tiempo, de costo, de material que el cliente está dispuesto a comprar, de factibilidad técnica (el grado de experiencia del alumno o sus habilidades con determinadas herramientas afectan en la decisión del tipo de escalera que voy a construir), etc.

En conclusión, el profesor no puede hacer su trabajo sin tener un profundo conocimiento del arte de la carpintería, de cómo hacer un corte, de las técnicas disponibles. No se puede separar la teoría de la carpintería de la práctica. Y en toda decisión tenemos que entender los objetivos, priorizarlos, ver los pros y las contras. Entender cada una de nuestras decisiones a cuáles de nuestros objetivos se aplican.

Como vemos, tomar decisiones sin tener en cuenta el juego de variables que intervienen tiene un costo: en el caso del diseño hay restricciones que impone el usuario, la arquitectura, el paradigma y el lenguaje de programación en el que vamos a implementar y que no es posible soslayar: si estamos haciendo un sistema de reclamos para la municipalidad ¿cómo diseñamos el reclamo de un ciudadano? ¿cómo es posible hacerlo sin saber si el sistema será web o cliente servidor, orientado a objetos o con programación funcional, en un lenguaje que hace chequeo de tipos estático o en otro que no tiene testing automatizado?

4 ¿Qué podemos esperar de la materia?

- Entender cuál debe ser nuestro rol como diseñadores en un proyecto
- Reflexionar sobre el proceso de diseño y sobre la responsabilidad de un trabajo profesional que exige pensar alternativas para tomar decisiones que cumplan determinadas cualidades
- Adquirir nuevas herramientas de diseño
- Implementar ideas de diseño
- Aprender a diseñar tomando en cuenta diferentes puntos de vista
- y por supuesto... divertirnos en ese camino, ¡bienvenidos!