Estilos Arquitectónicos

Ing. Ariel Cassan

Agenda

#	Tema	Duración
1	Que es un Patrón?	5 min
2	Introducción a estilos arquitectónicos	5 min
2.1	De Estructuración	20 min
2.2	Sistemas distribuidos	5 min
2.3	De Comunicación	10 min
2.4	Sistemas interactivos	15 min
2.5	Sistemas adaptables	10 min
2.6	Otros	5 min
3	Bibliografía	5 min.

Agenda

#	Tema	Duración
1	Que es un Patrón?	5 min
2	Introducción a estilos arquitectónicos	5 min
2.1	De Estructuración	20 min
2.2	Sistemas distribuidos	5 min
2.3	De Comunicación	10 min
2.4	Sistemas interactivos	15 min
2.5	Sistemas adaptables	10 min
2.6	Otros	5 min
3	Bibliografía	5 min.

¿Qué es un patrón?

- Los patrones son soluciones reutilizables.
- Tipos de patrones
  - De sistema (o estilos arquitectónicos)
  - De diseño (por ejemplo GoF)
  - De codificación (buenas prácticas y estándares)
  - Anti-Patterns
- Una arquitectura **combina** muchos patrones o estilos

Agenda

#	Tema	Duración
1	Que es un Patrón?	5 min
2	Introducción a estilos arquitectónicos	5 min
2.1	De Estructuración	20 min
2.2	Sistemas distribuidos	5 min
2.3	De Comunicación	10 min
2.4	Sistemas interactivos	15 min
2.5	Sistemas adaptables	10 min
2.6	Otros	5 min
3	Bibliografía	5 min.

Estilos arquitectónicos

- Un patrón o estilo arquitectónico define:
  - Qué forma tienen los **componentes**
  - Qué forma tiene la **comunicación** entre esos componentes
  - Qué **restricciones** se ponen a esa comunicación

Estilos arquitectónicos

- **Estructuración**
- Dataflow Systems
  - Batch Secuential
  - Pipes & Filters
- Call & Return
  - Hierarchical Layers
- **Sistemas distribuidos**
  - Broker
- **Comunicación**
  - Forwarder-Receiver
  - Client-Dispatcher-Server
  - Publisher-Subscriber
- **Sistemas Interactivos**
  - MVC
- **Sistemas Adaptables**
  - Microkernel

Agenda

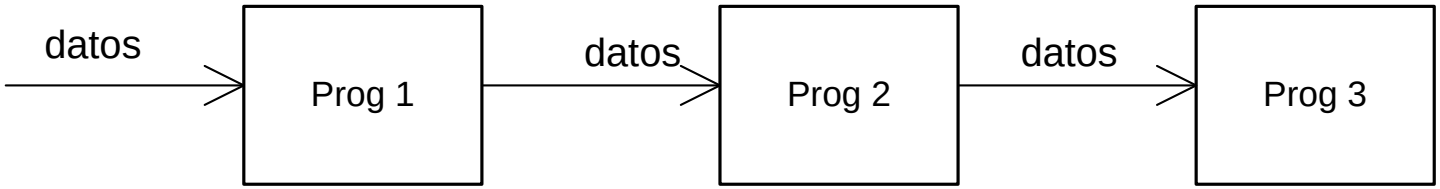
#	Tema	Duración
1	Que es un Patrón?	5 min
2	Introducción a estilos arquitectónicos	5 min
2.1	De Estructuración	20 min
2.2	Sistemas distribuidos	5 min
2.3	De Comunicación	10 min
2.4	Sistemas interactivos	15 min
2.5	Sistemas adaptables	10 min
2.6	Otros	5 min
3	Bibliografía	5 min.

Patrones de estructuración - Dataflow

- Serie de transformaciones de sucesivos datos de entrada.
- Los datos fluyen a través de los componentes del sistema.
- Estilos
  - Batch Secuential
  - Pipes & Filters

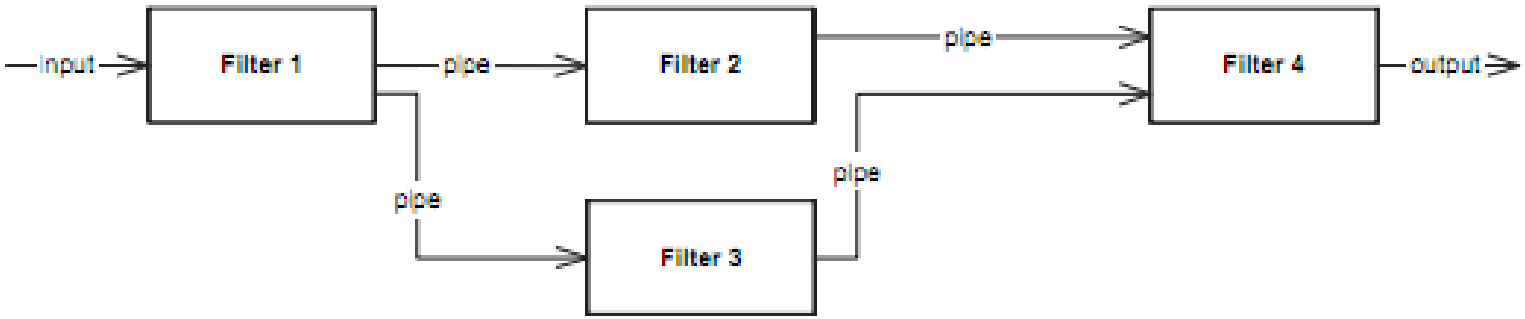
DataFlow - Batch Sequential

- Los pasos de procesamiento son programas independientes
- Cada paso se completa antes de comenzar el siguiente.
- Los datos se transmiten entre programas como un todo.



DataFlow - Pipe & Filters

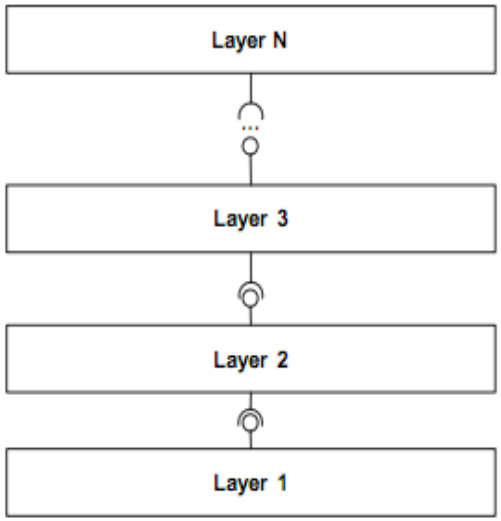
- Hincapié en la transformación incremental de los datos.
- Los filtros transforman los datos sin retener información de estado.
- Los pipes mueven los datos entre los filtros pero permiten la flexibilidad en las conexiones.



Patrones de estructuración – Call & Return

- Estilo arquitectónico dominante
- El sistema se ve como una serie de llamadas a procedimientos y funciones.
- Estilos
  - Hierarchical Layers
  - Main Program - Subroutines
  - Object Oriented

Call & Return – Hierachical Layers



- Cada capa:
  - Provee servicios a la capa superior
  - Es cliente de la capa inferior
- No todos los sistemas pueden ser arquitecturados en capas
- La mayor dificultad es encontrar los niveles de abstracción adecuados.

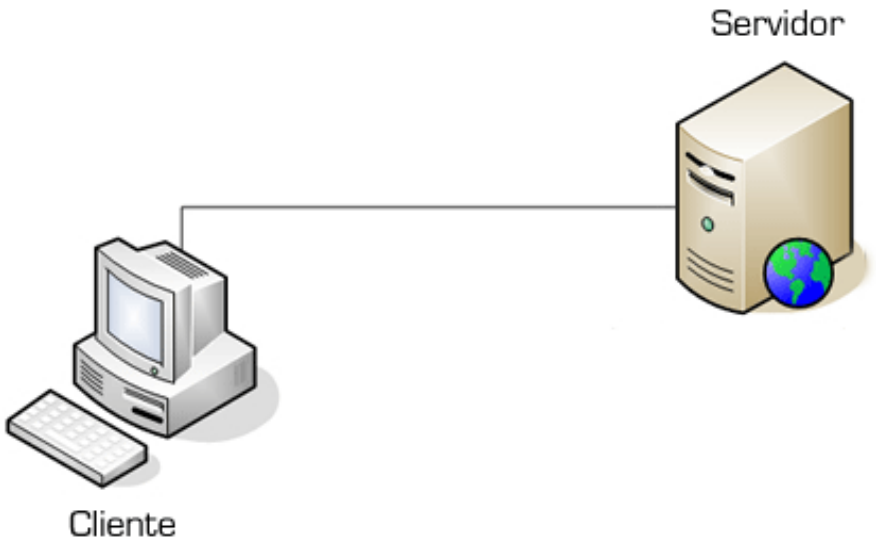
Call & Return – Hierachical Layers

- Ventajas:
  - Permite dividir un sistema complejo mediante abstracción.
  - Mejoras en una capa impactan en todo el sistema
  - Suponen reusabilidad y fácil intercambio
  - Facilitan el testing unitario de la funcionalidad de cada capa

- Desventajas:
  - Posible deterioro de performance con muchas capas.
  - Cambios suelen tener que replicarse en todas las capas

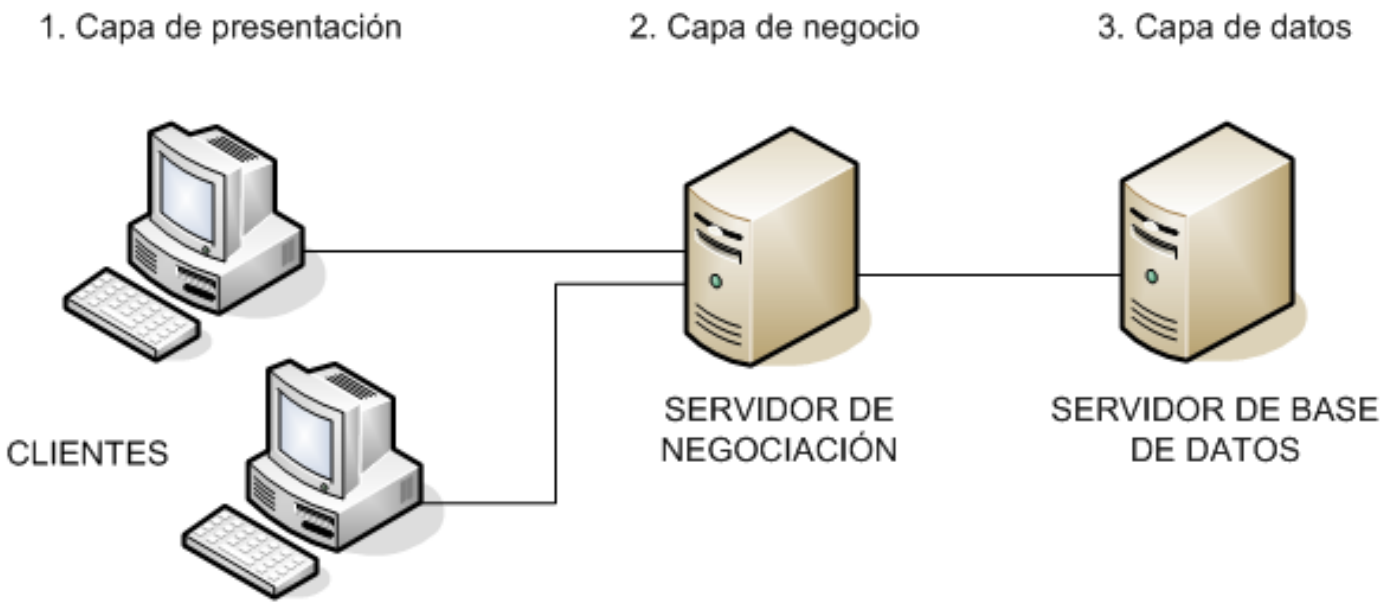
Call & Return – Hierachical Layers

- Arquitectura 2 capas: Cliente - Servidor



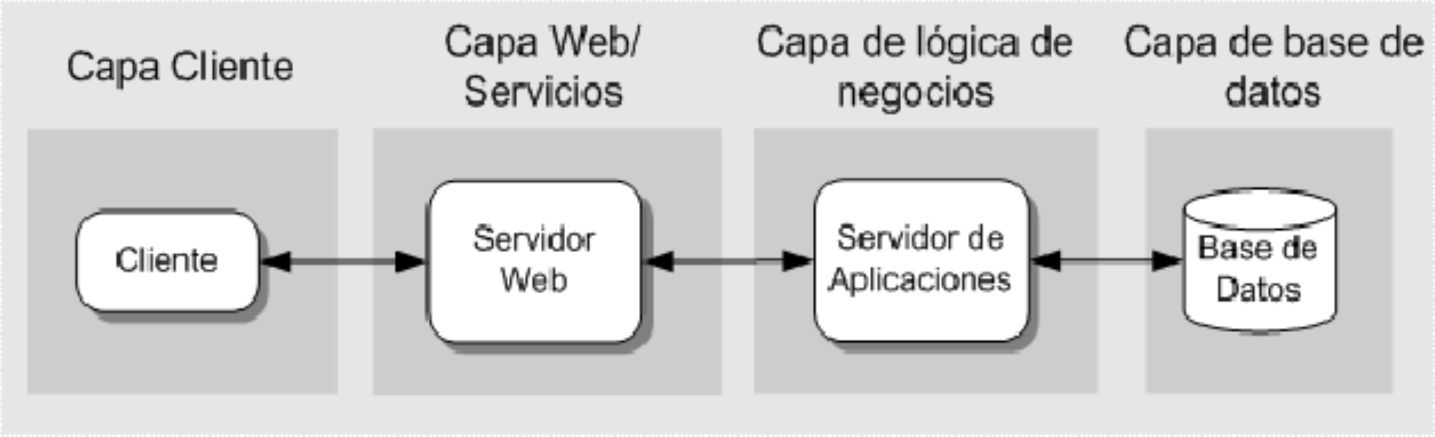
Call & Return – Hierachical Layers

- Arquitectura 3 capas



Call & Return – Hierachical Layers

- Arquitectura 4 capas



Agenda

#	Tema	Duración
1	Que es un Patrón?	5 min
2	Introducción a estilos arquitectónicos	5 min
2.1	De Estructuración	20 min
2.2	Sistemas distribuidos	5 min
2.3	De Comunicación	10 min
2.4	Sistemas interactivos	15 min
2.5	Sistemas adaptables	10 min
2.6	Otros	5 min
3	Bibliografía	5 min.

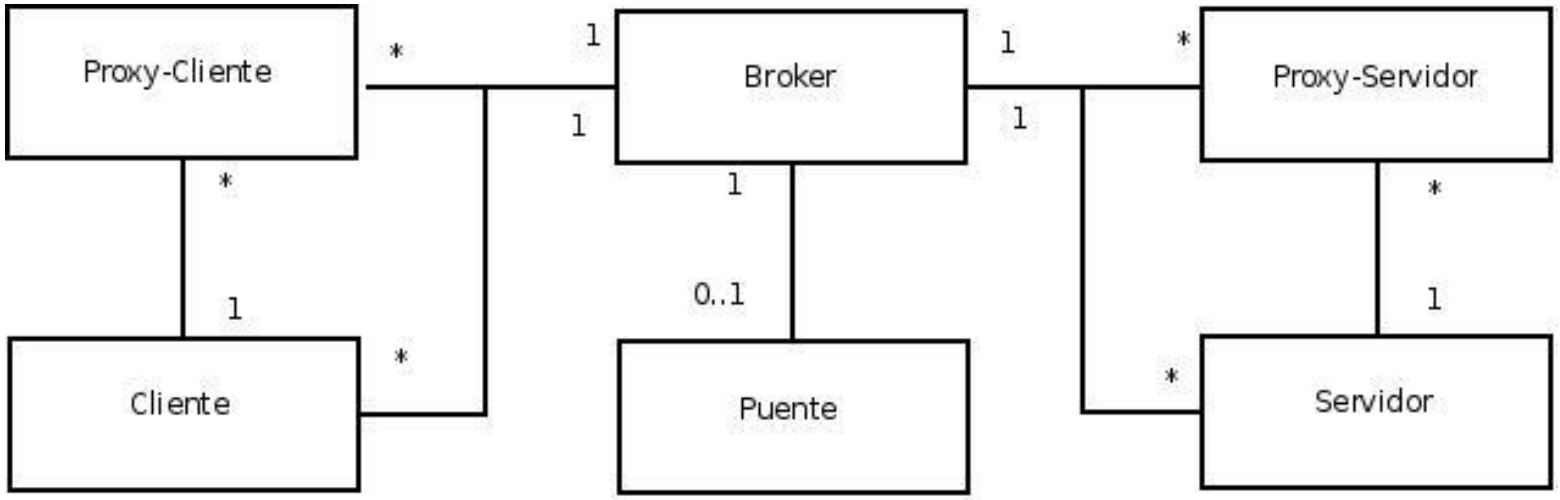
Patrones sobre sistemas distribuidos

Broker

- Un componente "Broker" coordina la comunicación entre clientes y servidores.
- Los clientes y servidores utilizan componentes "proxy" para comunicarse con el broker.
- Permite que la distribución de la aplicación sea transparente para los implementadores.
- Posibilita la interoperabilidad de tecnologías heterogéneas

Patrones sobre sistemas distribuidos

Broker



- Ejemplos de uso: CORBA, World Wide Web

Agenda

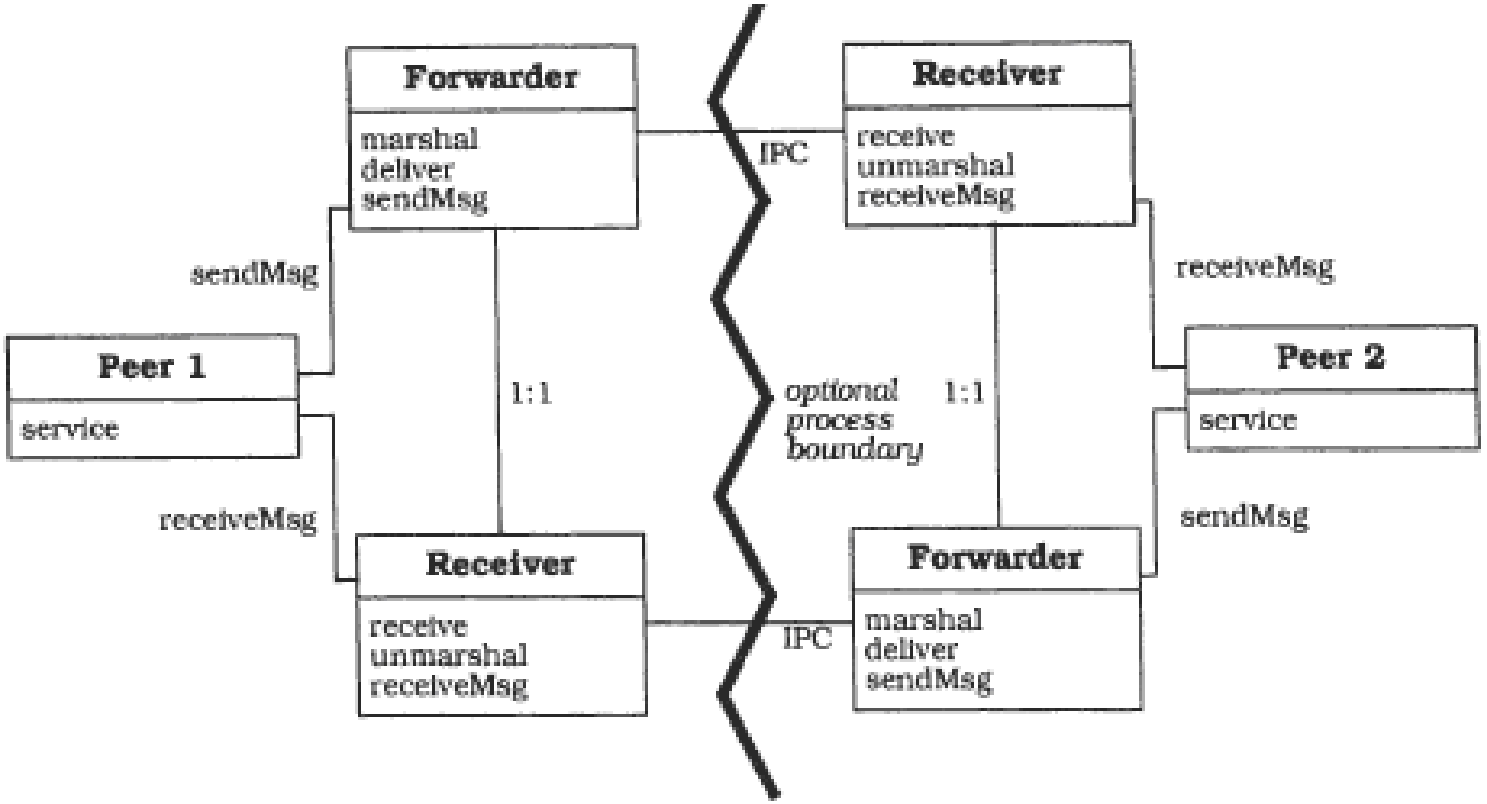
#	Tema	Duración
1	Que es un Patrón?	5 min
2	Introducción a estilos arquitectónicos	5 min
2.1	De Estructuración	20 min
2.2	Sistemas distribuidos	5 min
2.3	De Comunicación	10 min
2.4	Sistemas interactivos	15 min
2.5	Sistemas adaptables	10 min
2.6	Otros	5 min
3	Bibliografía	5 min.

Patrones de comunicación

- Muchos sistemas de software distribuyen sus componentes a través de redes, por lo que requieren comunicación.

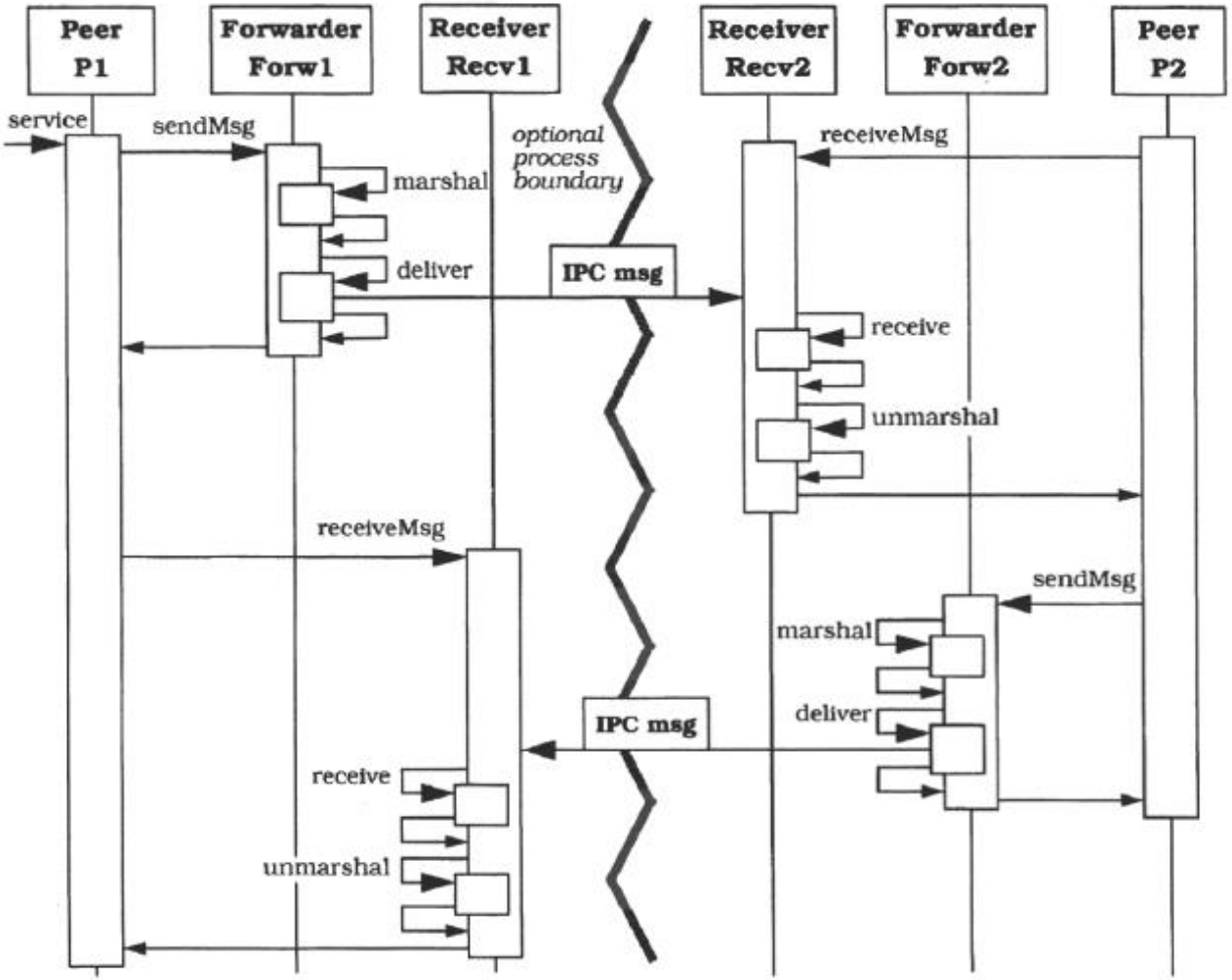
- Estilos:
  - Forwarder-Receiver
  - Client-Dispatcher-Server
  - Publisher-Subscriber

Forwarder-receiver



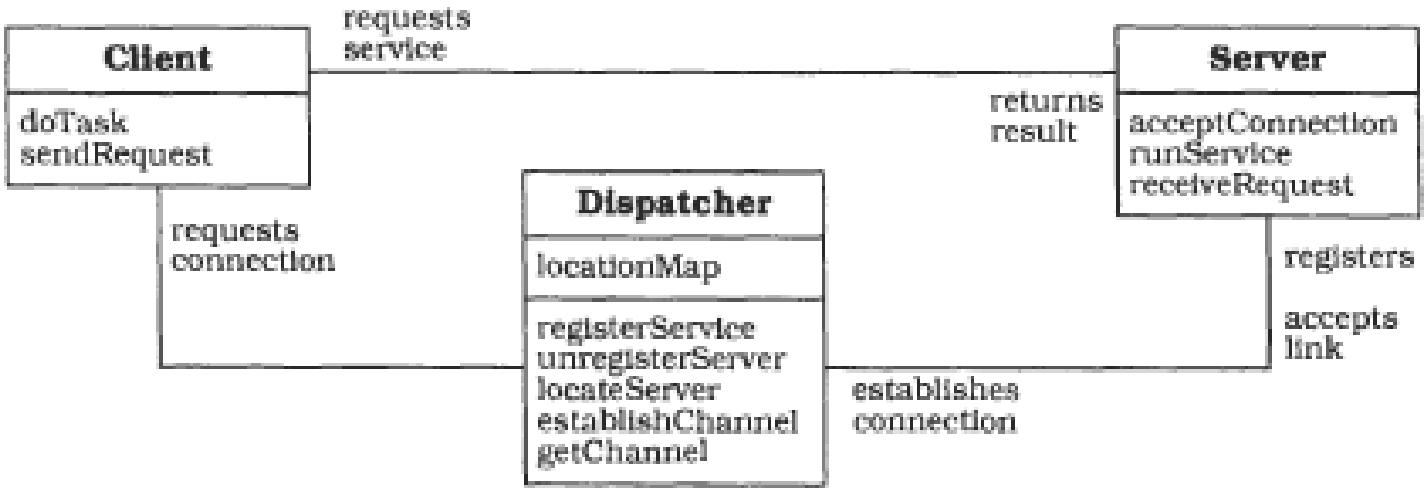
- Provee encapsulamiento del mecanismo de comunicación

Forwarder-receiver



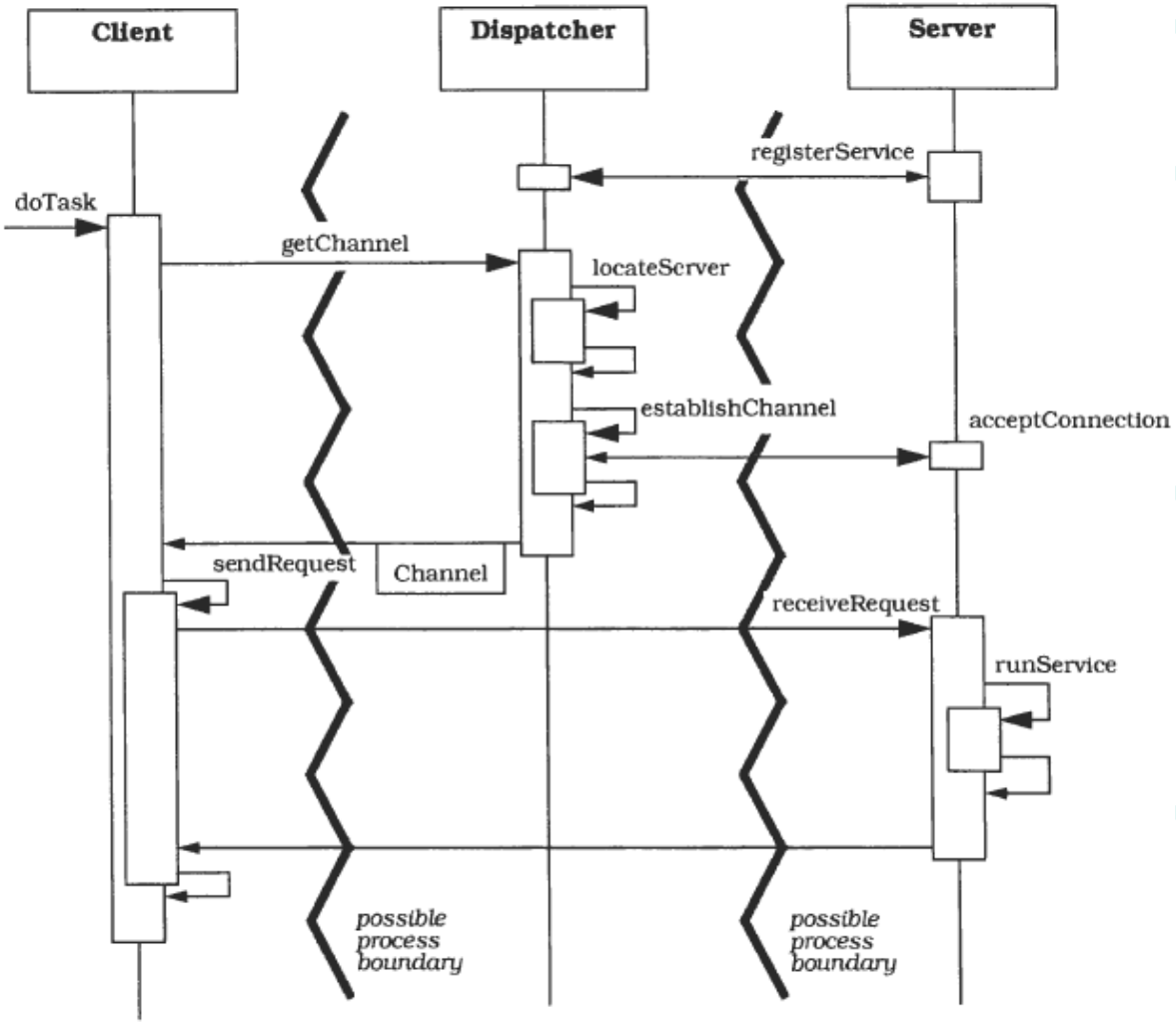
- Empleado en escenarios de comunicación peer to peer
- Desacopla a los clientes del mecanismo de IPC implementado
- Mejora la portabilidad del código cliente a otras plataformas
- Mejora la modificabilidad del mecanismo de IPC subyacente

Client-Dispatcher-Server



- Provee "Location Transparency" para el cliente
- Oculta los detalles del establecimiento del canal de comunicación entre cliente y servidor

Client-Dispatcher-Server

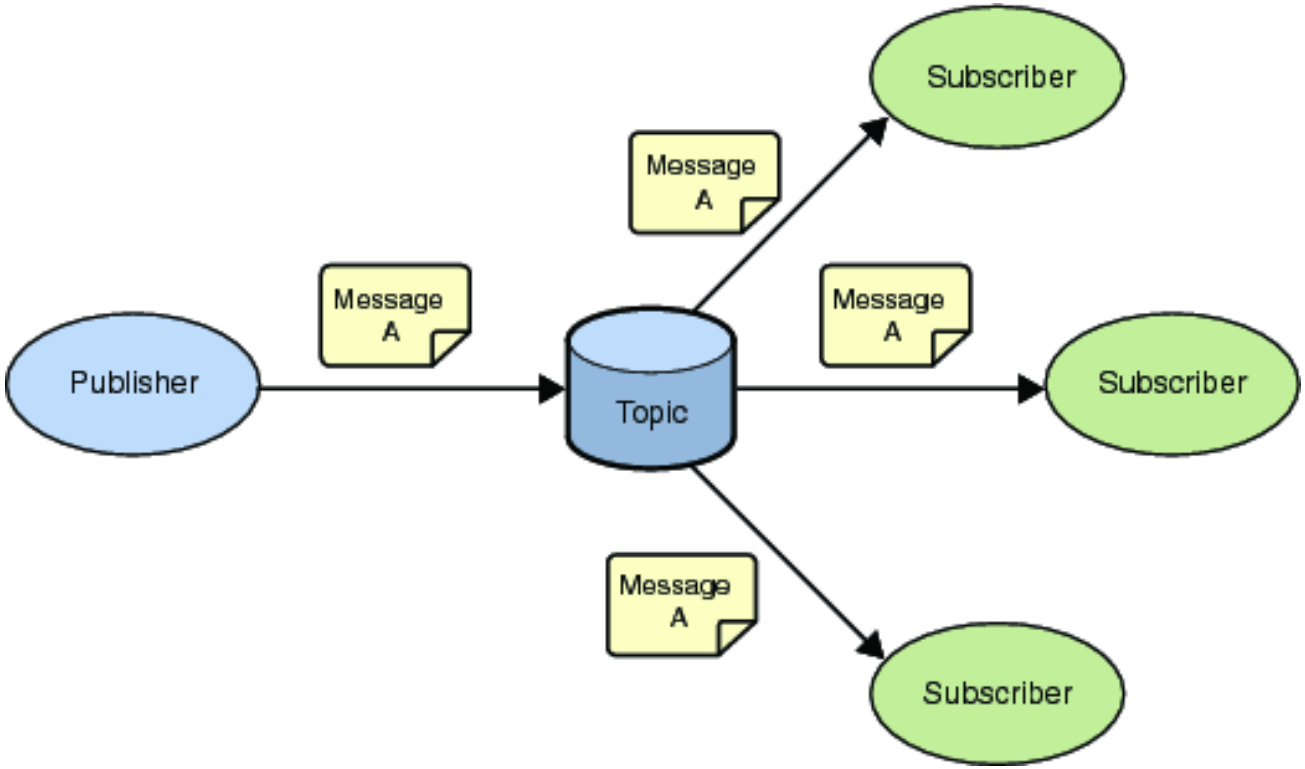


- El servidor se registra con el Dispatcher
- Un cliente solicita al Dispatcher un canal de comunicación para un servidor específico (por nombre)
- El Dispatcher localiza al servidor, establece un canal de comunicación con el mismo y lo retorna al cliente
- El cliente envía una solicitud al servidor de manera directa

Publisher-Subscriber

- Mantiene la sincronización entre componentes cooperativos.
- Un componente "publisher" notifica a muchos consumidores "subscribers" suscriptos a él.
- Análogo al patrón de diseño "Observer".

Publisher-Subscriber



Agenda

#	Tema	Duración
1	Que es un Patrón?	5 min
2	Introducción a estilos arquitectónicos	5 min
2.1	De Estructuración	20 min
2.2	Sistemas distribuidos	5 min
2.3	De Comunicación	10 min
2.4	Sistemas interactivos	15 min
2.5	Sistemas adaptables	10 min
2.6	Otros	5 min
3	Bibliografía	5 min.

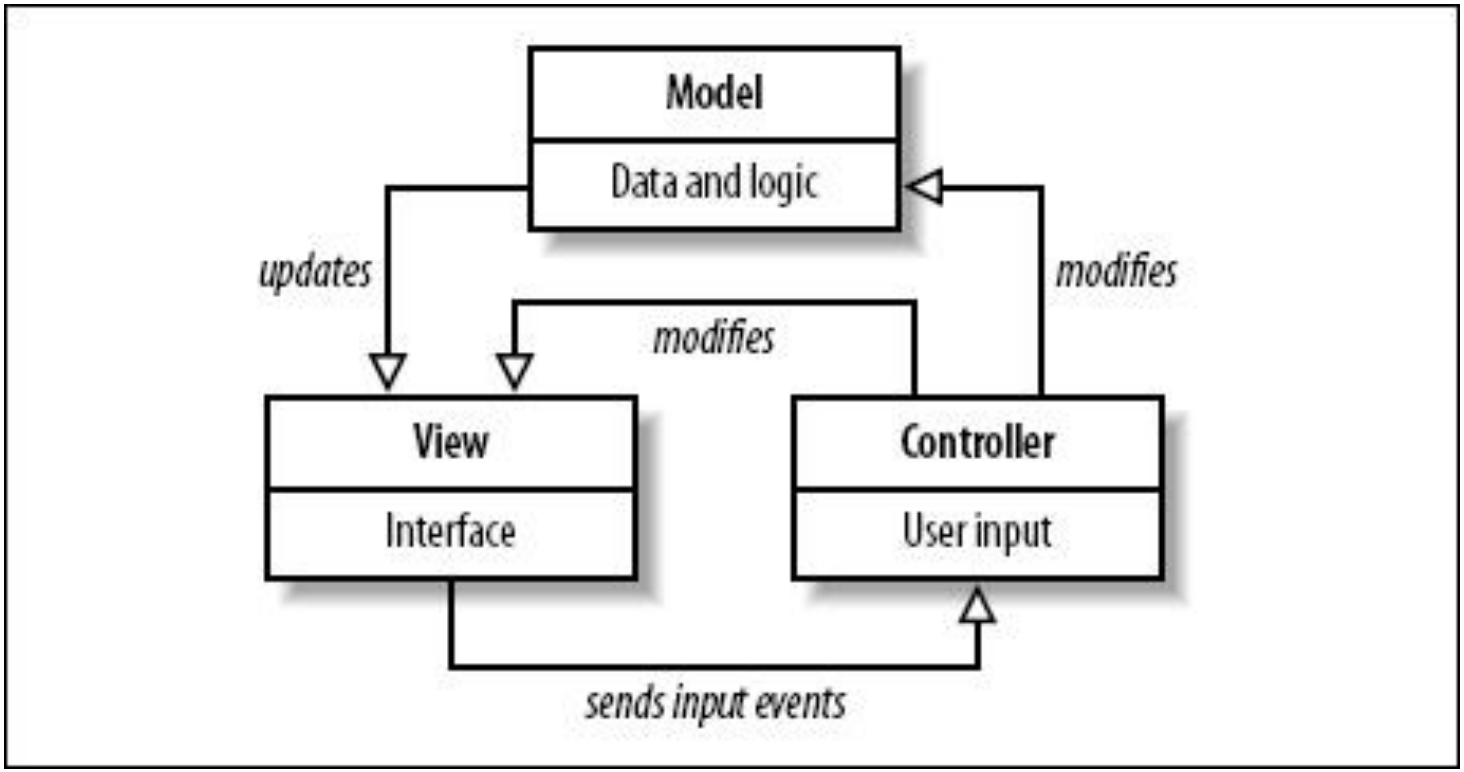
Patrones sobre sistemas interactivos

- Mayoría de software tiene algún tipo de interacción con el usuario
- El objetivo es separar la funcionalidad principal de las vistas de presentación.
- Estilos principales:
  - MVC
  - MVP
  - MVVM

MVC (Model – View – Controller)

- Origen a principios de 80s para Smalltalk
- Divide la aplicación en 3 tipos de componentes: modelos, vistas y controladores.
- Modelo: principal funcionalidad e información
- Vista: muestra información al usuario
- Controlador: controla los inputs del usuario

MVC (Model – View – Controller)



MVC (Model – View – Controller)

- Habitualmente combinado con el patrón de diseño "Observer" para las actualizaciones de vistas.

- Ventajas:
  - Múltiples vistas del mismo modelo
  - Vistas sincronizadas
  - Base potencial para construir un framework

- Desventajas:
  - Número de actualizaciones potencialmente alto
  - Alto acoplamiento entre los tres componentes

Agenda

#	Tema	Duración
1	Que es un Patrón?	5 min
2	Introducción a estilos arquitectónicos	5 min
2.1	De Estructuración	20 min
2.2	Sistemas distribuidos	5 min
2.3	De Comunicación	10 min
2.4	Sistemas interactivos	15 min
2.5	Sistemas adaptables	10 min
2.6	Otros	5 min
3	Bibliografía	5 min.

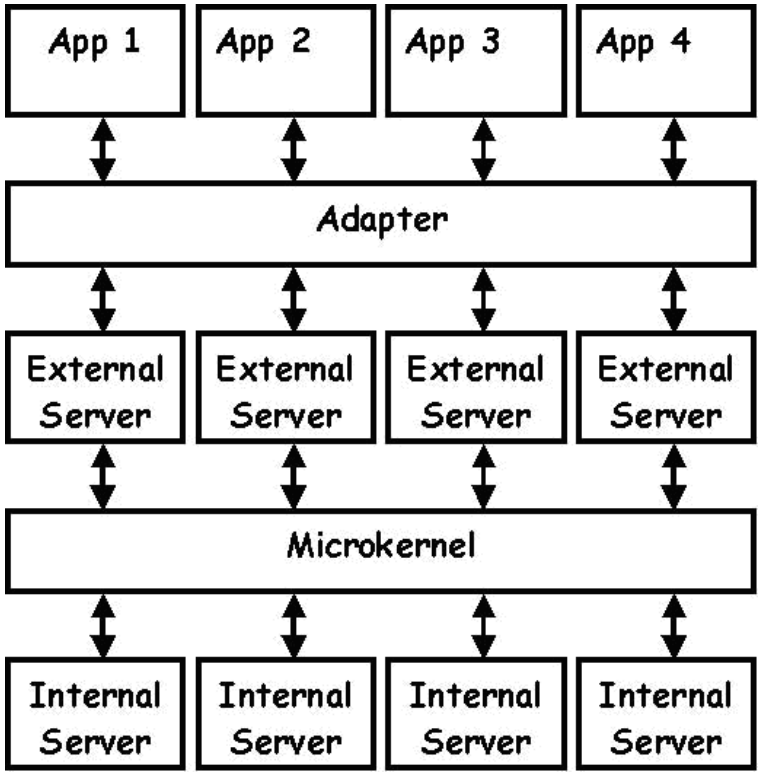
Patrones sobre sistemas adaptables

- Los sistemas de software cambian con el tiempo, así como los sistemas operativos, plataformas, etc
- Se buscan arquitecturas que se adapten a estos cambios a futuro.
- Estilos:
  - Microkernel

Microkernel

- Separa la funcionalidad core de extensiones y de aplicaciones del cliente.
- Es fundamental encontrar un equilibrio entre la cantidad de funcionalidades para el microkernel.
- Utiliza extensiones internas y externas para resolver las funcionalidades esperadas por los clientes.
- Encapsulan la funcionalidad elemental, que puede no cambiar a medida que el software/hardware evoluciona

Microkernel



- Ejemplos: API de un SO, IDEs

Microkernel

- Microkernel: ofrece los servicios fundamentales y administra los recursos.
- Internal Servers: ofrece funcionalidad específica de la plataforma. Son como extensiones del microkernel y se activan bajo demanda.
- External Servers: utiliza el microkernel para implementar su propio punto de vista de la plataforma y ofrece una API de mayor nivel a sus clientes.
- Clients: representan una aplicación que consume los servicios de un Servidor Externo a través de un Adapter.

Agenda

#	Tema	Duración
1	Que es un Patrón?	5 min
2	Introducción a estilos arquitectónicos	5 min
2.1	De Estructuración	20 min
2.2	Sistemas distribuidos	5 min
2.3	De Comunicación	10 min
2.4	Sistemas interactivos	15 min
2.5	Sistemas adaptables	10 min
2.6	Otros	5 min
3	Bibliografía	5 min.

Agenda

#	Tema	Duración
1	Que es un Patrón?	5 min
2	Introducción a estilos arquitectónicos	5 min
2.1	De Estructuración	20 min
2.2	Sistemas distribuidos	5 min
2.3	De Comunicación	10 min
2.4	Sistemas interactivos	15 min
2.5	Sistemas adaptables	10 min
2.6	Otros	5 min
3	Bibliografía	5 min.

Bibliografía

- **"Pattern Oriented Software Architecture: A System of Patterns"**. John Wiley & Sons, 1996. Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., Stal, M.
- **"Software Architecture: Perspectives on an Emerging Discipline"**. Prentice-Hall, 1996. Shaw, M., Garlan, D.
- **"Patterns of Enterprise Application architecture"**. Addison-Wesley, 2002. Fowler, M.
- **"Design Patterns. Elements of Reusable Object-Oriented Software"**. Addison-Wesley, 1995. Gamma, E., Helm, R., Johnson, R., Vlissides, J.