UTN.BA
DPTO. INGENIERÍA EN SISTEMAS DE INFORMACIÓN
CÁTEDRA DISEÑO DE SISTEMAS

# EL DISEÑO DE SISTEMAS

**Qué es y qué decisiones implica**

Autor: Ing. Luciano Straccia

# Tabla de contenido

# 1. El análisis y el diseño de sistemas de información

Considerando que en toda la bibliografía se considera al diseño como una etapa posterior al análisis, se hace necesario definir el alcance del análisis para, a partir de ello, poder comprender las actividades que continúan a dicha etapa.

La visión respecto al análisis y al diseño la hemos puesto de manifiesto en un nuevo trabajo llevado a cabo por diversos colegas de la ingeniería y el ámbito académico[1] donde se afirma que en el análisis de requisitos del sistema "se define la funcionalidad del software así como las restricciones del mismo. También se descubren las características de funcionamiento o ejecución que debe tener un sistema o componente de un sistema (tiempo de respuesta, tiempo de servicio, requisitos de seguridad, requisitos de interfaz, portabilidad, estándares)" [2], mientras que en el diseño "se traducen los requisitos en una representación del software atendiendo a la tecnología que se va a utilizar (sistema operativo, lenguajes de programación, etc.). Se debe definir base de datos, módulos o componentes, arquitectura, interfaces y documentos"[3].

En dicha bibliografía se hace referencia al software exclusivamente (dado el objetivo de la misma), sin embargo debe ampliarse el concepto de análisis y diseño a otros sistemas de información que no necesariamente sean informatizados. No es interés de este texto abordar las diferencias entre "sistemas de información" y "software" por lo cual no me detendré en ello y consideraré que un software es un subconjunto de los sistemas de información.

Tal como se afirma anteriormente el análisis implica la definición de las características requeridas por el sistema a construir. Estos requerimientos pueden ser categorizados en funcionales y no funcionales. Si bien los requerimientos no funcionales no serán materia prima del análisis, en general son recolectados y definidos en la misma instancia de relevamiento y análisis que los requerimientos funcionales. Los requerimientos funcionales posteriormente forman parte del proceso de análisis propiamente dicho y el modelo del sistema.

---

[1] Straccia, Luciano; Zanetti, Paula (coords.) (2016). Alcances y usos de la tecnología de la información en las organizaciones. Editorial CEIT. Universidad Tecnológica Nacional. Buenos Aires. En edición.

[2] Garbarini, Ramiro; Saavedra Martínez, Paola. En Straccia, Luciano; Zanetti, Paula (coords.) (2016). Alcances y usos de la tecnología de la información en las organizaciones. Capítulo 7. Editorial CEIT. Universidad Tecnológica Nacional. Buenos Aires. En edición.

[3] Ídem

El análisis de los sistemas implica elaborar los modelos conceptuales de los sistemas de información, modelar las características intrínsecas de los sistemas de información y seleccionar adecuadamente los modelos que mejor se adapten para dar soluciones a los problemas de información[4]. Para continuar con el recorrido hacia qué es el diseño, voy a plantear (para acordar con el lector) que el relevamiento y análisis brindarán una serie de resultados que serán utilizados por el diseño:

- Requerimientos funcionales definidos
- Requerimientos no funciones definidos
- Modelo de clases basado en características del dominio del problema[5], generalmente expresado en un Diagrama de Clases
- Modelo de datos basado en características del dominio del problema, generalmente expresado en un Diagrama de Entidad-Relación

Desde una perspectiva más teórica que pragmática, el modelo de datos puede ser realizado aun cuando no implique el desarrollo de software, ya que podría construirse como solución, por ejemplo, un formulario preimpreso y en él es relevante plasmar un buen análisis de cuáles son los datos requeridos y su estructuración. Por ello es que hasta aquí no realicé diferencias entre el análisis de sistemas de información y el análisis de software, aunque esta diferencia aparece mucho más presente en la idea de modelo de clases.

[4] Programa de la asignatura Análisis de Sistemas. Universidad Tecnológica Nacional, Facultad Regional Buenos Aires (UTN-FRBA). Disponible en www.sistemas.frba.utn.edu.ar

[5] En ocasiones he tenido algunos debates con colegas respecto de la diferencia entre modelos de negocio y modelos de análisis, tal como plantea RUP. Sin embargo, la ingeniería del software en la práctica actual no diferencia entre un aspecto y otro y el rol de todo analista es generar un modelo donde se pongan de manifiesto las características del sistema de información en el dominio del problema.

# 2. El diseño de sistemas

Pueden hallarse algunas definiciones en las cuales queda claro que no es posible desanclar el diseño de sistemas de una mirada superadora al diseño de software en sí.

Ian Sommerville afirma que "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"[6].

Rumbaugh sostiene que el diseño “es la estrategia de alto nivel para resolver problemas y construir un solución. Éste incluye decisiones acerca de la Organización del sistema en subsistemas, la asignación de subsistemas a componentes de hardware y software, y decisiones fundamentales conceptuales y de política que son las que constituyen un marco de trabajo para el diseño detallado.”[7]

En tanto, una definición más antigua del MIT define al diseño como “el proceso de aplicar distintas técnicas y principios con el propósito de definir un dispositivo, proceso, o sistema, con los suficientes detalles como para permitir su realización física”[8].

Diseñar sistemas de información involucra, además del software, diseñar las soluciones asociadas a procedimientos, sistematizaciones no informatizadas, documentación, etc. Sin embargo, a fines de la asignatura Diseño de Sistemas en la carrera, estos aspectos son cubiertos desde otras asignaturas que abordan estos contenidos. Algunos ejemplos de esto puede hallarse en que la sistematización no informatizada que no implica construir instrumentos sino simplemente plantear la sistematización es abordada en Análisis de Sistemas al momento de modelar los sistemas de información; y el concepto de procedimiento es abordado tempranamente en Sistemas y Organizaciones sobre el modelo de cursogramas[9] y posible de ser retomado en asignaturas vinculadas a la Gestión de Calidad y la Ingeniería del Software.

---

[6] Ian Sommerville, Ingeniería del Software 7a.edición , Editorial Pearson AddisonWesley, cap.1 pág.7

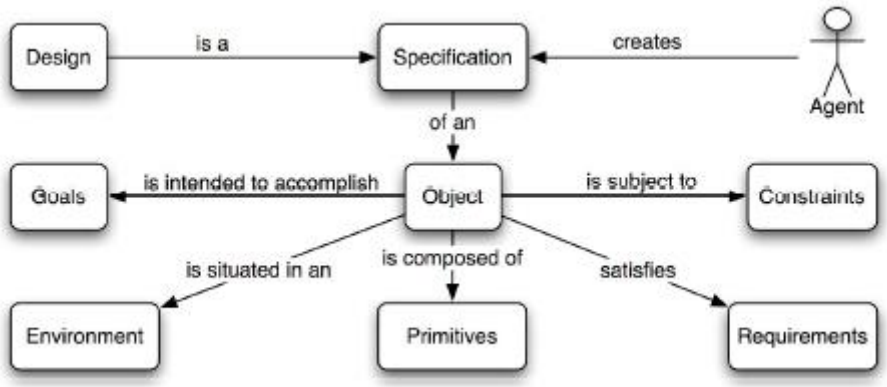
[7] Rumbagh, James y otros. Modelado y Diseño Orientados a Objetos. Ed. Prentice Hall 1997

[8] E.S.Taylor, An Interim Report on Engineering Design, Massachusetts Institute of Technology, 1959

[9] El cursograma es un modelo que es utilizado en pocas ocasiones por los ingenieros en sistemas de información, sin embargo su valor sustancial en la asignatura integradora de primer nivel radica en la posibilidad de construir procedimientos y considerar artefactos (especialmente documentos) involucrados en estos procedimientos.

A partir de lo expresado en el párrafo precedente continuaré con la noción de diseño de sistemas asociado al software. Sin embargo si utilizara el concepto de "diseño de software" eliminaría la noción sistémica de múltiples software interconectados, por lo cual llamaré a esta idea, como "diseño de sistemas software" para diferenciarlos de los previamente considerados. Un sistema de software es un sistema de información automatizado (informatizado[10]) que puede contar con múltiples subsistemas y elementos que a su vez serán informatizados también.

Para comprender los aspectos que impactan sobre el diseño creo importante la figura descrita en el trabajo de Ralph y Wand[11]:




[10] Informática proviene del francés *informatique*, que es una contracción de los términos *information* (información) y *automatique* (automática).

[11] Ralph, P.; Wand, Y. A Proposal for a Formal Definition of the Design Concept.

# 3. Aspectos del diseño de sistemas

En un diseño de alto nivel "se describen los componentes principales del sistema y el modo en que interactúan entre sí para lograr los objetivos del diseño"[12]. En el diseño están involucradas las siguientes actividades[13]:

- Elección respecto a las tecnologías y elementos arquitectónicos básicos del sistema
- Componentes y sus interfaces
- Interacciones entre componentes
- Modelo de datos de los componentes e interfaces

En base a estas actividades definiré cuatro aspectos del diseño:

- Diseño arquitectónico
- Diseño de componentes de software (en el paradigma orientado a objetos, diseño de clases)
- Diseño de interfaz entre pieza de software y actores externos, generalmente llamado diseño de interfaz de usuario
- Diseño de datos

En todos los casos el diseño puede contener restricciones definidas por el proyecto, entre las cuales se involucran por ejemplo las tecnologías.

A continuación presentaré las características del diseño en cada uno de estos aspectos.

## 3.1. Diseño de datos

Considero que en el análisis se realiza un modelo de datos basado en los datos asociados al dominio del problema.

Dado que como entrada al diseño se involucran requisitos no funcionales y entre las primeras definiciones del diseño se decide la tecnología a utilizar, el modelado de datos involucra modificar el modelo de datos del análisis considerando:

---

[12] MSDN Microsoft. Modelar la arquitectura de la aplicación. Disponible en https://msdn.microsoft.com/es-AR/library/dd490886.aspx

[13] Ídem

- cambios requeridos para que al momento de ser implementado el modelo en un motor de base de datos permita cumplir con los requisitos no funcionales especificados;
- por ejemplo uno de los cambios más habituales requeridos en la etapa de diseño es la toma de decisión respecto a atributos o entidades de auditoría (definiendo si se utilizan tablas de auditoría o campos dentro de las propias tablas existentes o ambas opciones);
- si en el análisis se involucran tipos de datos, el diseño debe considerar cuáles son específicamente los tipos de datos asociados a la tecnología en la cual se persistirá el modelo (por ejemplo, una cadena de 50 caracteres en el modelo de análisis, ¿qué tipo de datos implica en el modelo de diseño?, ¿varchar, char, etc.?

Una diferencia sustancial en el modelo de datos entre el análisis y el diseño es que el primero implica la definición de datos asociados al dominio del problema y al sistema de información, mientras que en el ámbito del diseño involucra la noción de persistencia por tratarse de un modelo que finalmente será llevado a la práctica en la construcción.

Respecto de la normalización, considero que dicha actividad forma parte del análisis de los datos y la optimización del proceso de análisis y categorización de los datos y no propiamente del diseño. Esta idea proviene sobre el valor que se le otorga en el análisis a una correcta categorización, división y organización de los aspectos del problema y la realidad, por lo cual la normalización, en vistas de cumplir dicho objetivo, forma parte del análisis de sistemas. La desnormalización sí forma parte del diseño dado que su aplicación o no depende de los requisitos no funcionales definidos y la tecnología seleccionada.

Finalmente en la visión planteada en este documento respecto del modelo de datos en el análisis, el uso de bases de datos no relacionales (sobre las cuales ha versado hasta ahora lo escrito respecto al modelo de datos) no deja de requerir el modelo de datos, por lo cual será parte del diseño la decisión acerca de si se utilizan bases de datos relacionales o no relacionales para la construcción del producto solución de software.

En resumen, el diseño de datos implica:

1. decidir si se utilizarán bases de datos relacionales o no relacionales;
2. definir qué cambios se realizarán sobre el modelo de datos construidos en el análisis a partir de las decisiones tecnológicas y las necesidades basadas en el cumplimiento de requerimientos no funcionales, incluyendo especialmente decisiones respecto a aspectos de auditoría y de tipos de datos, siendo estos aspecto no excluyentes;

3. decidir si es conveniente desnormalizar estructuras de datos que fueron previamente normalizadas en el análisis;

## 3.2. Diseño arquitectónico

La arquitectura de software representa la "estructura o estructuras del sistema que consiste en componentes de software, las propiedades externas visibles de esos componentes y las relaciones entre ellos"[14]. La IEEE la define como la organización fundamental de un sistema, formada por sus componentes, las relaciones entre ellos, el contexto en el que se implantarán, y los principios que orientan su diseño y evolución[15]. En estas definiciones se pone de relieve la idea de componente.

En el marco del desarrollo de software basado en componentes (CBSE, *Componente Based Software Engineering)*, se define a un sistema como "un conjunto de mecanismos y herramientas que permiten la creación e interconexión de componentes software, junto con una colección de servicios para facilitar las labores de los componentes que residen y se ejecutan en él"[16].

Un componente es una pieza de código preelaborado que encapsula alguna funcionalidad expuesta a través de interfaces estándar. Szyperski, exarquitecto de software de Microsoft, define componente como una "unidad de composición de aplicaciones software que posee un conjunto de requisitos y que ha de poder ser desarrollado, adquirido, incorporado al sistema y compuesto por otros componentes, de forma independiente en tiempo y forma"[17].

El diseño de sistemas software en su aspecto arquitectónico involucra:

- definir cuáles serán las piezas de software que operarán como componentes del sistema integral, considerando cuáles serán desarrolladas y cuáles serán servicios, aplicaciones, etc. externas que serán utilizadas por las piezas de software desarrolladas;

---

[14] Eclipse.org. Concepto: Software Architectura. Disponible en http://epf.eclipse.org/wikis/openupsp/openup_basic/guidances/concepts/software_architecture,__O7tAMVvEduLYZUGfgZrkQ.html

[15] IEEE Std. 1471-2000.

[16] Fuentes, L.; Troya, J.M.; Vallecillo, A. Desarrollo de Software Basado en Componentes. ETSI Informática. Universidad de Málaga. Málaga, España.

[17] Szyperski, C. Component Software. Beyond Object-Oriented Programming. Addison-Wesley. (1998)

- definir la arquitectura general del sistema integral, indicando como se despliegan físicamente los diferentes componentes de software (servidores, vínculos entre servidores, repositorios, etc.) considerando en esta instancia los estilos y patrones arquitectónicos existentes;
- definir cómo se comunicarán los diferentes componentes en la arquitectura definida (por ejemplo protocolos de comunicación, si serán procesos sincrónicos o asincrónicos, etc.);
- definir la arquitectura de cada pieza de software, indicando cómo los diferentes componentes (que serán descritos en el diseño de clases) se despliegan e interactúan entre sí.

## 3.3. Diseño de clases

Considero que en el análisis se realiza un modelo de clases basado en los conceptos asociados al negocio y al sistema de información en general.

Dado que como entrada al diseño se involucran requisitos no funcionales y entre las primeras definiciones del diseño se decide la tecnología a utilizar, el modelado de clases involucra modificar el modelo de clases del análisis considerando:

- cambios requeridos para que al momento de ser implementado en un lenguaje de programación específico permita cumplir con los requisitos no funcionales especificados;
- uno de los cambios más habituales requeridos es la aplicación de patrones de diseño para favorecer el cumplimiento de los requisitos;
- además implica la definición de niveles de acoplamiento entre las diferentes clases y la definición de la responsabilidad que cargará cada clase (o qué responsabilidad será derivada a otra clase, aunque en el modelo de análisis pudiera implicar una única clase)
- si en el análisis se involucran tipos de datos para los atributos de las clases, el diseño debe considerar cuáles son específicamente los tipos de datos asociados a la tecnología en la cual se programará el modelo y modificar el modelo en caso que el lenguaje tenga tipos de datos propios (por ejemplo, los tipos de datos de las bibliotecas estándares de Java, en caso de haber seleccionado dicho tecnología).

Una diferencia sustancial en el modelo de clases entre el análisis y el diseño es que el primero implica la definición de conceptos (clases) asociados al dominio del problema y el sistema de información mientras que en el ámbito del diseño involucra la noción de elementos del software.

Por último es deseable, más allá de los requisitos definidos, considerar los principios de diseño, por ejemplo SOLID[18] para el diseño de estos componentes.

En resumen, el diseño de clases implica:

- definir qué principios de diseño serán considerados prioritariamente para la definición de componentes del software, especialmente determinando los grados de acoplamiento entre componentes y las responsabilidades asignadas a cada uno;
- definir qué cambios se realizarán sobre el modelo de clases construidos en el análisis a partir de las decisiones tecnológicas y las necesidades basadas en el cumplimiento de requerimientos no funcionales, incluyendo especialmente decidir respecto a la necesidad de aplicación de patrones de diseño por aspectos puntuales que lo requieran;

## 3.4. Diseño de interfaz de usuario

La interfaz de usuario es el componente de un software encargado de vincular a los actores externos (usuarios) del sistema con el mismo.

El diseño de una interfaz de usuario implica definir cuáles serán esos componentes (esas interfaces) y la interacción existente entre las diferentes interfaces, teniendo en cuenta que ambos aspectos deben satisfacer los requerimientos no funcionales.

El diseño preliminar de cada componente (cada interfaz de usuario) puede ser realizado mediante un wireframe, que es "una representación visual muy sencilla del esqueleto o estructura de una página web, con la que se sientan las bases del diseño y se establece su punto de partida. Por regla general, se dibuja en blanco y negro, con trazos simples y pocos detalles, ya que se compone sólo de cajas y texto. Se centra en la funcionalidad, la experiencia de usuario y en dar prioridad a los contenidos del proyecto"[19]. Si bien es un concepto generalmente utilizado en páginas web, es extensible a otro tipo de soluciones. Los wireframes también son conocidos como Prototipos de Baja fidelidad, StoryBoard, Schematics, Blueprints o Page

[18] SOLID es un acrónimo introducido por Roberto Martín sobre cinco principios de diseño: Single Responsabilty; Open-Closed; Liskov Substitution; Interface segregation y Dependency Inversion.

[19] InboundCycle. ¿En qué consisten los wireframes? Disponible en http://www.inboundcycle.com/blog-de-inbound-marketing/bid/195267/En-qu-consisten-los-wireframes

Architecture[20] y posteriormente pueden ser refinados generando un prototipo final que será la base para su construcción.

Estos componentes de interfaz de usuario interactúan entre sí, por lo cual es importante definir de qué manera interactuarán, generando un árbol de navegabilidad. La construcción de este árbol de navegabilidad permite analizar las características de navegabilidad de una aplicación y contrastarla con los requisitos de usuario (por ejemplo, si se requiriera determinado tiempo máximo para el acceso a determinada funcionalidad o el cumplimiento de determinada tarea, se puede evaluar esto a partir del análisis de un grafo generado por este árbol de navegabilidad, siendo cada interfaz de usuario sus nodos).

En resumen, el diseño de interfaz de usuario implica diseño de prototipo de interfaz de usuario y árbol de navegación con el objetivo de cumplir los requisitos no funcionales del sistema e, indirectamente, permitir la utilización del sistema en general posibilitando así cumplir con a los requisitos funcionales.

[20] StrangeSystems. Using wireframes. Disponible en http://www.strangesystems.net/archives/2005/03/using_wireframe.php

# 4. Aspectos afines al diseño de sistemas

## 4.1. La definición de requisitos no funcionales

Tal como afirmé anteriormente la definición de requisitos no funcionales no forma parte del diseño de sistemas en sí, sino que son parte de una etapa de relevamiento o, en algunas realidades empresariales, realizadas en el marco de una etapa que conjuga relevamiento y análisis. Los requisitos no funcionales operan como información de entrada para el diseño.

Dadas las falencias que los alumnos poseen en la definición de requisitos no funcionales y que éstos no constituyan meras declaraciones de deseos de usuarios, tales como “un software fácil de usar”, sino que constituyan realmente un requisito, como una característica inherente al software que deberá ser cumplida en la construcción del mismo, se considera pertinente en el marco de la asignatura trabajar sobre la definición de requisitos no funcionales basados en atributos de calidad, ahondando en modelos de especificación de estos requisitos.

## 4.2. Las pruebas de software

El testing de software es otro aspecto que no forma parte del diseño en sí mismo. Sin embargo al ser Diseño de Sistemas la primera asignatura enfocada específicamente a las soluciones informatizadas (según la visión descrita en este documento) se hace necesario poner énfasis en la importancia del testing de software, su planificación y la definición de casos de prueba. Considero que las técnicas específicas para el testing de las funcionalidades desde una perspectiva funcional superan el objetivo de la asignatura y deberán ser tratadas en la asignatura Ingeniería de Software (tanto las pruebas manuales, como las pruebas automatizadas).

Sin embargo el testing de requisitos no funcionales por su fuerte vínculo con el diseño de sistemas y porque es la actividad que permite validar el cumplimiento de los requisitos atendidos en el propio diseño, es una actividad que debe ser ahondada con técnicas específicas para este tipo de requisitos.

# 5. Resumen de decisiones de diseño

Respecto del diseño de datos:

1. decidir si se utilizarán bases de datos relacionales o no relacionales;
2. definir qué cambios se realizarán sobre el modelo de datos construidos en el análisis a partir de las decisiones tecnológicas y las necesidades basadas en el cumplimiento de requerimientos no funcionales, incluyendo especialmente decisiones respecto a aspectos de auditoría y de tipos de datos, siendo estos aspecto no excluyentes;
3. decidir si es conveniente desnormalizar estructuras de datos que fueron previamente normalizadas en el análisis.

Respecto del diseño arquitectónico:

4. definir cuáles serán las piezas de software que operarán como componentes del sistema integral, considerando cuáles serán desarrolladas y cuáles serán servicios, aplicaciones, etc. externas que serán utilizadas por las piezas de software desarrolladas;
5. definir la arquitectura general del sistema integral, indicando como se despliegan físicamente los diferentes componentes de software (servidores, vínculos entre servidores, repositorios, etc.) considerando en esta instancia los estilos y patrones arquitectónicos existentes;
6. definir cómo se comunicarán los diferentes componentes en la arquitectura definida (por ejemplo protocolos de comunicación, si serán procesos sincrónicos o asincrónicos, etc.);
7. definir la arquitectura de cada pieza de software, indicando cómo los diferentes componentes (que serán descritos en el diseño de componentes) se despliegan e interactúan entre sí.

Respecto del diseño de interfaz de usuario:

8. diseño de prototipo de interfaz de usuario y árbol de navegación con el objetivo de cumplir los requisitos no funcionales del sistema e, indirectamente, permitir la utilización del sistema en general posibilitando así cumplir con a los requisitos funcionales.

Respecto del diseño de clases:

9. definir qué principios de diseño serán considerados prioritariamente para la definición de componentes del software, especialmente determinando los grados de acoplamiento entre componentes y las responsabilidades asignadas a cada uno;
10. definir qué cambios se realizarán sobre el modelo de clases construidos en el análisis a partir de las decisiones tecnológicas y las necesidades basadas en el cumplimiento de requerimientos no funcionales, incluyendo especialmente decidir respecto a la necesidad de aplicación de patrones de diseño por aspectos puntuales que lo requieran.