IIC2113 - Diseño Detallado de Software

CLASE 0 - PRESENTACIÓN

Pontificia Universidad Católica de Chile
2020-2

Índice

01. Motivación

¿De qué trata este curso?

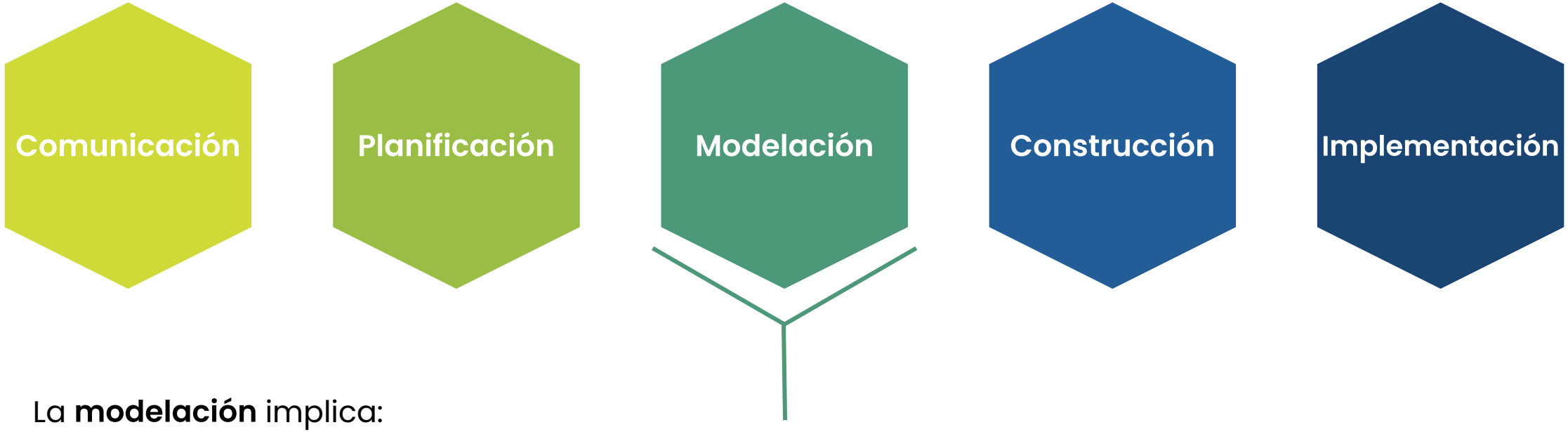
Desarrollo de software

Los procesos de Desarrollo de Software contemplan una serie de actividades comunes. Dependiendo de la metodología el orden de cada etapa podría variar y repetirse.

El **ciclo de vida de un proyecto de Software** varía dependiendo de la metodología (como Ágil, Cascada, Espiral, entre otras). Pero todas mantienen en común algunas actividades como: comunicación, planificación, modelación, construcción e implementación.

Desarrollo de software

Los procesos de Desarrollo de Software contemplan una serie de actividades comunes. Dependiendo de la metodología el orden de cada etapa podría variar y repetirse.



La **modelación** implica:

- Análisis del problema → entender el problema no necesariamente desde una vereda "programática". Es importante comprender el **contexto** y el **negocio**.
- **Diseño de la solución** → determinar y guiar a cómo resolver el problema. Se deciden los **componentes**, **la forma de los datos** y la **arquitectura del sistema**.

¿Cómo es posible **tomar decisiones** sobre un proyecto sin involucrarse de forma integral en el proceso?

¿Cómo es posible **tomar decisiones** sobre un proyecto sin involucrarse de forma integral en el proceso?

↑

Esta es la motivación tras este curso

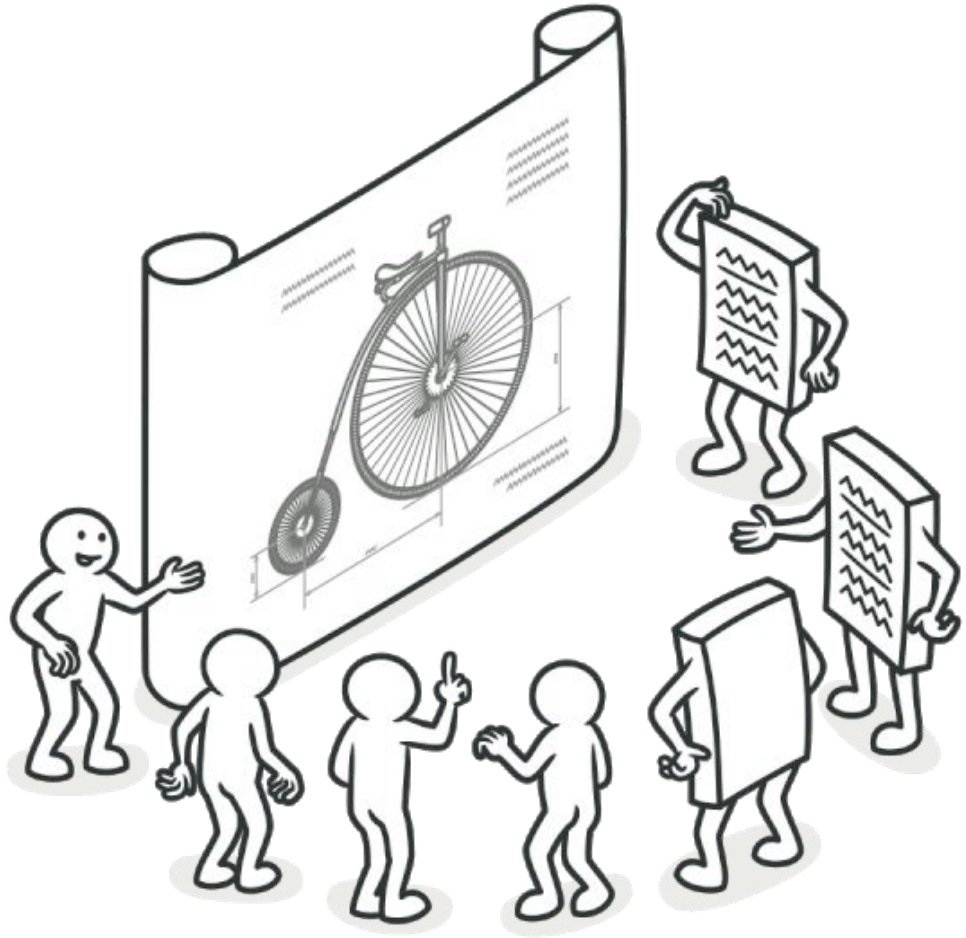
¿De qué trata el curso?

Entregarles las herramientas de forma integral para poder diseñar de forma detallada software que resuelva un problema y que perdure en el tiempo.

1

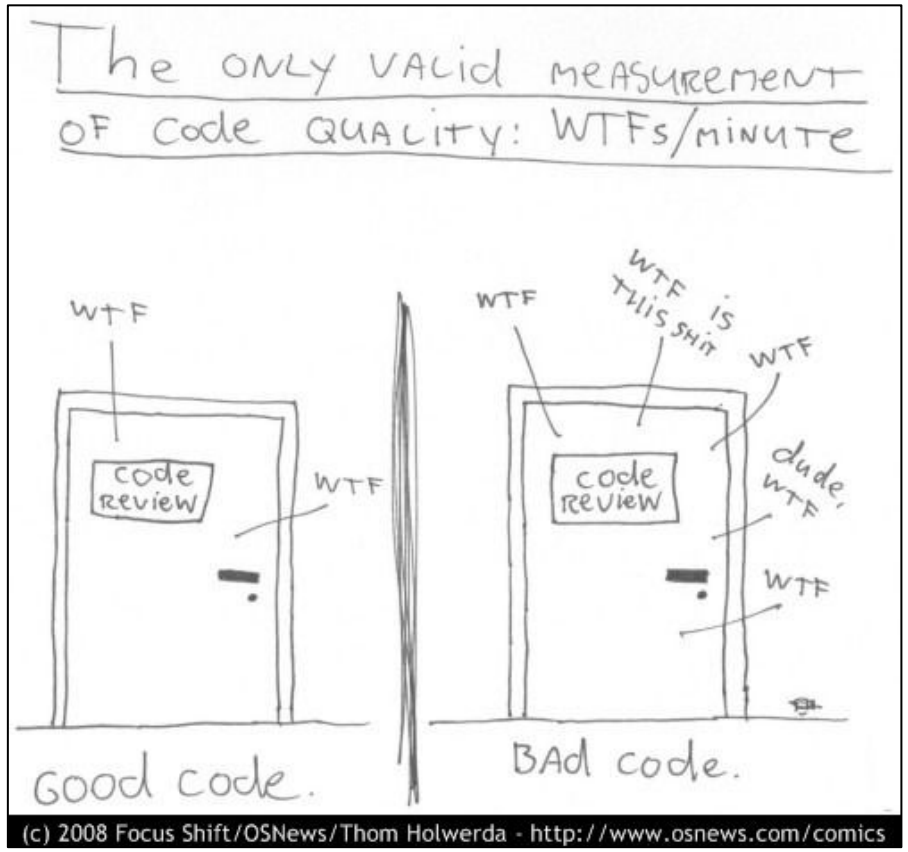
Existen problemas no tan complejos que pueden prescindir del diseño. Sin embargo, los problemas en la vida real suelen ser complejos.

Cuando tenemos que trabajar en un área compleja tenemos que comunicar decisiones y orientar el diseño a las cualidades de la solución (extensibilidad, *performance*, requisitos no funcionales, etc.-)



1: Ilustración de Dmitry Zhart, sitio web de https://refactoring.guru/.

2



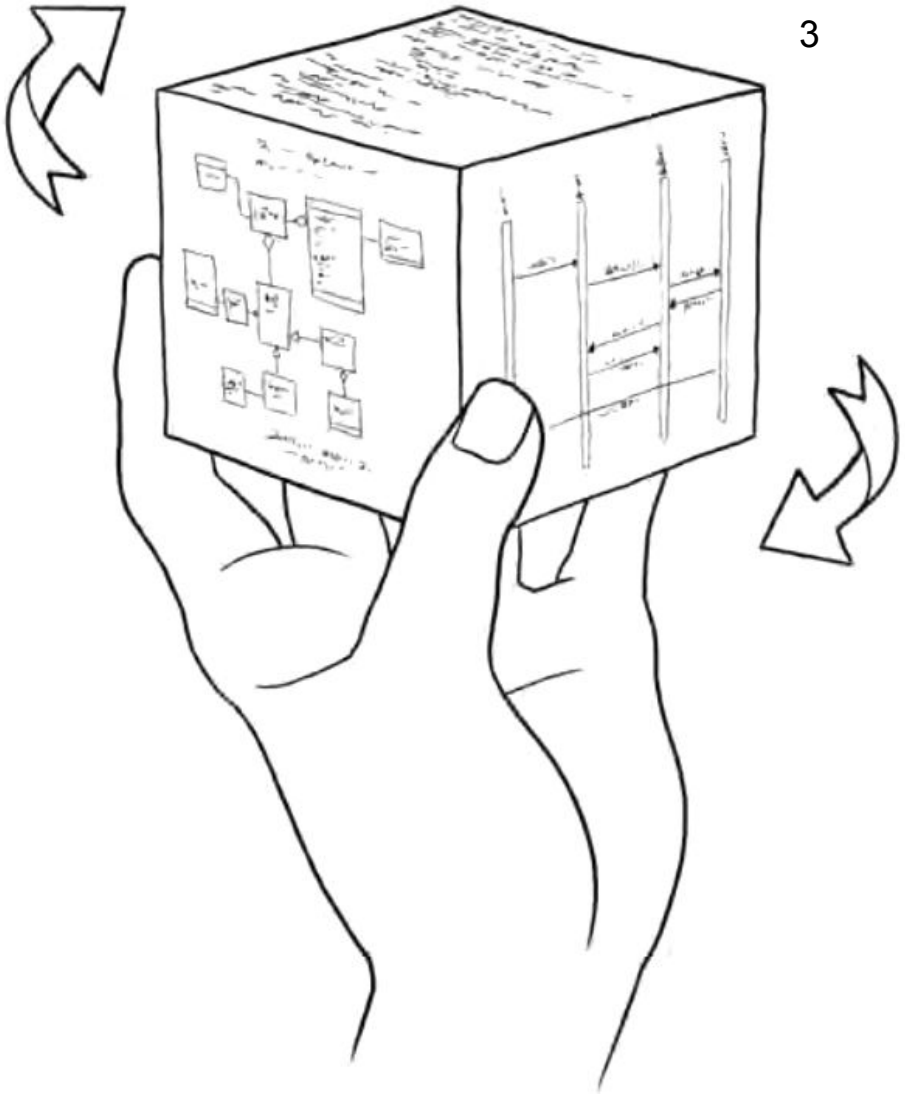
Estudiaremos cómo lograr código entendible y cuyo diseño permita extensibilidad de la solución.

Entenderemos las diferencias entre un "**buen código**" y un "**mal código**".

2: Ilustración de Thom Holwerda, sitio web de https://www.osnews.com/comics/.

Diseñar en distintas notaciones permite un mejor entendimiento para las distintas áreas involucradas.

Los distintos diagramas son **herramientas** que facilitan el trabajo de **comunicar** el problema y la solución.



3

3: Ilustración de Yen Quach. Libro *Software Design Decoded: 66 Ways Experts Think.*

02. Aspectos Administrativos

Aspectos Administrativos - Información General

Horario de clases	Viernes módulos 4 y 5
Horario de ayudantía	Viernes módulo 6
Profesora	M. Fernanda Sepúlveda (mfsepulveda@uc.cl)
Ayudantes	Por definir
Requisitos	IIC2143 (Ingeniería de Software)

Aspectos Administrativos - Objetivos

- Aplicar técnicas y herramientas de construcción de software, incluyendo enfoques basados en estados y dirigidos por tablas para diseño de bajo nivel de software.
- Usar patrones de diseño en la modelación de software.
- Realizar diseño y programación orientados a objetos con pericia.
- Analizar software para mejorar su eficiencia, confiabilidad, y mantenibilidad.
- Modificar diseños usando enfoques rigurosos de control de cambios.
- Usar técnicas de ingeniería inversa para recuperar el diseño de un producto de software.

Aspectos Administrativos - Contenidos

01 **Motivación**

02 **Diagramas UML**

03 **Buenas prácticas de desarrollo**

04 **Domain Driven Design**

05 **Patrones y arquitectura CLEAN**

06 **Ingeniería Inversa**

07 **Paradigmas de programación**

Aspectos Administrativos - Canales de comunicación

- Clases vía Zoom. Canal privado.
    - ID: Por definir.
    - Password: Por definir.
- Entrega de evaluaciones teóricas vía Canvas UC.
- Entrega de evaluaciones prácticas vía Github (en repositorio asignado al grupo, se considerará el último commit antes de la fecha límite entrega).
- Foro de conversaciones, preguntas y discusiones a través del Repositorio de Recursos en Github.
- Problemas u otros temas escribir a mi correo: mfsepulveda@uc.cl (para dudas puntuales prácticas o teóricas, preferir las Issues del Repositorio de Recursos).

03. Evaluaciones

Evaluaciones - Evaluación teórica

- Habrán dos interrogaciones y un exámen (eximible). Cada interrogación consta de dos partes:
  - Una parte escrita que se resolverá durante medio módulo.
  - Un ejercicio práctico que se podrá resolver de forma asíncrona y podrá entregarse durante las 24 horas siguientes. Podrá ser resuelto en Ruby o en C#. Al menos una de las evaluaciones que haga el alumno debe ser en C#.
- El exámen por su parte será escrito en su totalidad.

[I1] Interrogación 1	Viernes 11 de Septiembre, módulo 5.
[I2] Interrogración 2	Viernes 13 de Noviembre, módulo 5.
[EX] Exámen	Martes 15 de Diciembre, 15:30 hrs.

Evaluaciones - Evaluación práctica

- Existirá un proyecto con 3 entregables. El objetivo del proyecto es aplicar el contenido estudiado en el curso y **fomentar el trabajo colaborativo entre los alumnos**.
- En grupos de 5 alumnos.
- El lenguaje y framework del proyecto será **Ruby on Rails**. Se exigirá el uso de Gitflow y Code Review en el equipo.
- Los entregables serán:
  - [E1] Entrega 1: Diseño general de la solución.
  - [E2] Entrega 2: Implementación de la solución y actualización del diseño.
  - [E3] Entrega 3: Implementación de nuevas *features* sobre un Software Legacy y presentación de lecciones aprendidas.
  - El Software Legacy que recibirán en la entrega 3 será el proyecto de otro grupo asignado de forma aleatoria. El criterio de evaluación de la E3 será relativo a lo que recibieron en la E2.
  - Más detalles en el enunciado del proyecto.

Evaluaciones - Criterio de Aprobación

- La cátedra y el proyecto se aprueban por separado.

La nota de cátedra (NC) se calcula como sigue:

$$NP = \frac{I1 + I2}{2}$$

$$NEX = \begin{cases} NP \; ssi \; NP \geq 5{,}4 \\ \max\{EX, NP\} \; ssi \; NP < 5{,}4 \end{cases}$$

$$NC = \frac{I1 + I2 + 2 * NEX}{4}$$

NP: Nota de presentación.
NEX: Nota considerada del exámen.
NC: Nota de cátedra.
NE: Nota entregas proyecto.
NF: Nota final.

La nota del proyecto (NE) se calcula como sigue:

$$NE = 0{,}2 * E1 + 0{,}4 * E2 + 0{,}4 * E3$$

La nota final (NF) y criterio de aprobación es $NF \geq 3{,}95$

$$NF = \begin{cases} \frac{NC + NE}{2} \; ssi \; NC \geq 3{,}95 \; y \; NE \geq 3{,}95 \\ \min\{NC, NE\} \; ssi \; NC < 3{,}94 \; \text{ó} \; NE < 3{,}94 \end{cases}$$

04. Próxima Clase

Próxima clase

- En 30 minutos más → Clase 1 - Motivación.
- Durante el receso, rellenar el formulario inicial quienes falten (no tomará más de 5 minutos).

Clase 0 - Presentación

Bibliografía

- R. Pressman, Software Engineering, A Practitioner's Approach.
- R. Martin, Agile Principles, Patterns, and Practices in C#.
- R. Martin, Clean Code: A Handbook of Agile Software Craftsmanship.
- R. Martin, Clean Architecture: A Craftsman's Guide to Software Structure and Design.
- M. Fowler, UML Distilled: A Brief Guide to the Standard Object Modeling Language.
- M. Fowler, Refactoring: Improving the Design of Existing Code.
- M. Fowler, Patterns of Enterprise Application Architecture.
- C. Larman, Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development.
- S. McConnell, Code Complete: A Practical Handbook of Software Construction, Second Edition.
- E. Gamma, Design Patterns: Elements of Reusable Object-Oriented Software.
- E. Evans, Domain-Driven Design: Tackling 3Complexity in the Heart of Software.

IIC2113 - Diseño Detallado de Software

Fernanda Sepúlveda - mfsepulveda@uc.cl

Pontificia Universidad Católica de Chile
2020-2