

Orientacion a Objetos 2

Dra Alejandra Garrido

Dr. Alejandro Fernandez

Dr. Federico Balaguer

Dr. Gustavo Rossi



[garrido, alejandro.fernandez,federico.balaguer, gustavo]@lafia.info.unlp.edu.ar



Laboratorio de Investigación y Formación en Informática Avanzada
Facultad de Informática, Universidad Nacional de La Plata
Calle 50 esq. 115 - Primer piso (1900) La Plata, Argentina
TE: (0221) 422 8252 <http://www-lafia.info.unlp.edu.ar>



Contexto: Que software desarrollamos hoy

- Gran cantidad de empresas cuyo mayor (único?) activo es el software: AirBnB, Uber, Booking, Twitter, Google
- Aplicaciones Distribuidas que combinan Software, Comunicaciones, Sensores, Hardware, Personas, todas ellas en dominios novedosos
- Cantidad creciente de funcionalidad “importada” de terceros, desde servicios específicos de bajo nivel (Cloud), o alto nivel (funcionalidad multimedia, redes sociales, etc), pasando por servicios de todo tipo
- “Destilación” permanente de datos mediante análisis de grandes volúmenes de información para cambiar el comportamiento general de las aplicaciones

Que tipo de aplicaciones

- Aplicaciones que combinan “partes” de otras aplicaciones (por ejemplo via servicios). El reuso pasó a ser imprescindible. Se diseña para reusar.
- Con tipos de interacciones e interfases cada vez mas sofisticadas
- Que pueden crecer en forma completamente ininmaginable..
- Que se componen y descomponen y cuyas partes son usadas por otros....
- Con Requerimientos no-funcionales “nuevos”: Escalabilidad, accesibilidad, usabilidad, disponibilidad....
- “Nuevos” dominios y problemas: Aplicaciones inter-vehiculares, Integración de Big Data e IA en software “convencional”, Internet de las cosas, etc

Que nuevos problemas nos generan?

- “Divide and Conquer” complejizado a niveles nunca imaginados
- Los módulos (componentes, clases, servicios, etc) de nuestra aplicación ya no co-existen (incluso ya no lo hacen “amigablemente”) en la misma plataforma
- Los módulos pueden estar desarrollados en diferentes lenguajes, comunicarse vía diferentes mecanismos, obviamente estar desarrollados por equipos diferentes, quizás sin relación entre ellos.
- El problema de búsqueda de modularidad para asegurar mejor mantenimiento obviamente se complica

Como los resolvemos?

- Teniendo buenas estrategias de separación de concerns para atacar problemas diferentes en módulos diferentes
- Aprendiendo los conceptos que nos permiten hacer “divide and conquer” (diseño) en diferentes niveles
- Aprendiendo mecanismos de interoperabilidad y “sharing” de información entre módulos y aplicaciones
- Conociendo los requerimientos no-funcionales mas usuales
- Conociendo patrones, estilos y ejemplos de arquitecturas exitosas



Decisiones de diseño vs otras

- ✓ Que tipo de decisiones tenemos que tomar? Cuando las tomamos?
- ✓ Arquitectura vs. Diseño vs. Programación
- ✓ Donde quedan nuestros lenguajes?

Acerca del Diseño y su proceso

- Diseñar implica tomar decisiones para satisfacer objetivos, requerimientos y restricciones
- Un diseño por lo tanto refleja como se cumplirán dichos objetivos, requerimientos y restricciones

Es difícil diseñar?

- El diseño de algo “conocido” suele ser simple porque contamos con experiencia acumulada
- Afortunadamente una parte de los problemas que debemos enfrentar no son novedosos
- Para enfrentar estos casos contamos con ejemplos de éxito, fragmentos de dichos casos, “bloques” de construcción (los conceptos de diseño), a veces representados por “patrones” o “estilos”
- Y básicamente combinamos, reusamos o adaptamos estos casos exitosos o bloques de construcción

Niveles de Diseño

- Igual que en el Diseño urbanístico o de casas, en software también tenemos diferentes niveles y problemas de diseño, de diferente granularidad
- Diseño de la colaboración entre objetos, diseño de estructuras de información, diseño de la relación y comunicación entre componentes, etc.
- Durante el curso atacaremos los problemas que involucran micro-arquitecturas (de objetos por ejemplo)

Arquitectura “urbana”

- ✓ Conocemos muchos “estilos”. Y aún los que no somos arquitectos entendemos a que se refiere cada uno
 - ✓ Casa chorizo
 - ✓ Duplex
 - ✓ Chalet
 - ✓ Propiedad Horizontal
 - ✓ Edificio
 - ✓ Country
- ✓ Veremos que en software no es muy diferente

Arquitectura Urbana

- ✓ Sabemos incluso:
 - ✓ Ciertas reglas para “usar” cada uno
 - ✓ Que ciertos estilos arquitectónicos no se combinan (Country con edificios o PHs)
 - ✓ Otros si se combinan (PHs y Dúplex)

Arquitectura?

- ✓ Tenemos que construir un edificio:
 - ✓ Cuan relevante en el proceso es la marca de los ladrillos? O donde compramos el cemento?
 - ✓ No podemos pasar los caños de gas sin tener la “estructura”
 - ✓ Sin albañiles no hay edificio, pero quien lo “diseña”? Que habilidades requiere?
 - ✓ Las “primeras” decisiones son críticas (aunque como en software, las últimas también pueden hacerlo inviable)

Arquitectura y Arquitectura de Software

- ✓ **IEEE Definition:** Architecture is the highest-level concept of a system in its environment.
- ✓ The architecture of a software system (at a given point in time) is its organization or structure of significant components interacting through interfaces, those components being composed of successively smaller components and interfaces

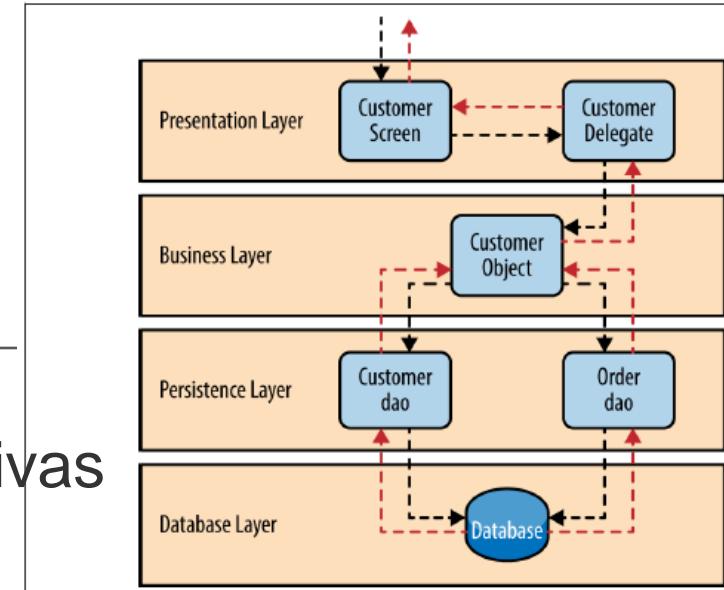
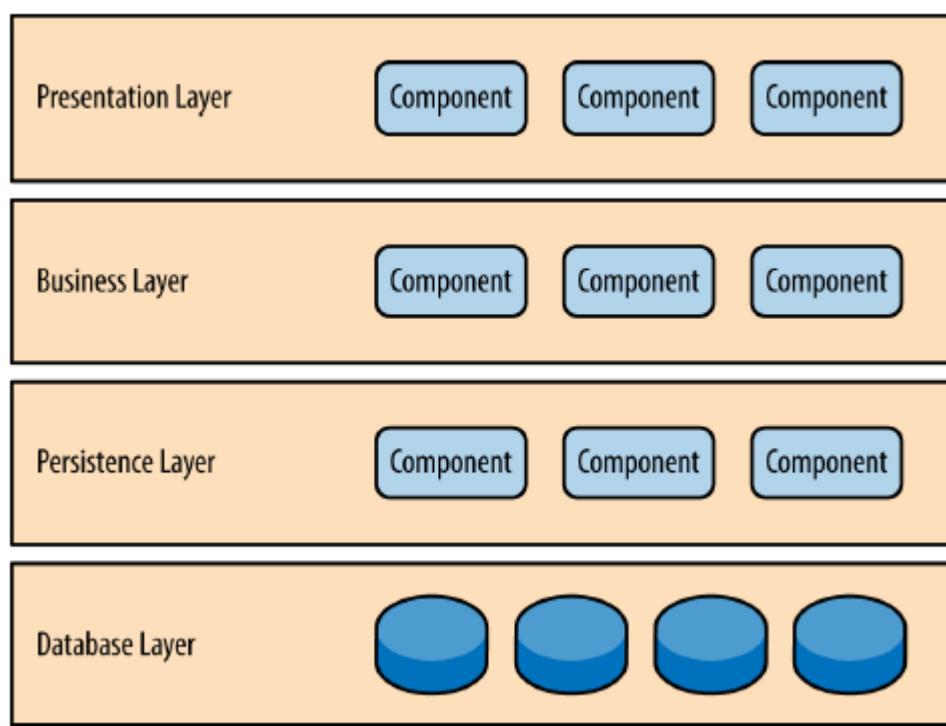
Arquitectura

- ✓ Ford (2020): Software architecture consists of the structure of the system, combined with the architecture characteristics (“-ilities”) that the system must support, architecture decisions, and finally design principles.
 - ✓ The structure of the system refers to the type of architecture style or styles (such as microservices, layered, etc).
 - ✓ The architecture characteristics define the success criteria of the system (availability, security, scalability.....).
 - ✓ Architecture decisions define the rules for how a system should be constructed. For example, an architect might make an architecture decision that only the business and services layers within a layered architecture can access the database restricting the presentation layer from making direct database calls. Architecture decisions form the constraints of the system and direct the development teams on what is and what isn't allowed.

Aspectos a tener en cuenta

- ✓ Distribución del sistema en partes (componentes)
- ✓ Alocación de esas componentes a dispositivos físicos (servidores, clientes, etc)
- ✓ Tipo de comunicación entre las componentes
- ✓ Reglas de comunicación (cualquiera puede comunicarse con cualquiera?)
- ✓ OO2 se concentra en un aspecto “intermedio”, el diseño

Arquitectura en Capas



Típico en aplicaciones Web e interactivas

Otras...

- ✓ Arquitectura orientada a Servicios
- ✓ Arquitectura de microservicios
- ✓ Microkernel
- ✓



“Netflix” gratuito

- ✓ Tenemos que desarrollar un software tipo Netflix pero libre y gratuito
- ✓ Se nutre de videos de youtube, vimeo y otros
- ✓ Tiene que poder competir con Netflix
- ✓ Los usuarios pueden subir pelis como si fuera youtube y comentar como en twitter

Problemas

- ✓ Empezamos de cero?
- ✓ Como organizamos la “lógica”, la persistencia, la interfaz, la seguridad, la privacidad
- ✓ Como programamos el reproductor?
- ✓ Como recomendamos? Las formas de recomendar dependen de que?
- ✓ Como reproducimos?
- ✓ “Almacenamos” los videos? Almacenamos algo? Donde?
- ✓ Queremos reproducir en smart tv...como? Otro soft?
- ✓ Como evolucionamos (e.g. Netflix que empezo mandando DVDs por correo como un videoclub!!!)

Las aplicaciones son interactivas

- ✓ En la Web en un smartphone, tableta etc, lo esencial es la interaccion
- ✓ Como construimos interfaces interactivas? Siguiendo que principios?
- ✓ Como hacemos para que la funcionalidad no dependa de la caracteristica del dispositivo? (tablet, Web, etc)

Como desarrollamos?

- ✓ Cuanto tiempo nos puede llevar diseñar un sistema ultra-flexible?
- ✓ Es aceptable para los clientes?
- ✓ Hay alternativas?

Metodos Agiles

Somos Agiles...y entonces?

- ✓ Los métodos agiles suelen buscar resultados “rápidamente”, para chequear requerimientos con los clientes
- ✓ Pero los diseños entonces quedan “feos”
- ✓ Aparecen “malos olores” que dificultan la evolución
- ✓ Debemos eliminarlos sin romper el sistema

Refactoring

Que pasa con la “correctitud” del sistema

- ✓ Somos agiles y ademas...
- ✓ Nuestros diseños son modulares (soportan el cambio)
- ✓ Pero como sabemos que el sistema funciona? Esperamos hasta...cuando?

Test Driven Development (TDD)



Mas allá de la Agilidad...

- ✓ Tenemos que diseñar correctamente!!!
- ✓ Precisamos diseños modulares
- ✓ Extensibles
- ✓ Comprensibles

Problemas en el diseño con objetos

- ✓ Encontrar las clases adecuadas
- ✓ Identificar correctamente las responsabilidades (métodos)
- ✓ Jerarquías de clases adecuadas
- ✓ Diseñar para el cambio (incluso para el no previsto)
- ✓ Reusar código o diseño existente

.....

.....

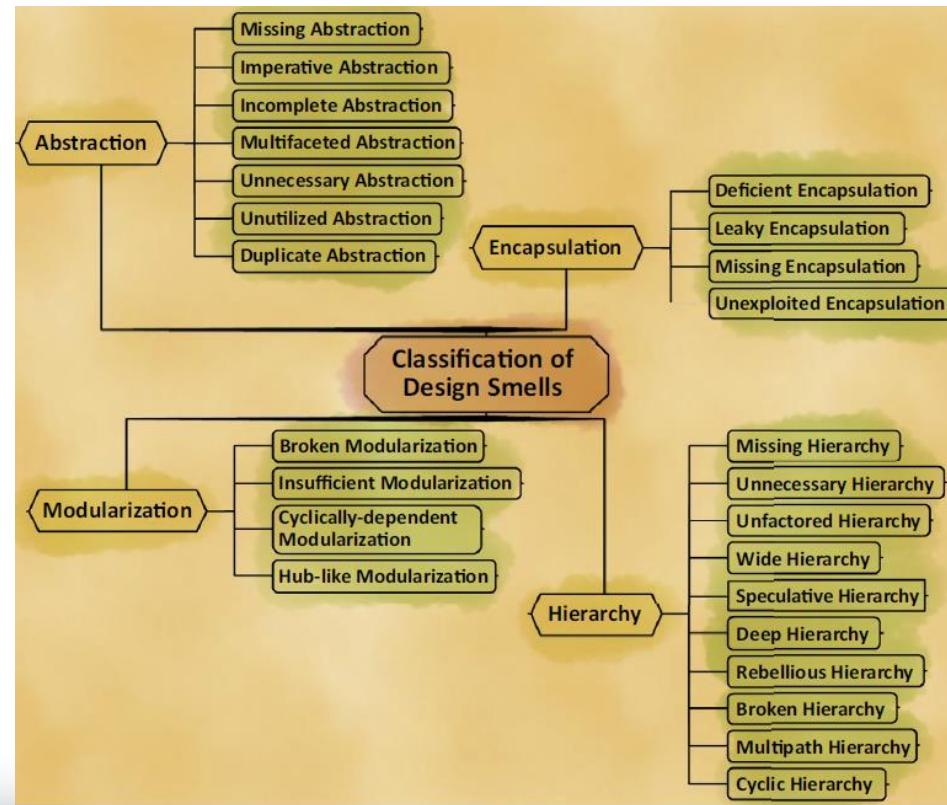
Hay criterios para saber si un diseño es “correcto”?

✓ Diseñar siguiendo “principios” de diseño como estos....

Design Principle	Description
Abstraction	The principle of abstraction advocates the simplification of entities through reduction and generalization: reduction is by elimination of unnecessary details and generalization is by identification and specification of common and important characteristics [48].
Encapsulation	The principle of encapsulation advocates separation of concerns and information hiding [41] through techniques such as hiding implementation details of abstractions and hiding variations.
Modularization	The principle of modularization advocates the creation of cohesive and loosely coupled abstractions through techniques such as localization and decomposition.
Hierarchy	The principle of hierarchy advocates the creation of a hierarchical organization of abstractions using techniques such as classification, generalization, substitutability, and ordering.

O....

- ✓ Diseñar como se pueda, chequear “malos olores” y refactorizar



Aprovechar la experiencia...

- ✓ Hacer diseños mas sensibles a la evolucion
- ✓ Como me doy cuenta que necesito en este caso particular?
- ✓ Con mas experiencia...mia, o de otros

Patrones de Diseño

Como estructuramos sistemas “grandes”?

- ✓ Los sistemas rara vez son “pequeños” programas ejecutandose en un equipo
- ✓ Habitualmente la funcionalidad esta distribuida (como en una app Web o mobile)
- ✓ Como organizamos las componentes de tales sistemas?
- ✓ Como “consumimos” informacion o funcionalidad de otros sistemas?

Patrones Arquitecturales

Interfaces, Persistencia...

- ✓ Queremos que nuestro sistema muestre informacion en distintas formas (barras, estadisticas, mapas....)
- ✓ Y queremos guardar los datos en diferentes formatos....
- ✓ Pero todo esto no tiene que ver con nuestro problema....

Frameworks

Interaccion, Interfaz

- ✓ Nuestros sistemas son usables?
- ✓ Un usuario “promedio” consigue llevar a cabo las tareas que el sistema soporta?
(e.g. una transferencia en home banking,
una compra en un e-commerce, un
remate en e-bay)



Inicio — Italia 155.256 alojamientos — El Lacio 13.690 alojamientos

— Lago de Bracciano 172 alojamientos — Hoteles en Manziana 7 alojamientos

Ver fechas disponibles

Guardar en una lista

Guardado en 129 listas

Igualamos el precio

Buscar

Destino/Nombre del hotel:

Manziana

viernes, 2 de junio de 2017

lunes, 5 de junio de 2017

Estancia de 3 noches

¿Viajas por trabajo?

Habitaciones 1

Adultos 2

Niños 0

.genius Mostrar descuentos Genius primero

Buscar



Ver en el mapa

Google

Map data ©2017 Google

¿Te gusta este, pero no te acabas de decidir?

Mostrar alojamientos similares

31 clientes han tenido muy buenas experiencias

9,6 "The view from the property was beautiful. The decor was quirky and individual and the staff were very welcoming."

Julie Reino Unido Comentario auténtico

10 "A lovely hotel, in a quiet village. Easy to get to from Rome and perfect for a few days getaway."

Jonathan

Info habitaciones y suites Servicios > Léeme A tener en cuenta Experiencias de nuestros clientes (55) >

Villa Clodia Relais ★★★

Via del Mattiolo 3, 00066 Manziana, Italia —

Ubicación ideal. Mostrar mapa

Ver todas las fotos Comida y bebida Habitaciones

Muy bien 8 /10
Puntuación basada en 55 comentarios

Ambiente familiar e disponibilidad personal e proprietaria

Alessandro, Italia

Personal

8,4

El Relais Villa Clodia está situado en Manziana, en la campiña romana, a solo 6 km del lago de Bracciano. Ofrece habitaciones elegantes y un jardín grande con piscina al aire libre.

Todas las habitaciones del establecimiento Villa Clodia están decoradas de forma individual y cuentan con zona de estar y baño privado. Hay conexión Wi-Fi gratuita en la recepción y en el salón común.

El establecimiento sirve desayunos en su interior durante el invierno y en la terraza del jardín, en verano.

Además, el establecimiento Villa se halla a solo 200 metros de una parada de autobuses que conducen a Roma y Viterbo y a 400 metros de la estación Manziana-Canale Monterano, desde la cual también se puede llegar a dichas ciudades. El bosque de Macchia Grande está a 1 km.

El Villa Clodia Relais recibe clientes de Booking.com desde el 23 dic 2011.
Habitaciones del hotel : 8

A los clientes les gustó por...

Consigue lo que necesitas

Este alojamiento responde rápido a cualquier pregunta que tengas después de reservar.

¡ideal para estancias de 3 noches!

La mejor ubicación. Los viajeros recientes le dan una puntuación alta (8,3)

Información sobre el desayuno

Buffet

P Hay parking privado gratis en el alojamiento

Reserva ahora

Guardar en una lista

Guardado en 129 listas

No tenemos disponibilidad en nuestra página web para este alojamiento entre el vie, 2 jun 2017 y el lun, 5 jun 2017

Lo sentimos, el último espacio disponible en el Villa Clodia Relais en nuestra página web ha sido reservado hace 6 días, 44 minutos

Tipo de habitación	Máx.	Precio
Habitación Cuádruple Económica - Anexo	4	No disponible en nuestra web para tus fechas
Habitación Doble Económica - Dependance	2	No disponible en nuestra web para tus fechas
Habitación Doble Deluxe	2	No disponible en nuestra web para tus fechas
Habitación Triple	3	No disponible en nuestra web para tus fechas
Suite de 2 dormitorios	4	No disponible en nuestra web para tus fechas
Habitación Doble	2	No disponible en nuestra web para tus fechas
Habitación Cuádruple	4	No disponible en nuestra web para tus fechas
Suite Junior	2	No disponible en nuestra web para tus fechas

Interfaz, interaccion...

- ✓ Que guias tenemos para organizar la interfaz, la interaccion?
- ✓ Como nos aseguramos que el “diseño” de la interfaz sea usable?

Patrones de Interfaz/interaccion



La importancia de la Experiencia de Diseño

- ✓ Diseñar Software es difícil (aun usando buenas tecnicas)
- ✓ La experiencia es insustituible
- ✓ Como la capturamos, transmitimos y usamos?

Ejemplo



Soluciones?



✓ Rompemos la pared?

✓ Rompemos el enchufe?

✓ Buscamos como adaptarlo?

Discusión

- ✓ Entender que el problema no lo tiene ni la pared ni la heladera
- ✓ Es un problema de interfaces incompatibles, que además no se agota en mi heladera y mi pared o en problemas de corriente eléctrica
- ✓ El concepto de “adaptador” va más allá de los artefactos que compramos en la ferretería. Implica el reconocimiento de un problema y de la mejor solución

Design Patterns

- ✓ Originados en la Arquitectura (Alexander)
- ✓ Diferentes tipos de patrones: de diseño, programacion, arquitectura de software, testeo, interfaz grafica, Web

Patterns-Definiciones

✓ Según Alexander

“A pattern describes a problem that occurs once and again in a context, and describe the core of the solution in such a way that it can be used millions of times without doing the same twice”

Patterns-Definiciones

- ✓ Un patron es un par problema-solucion
- ✓ Los patrones tratan con problemas recurrentes y buenas soluciones a esos problemas
- ✓ Los patrones deben incluir una regla:
Cuando aplico el patron?

Patrones de Diseño

Definicion (GangOfFour):

A design pattern names, abstracts, and identifies the key aspects of a common design structure that make it useful for creating a reusable object-oriented design. The design pattern identifies the participating classes and instances, their roles and collaborations, and the distribution of responsibilities. Each design pattern focuses on a particular object-oriented design problem or issue. It describes when it applies, whether it can be applied in view of other design constraints, and the consequences and trade-offs of its use.

***Gamma, Helms, Johnson, Vlissides: Design Patterns.
Elements of Reusable Object-Oriented Software.***

Clasificacion de Patrones

✓ De acuerdo a la actividad

- ✓ Patrones de Soft
- ✓ Patrones de Testing
- ✓ Patrones de Proceso
- ✓ Patrones pedagogicos

✓

✓

Clasificacion de Patrones

- ✓ De acuerdo al nivel de abstraccion
 - ✓ Patrones de Analisis
 - ✓ Patrones de Arquitectura
 - ✓ Patrones de Diseño
 - ✓ Idioms

Clasificacion..

✓ De acuerdo al dominio de aplicacion:

- ✓ Financiero
- ✓ Comercio electronico
- ✓ Salud
- ✓ Tiempo Real
- ✓ ..
- ✓ ..

Clasificacion

✓ De acuerdo al ambiente/paradigma:

- ✓ Patrones Web
- ✓ Patrones de Modelos de datos
- ✓ XML
- ✓ Interaccion

Clasificación de Patrones de Diseño

✓ De acuerdo al propósito:

✓ **Creationales**

✓ **Estructurales**

✓ **De Comportamiento**

Descripcion de un patron

✓ Según GoF:

- ✓ *Name*
- ✓ *Intent*
- ✓ *Motivation (Problem)*
- ✓ *Aplicability*
- ✓ *Structure*
- ✓ *Participants*
- ✓ *Collaborations*



Descripcion....

- ✓ *Consequences*
- ✓ *Implementation*
- ✓ *Code*
- ✓ *Known Uses*
- ✓ *Related Patterns*