Introducción al desarrollo de software

Pantallazo sobre sistemas software, su importancia y arquitectura, y sus metodologías de desarrollo

Franco Bulgarelli, Marzo 2014

Introducción

Es nuestro primer día en Diseño de Sistemas. Ya tenemos una idea sobre qué es diseñar. Pero entonces, nos preguntamos: ¿qué es exactamente lo que diseñaremos? ¿Cómo lo haremos? ¿Cómo se relaciona con la actividad, más grande, de desarrollo? A continuación, algunas respuestas a modo de introducción a cuestiones que trataremos con más detalle en otros apuntes y clases.

Sistema vs sistema software

La materia se llama diseño de sistemas, pero haremos foco en sistemas software. ¿Dónde está la diferencia? Que la noción de sistema es más abarcativa y refiere a diseño de procesos de obtención, movimiento, almacenamiento y transformación de información, aún en contextos manuales. Mientras que el sistema software es aquel que realiza esto empleando software

¿Por qué esto? Porque hace más de 40 años que la tendencia ha sido reemplazar toda operatoria manual con software, por lo que **centrarse en diseño manual es anacrónico**. Y sin embargo, los sistemas software deben resolver todas las problemáticas de manejo de la información, más aquellas específicas de la tecnología. Por lo que quien sepa construir buen software y de la manera correcta, es capaz de construir luego, de ser necesario, sistemas no software. A la inversa no sucede.

Arquitectura de un sistema software

¿Como se compone un sistema software? Es un conjunto de componentes físicos como programas y scripts (“programitas”) computadoras, redes, etc, y componentes lógicos como objetos, funciones, procedimientos, según el paradigma.

Los agrupamos en cuatro grandes categorías que llamamos componentes arquitecturales,

análogos a los que mencionamos recién en los sistemas manuales: **presentación**, integración/comunicación, **dominio** y **persistencia**.

Motores del diseño

Wow, eso es mucho. ¿Por dónde empezar? Hay distintos enfoques, están quienes proponen comenzar por diseñar el modelo de datos (diseño data céntrico), o quienes proponen iniciar por plantear la arquitectura de comunicación entre los componentes (físicos y lógicos).

Comenzar por diseñar las pantallas y la interacción con el usuario también es muchas veces una buena idea, porque nos da un diseño centrado en el usuario, y es un enfoque moderno muy fuerte en muchos ámbitos.

Nosotros iremos finalmente por el enfoque de diseñar guiado por el dominio, que plantea Eric Evans.

Diseño guiado por el dominio

¿Qué es esto? Comenzar por modelar el problema que el usuario nos plantea, más allá de como se deba presentar o persistir o integrar con el resto de la aplicación. De eso nos encargaremos luego, o al menos, en paralelo.

¿Por qué? No es necesariamente siempre la mejor opción, pero es un enfoque que nos parece válido, sobre todo cuando tenemos dominios complejos y cambiantes.

Además, para lo que es la materia, les permitirá pararse sobre lo que ya saben hasta el momento: modelar (lo haremos con objetos) y analizar problemas a partir de sus requerimientos e interacciones (plasmadas a través de historias de usuario o casos de uso o cualquier estructura similar).

Si bien el diseño es **dependiente de la tecnología**, este enfoque nos permitirá dejar las decisiones más tecnológicas para más adelante, lo que en la cursada nos es útil.

En la realidad, los cuatro enfoques son complementarios, la experiencia que ganen en esta materia y en su actividad profesional es la que les dirá luego como combinarlos.
Desconfíen siempre de los puristas.

Diseño y desarrollo

¿Y cómo trabajaremos? ¿En qué contexto se diseña el software real?

Primero, entendamos que en ningún proyecto se diseña para no construir, o se analiza para no diseñar. Siempre **el diseño está enmarcado en un proyecto de desarrollo**.

Metodologías de desarrollo

En la mayoría de los contextos reales (o al menos, en la mayoría de los que vale la pena trabajar), se trabaja de forma **iterativa e incremental**. Esto es, analizando el problema y diseñando y construyendo la solución a la par, dando pequeños pasos.

En cada uno de esos pasos se irá agregando complejidad al sistema y nos replantearemos qué es aquello de lo analizado y construido hasta ahora que nos sirve, aquello que debemos modificar, y aquello que simplemente debemos descartar.

Y lo haremos hasta tener un sistema lo suficientemente bueno.

En la naturaleza del software, comparada con las construcciones civiles, está la facilidad de construir, destruir y remodelar, entonces vamos a aprovechar sus cualidades para aplicar siempre que podamos el modelo iterativo e incremental. Porque además de aumentar nuestras chances de éxito al poder atacar más rápidamente cambios de requerimientos o errores de análisis, diseño o construcción, **reduce nuestro estrés ante la hoja en blanco**.

Factores externos

Somos de la visión de que la forma iterativa e incremental es la única manera de construir software de forma repetible y segura.

Sin embargo, también reconocemos que en ciertos contextos políticos y económicos, este modelo no es siempre aplicable (aumentando muchas veces significativamente el riesgo del proyecto de desarrollo). Y algunos de estos proyectos también tienen éxito.

Esto es tan sólo un ejemplo de que los **factores externos al contexto tecnológico** pueden llevarnos a tomar decisiones que en condiciones ideales evitaríamos, tanto en lo metodológico como en lo tecnológico. Estos factores externos, entonces, no son algo para menospreciar, porque en la vida real impactarán no sólo en nuestro proceso de desarrollo, sino incluso en nuestro diseño.