Patrones de Software

Juan Carlos Cerón Barreto
ceron.juan@javeriana.edu.co

Agenda

- Recordando….
- Patrones Estructurales
  - Client / Server
  - Client / Dipatcher / Server
- Patrones de Comunicación

  Client – Dispatcher – Server

Recordando….

- Que es el Diseño?

- El diseño son las justificaciones, consideraciones, restricciones y trade offs que respaldan las decisiones

- No son solo diagramas de UML

- Apoya necesidades del negocio
  - Requerimientos funcionales
  - Requerimientos no funcionales

Recordando….

- Arquitectura de Software

  "La Arquitectura De Software es la estructura o estructuras del sistema que comprende ***elementos de software****, las propiedades visibles de forma externa de estos elementos y las relaciones entre los elementos"*

Recordando….

- Que es un patrón de software?

  "Un **patrón de arquitectura de software** describe un **problema particular recurrente de diseño** que surge en contextos específicos de diseño, y presenta un esquema genérico de solución probado. El **esquema de solución** se especifica describiendo sus componentes constituyentes, sus responsabilidades y relaciones, y las formas en que colaboran."

Recordando….

- 3 Tipos de Patrones

Recordando….

- Elementos de un patrón

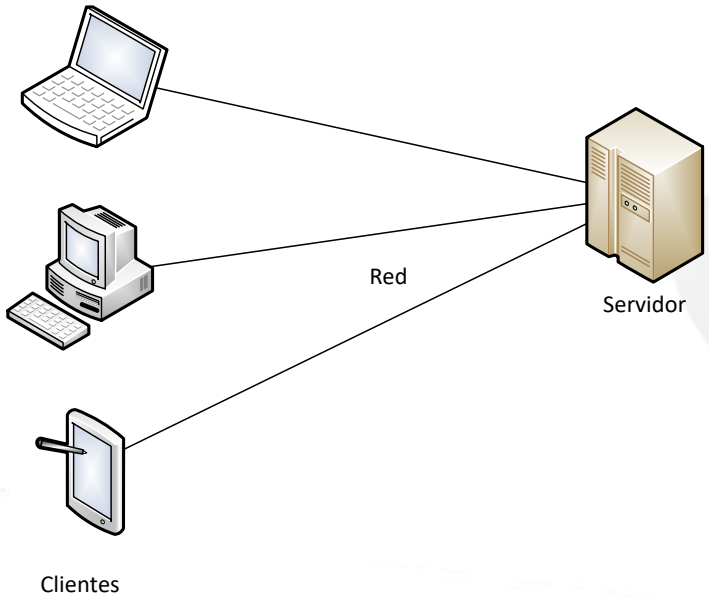
Contexto	• Situación en la que se genera el problema
Problema	• Inconvenientes de diseño
Fuerzas	• Requerimentos por cumplir • Restricciones
Solución	• Elementos, relaciones, responsabilidades y colaboraciones
Consecuencias	• Resultados de aplicar el patrón

PATRONES ESTRUCTURALES


Patrones
Negocio
Arquitectura
Análisis
Diseño
Idiomas
EAI
SOA
Estructurales
Sistemas Distribuidos
…
Estructurales
Control de Acceso
Comunicación
…

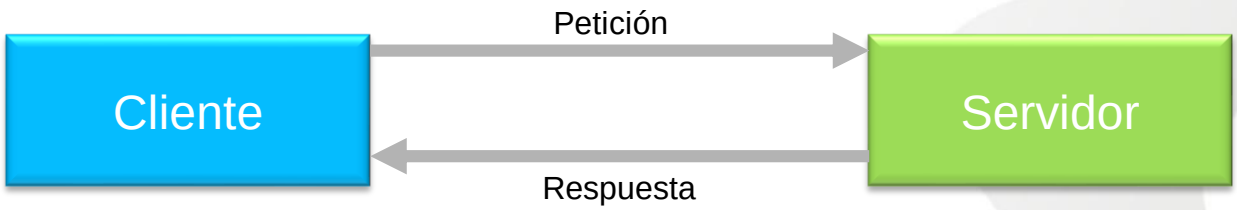
Cliente Servidor

- Es un estilo de arquitectura aplicado a sistemas distribuidos.
- Es un método arquitectural para proporcionar información a un usuario final



Definición

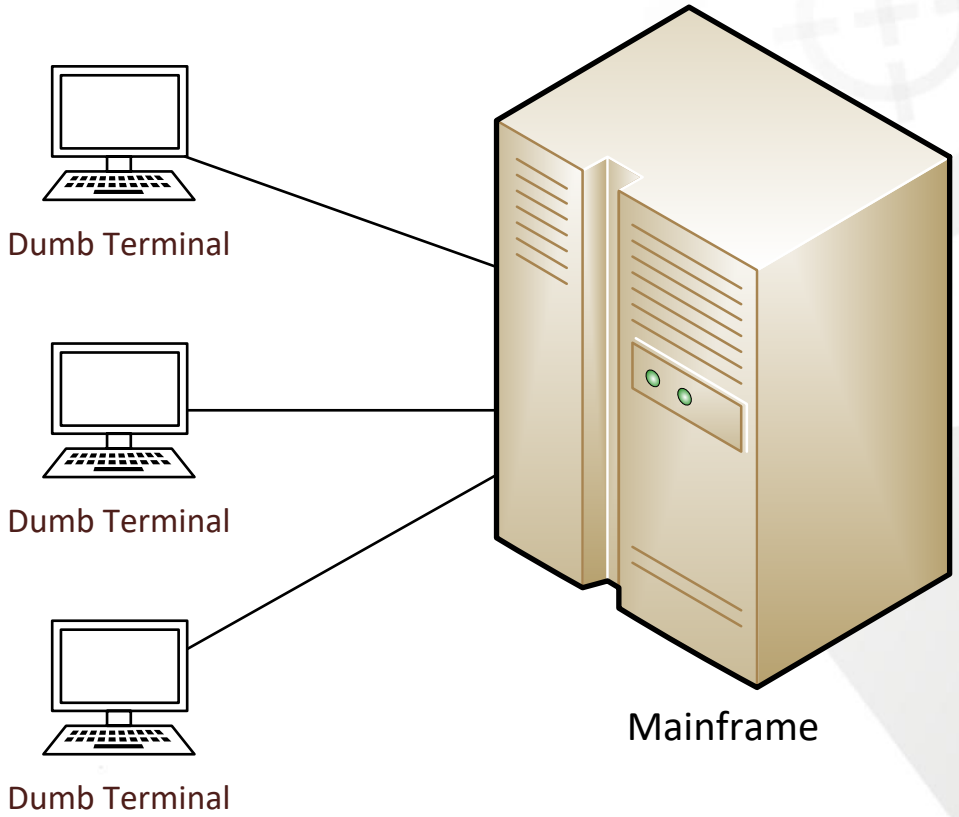
- Sistema distribuido entre múltiples procesadores donde hay clientes que solicitan servicios y servidores que los proporcionan.
- Separa los servicios situando cada uno en su plataforma más adecuada.



Características

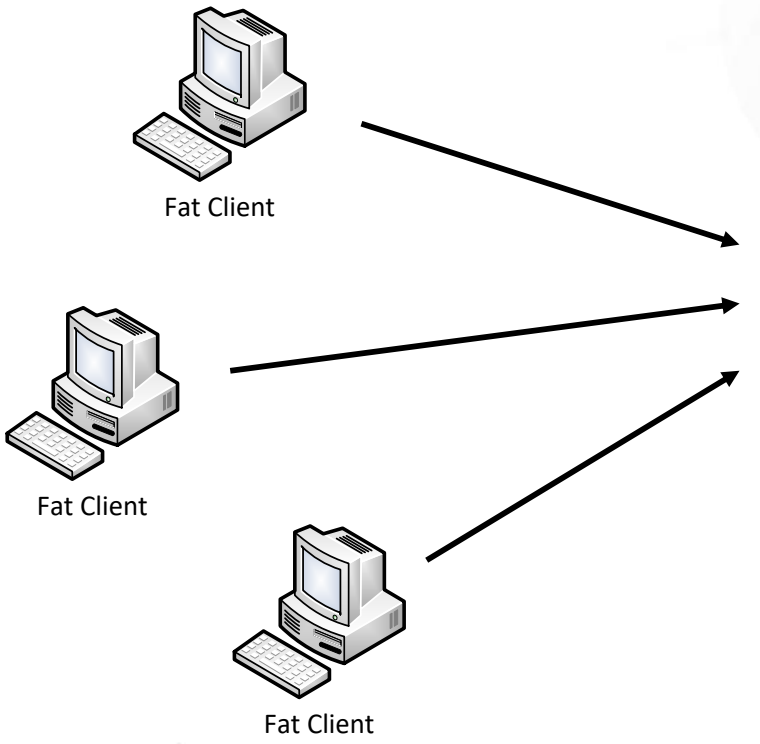
- Procesamiento en **hilos separados**
- **Sobrecarga IPC (Inter Process Communication)**
  - Marshall unmashall (request/ response)
  - Tráfico de la red
- **Componentes intermedios**
  - Caché, Seguridad, Balanceo de carga, ...
- **Callbacks**
  - Notificación de eventos
  - Transición al modelo 2-peer
- **Session**
  - Servidores sin estado
  - Servidores con estado

Arquitecturas C/S Monolíticas One - Tier



Arquitecturas C/S
Two - Tier

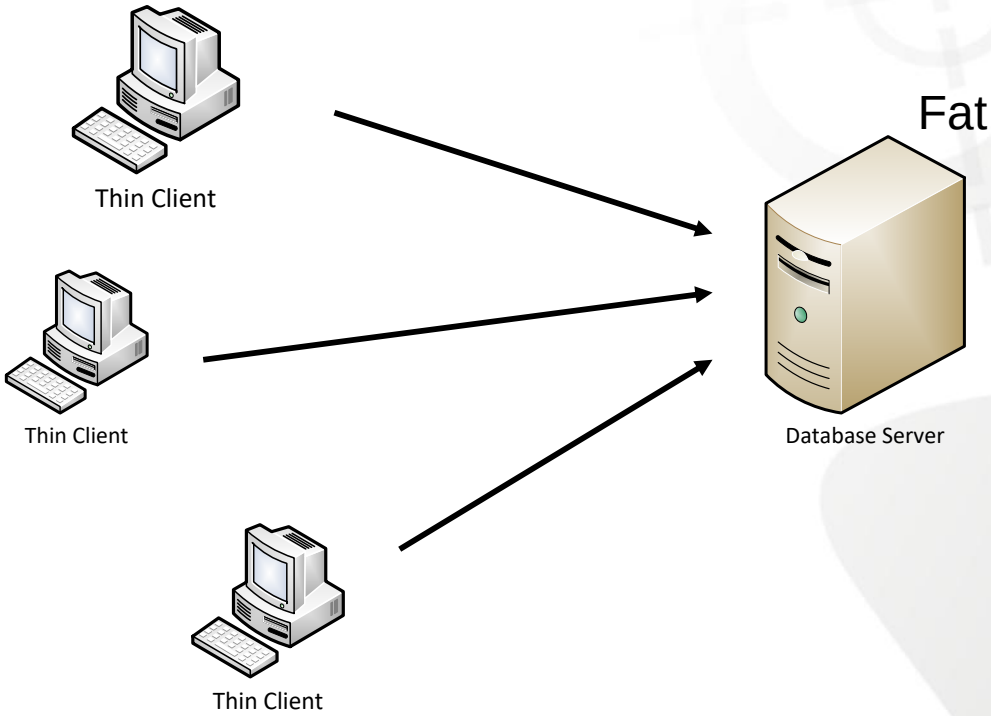
Fat Clients
- dBase
- FoxPro
- Clipper
- Paradox



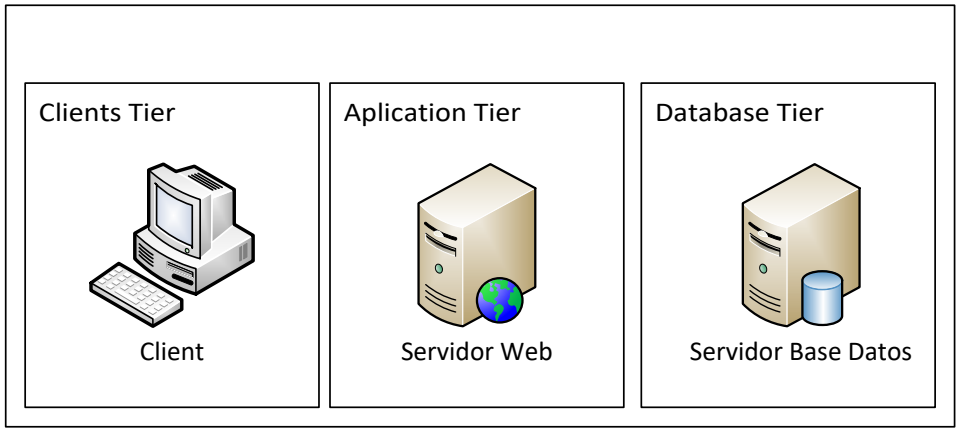
Arquitecturas C/S
Two - Tier

GUI

Visual Basic
PowerBuilder
Delphi

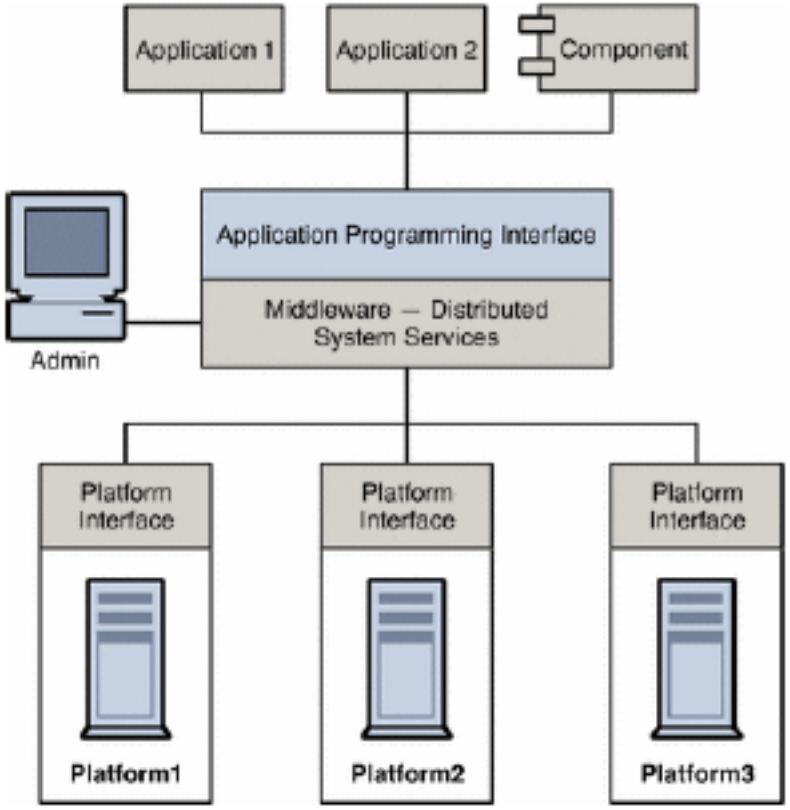


Arquitecturas C/S Three - Tier



- Independencia
- Requerimientos
  - Cuellos de Botella red
  - Pool de Conexiones
- Lenguajes de alto nivel en la Capa de Aplicación
- Componentes Adicionales: Cache, Seguridad, LB

Arquitecturas C/S
Three- Tier / Fat Middle



- Manejo de Protocolos
- Traducción de Transacciones
- Transparencia localización
- Procesos => Aplicaciones
- Respuesta a Cambios

Objetivos Client Server

- Localización transparente.
- Recursos compartidos.
- Escalabilidad
  - Horizontal: > nº estaciones.
  - Vertical: migración a otras plataformas.
- Interoperabilidad entre distintos Hw.y Sw
- Transparentes: ocultan el hecho de que los recursos estén distribuidos
  - Transparencia de acceso
  - Transparencia de ubicación y migración
  - Transparencia de replicación
  - Transparencia ante fallos

Ventajas / Inconvenientes

- Ventajas
  - Economía: compartir recursos costosos. Ahorro de costos
  - Flexibilidad: (depende del # tiers)
  - Fiabilidad: tolerancia a fallos - recursos críticos pueden ser replicados
- Escalabilidad: no limitado a recursos de un único equipo. posibilidad de introducción de nuevos
- Invonvenientes
  - Dificultad en el desarrollo del software –Limitaciones de las redes (ancho de banda, latencias, ...)
  - Problemas de seguridad: control de accesos, confidencialidad, Integridad.

PATRONES DE COMUNICACIÓN


Patrones
Negocio
Arquitectura
Análisis
Diseño
Idiomas
EAI
SOA
Estructurales
Sistemas Distribuidos
…
Estructurales
Control de Acceso
Comunicación
…

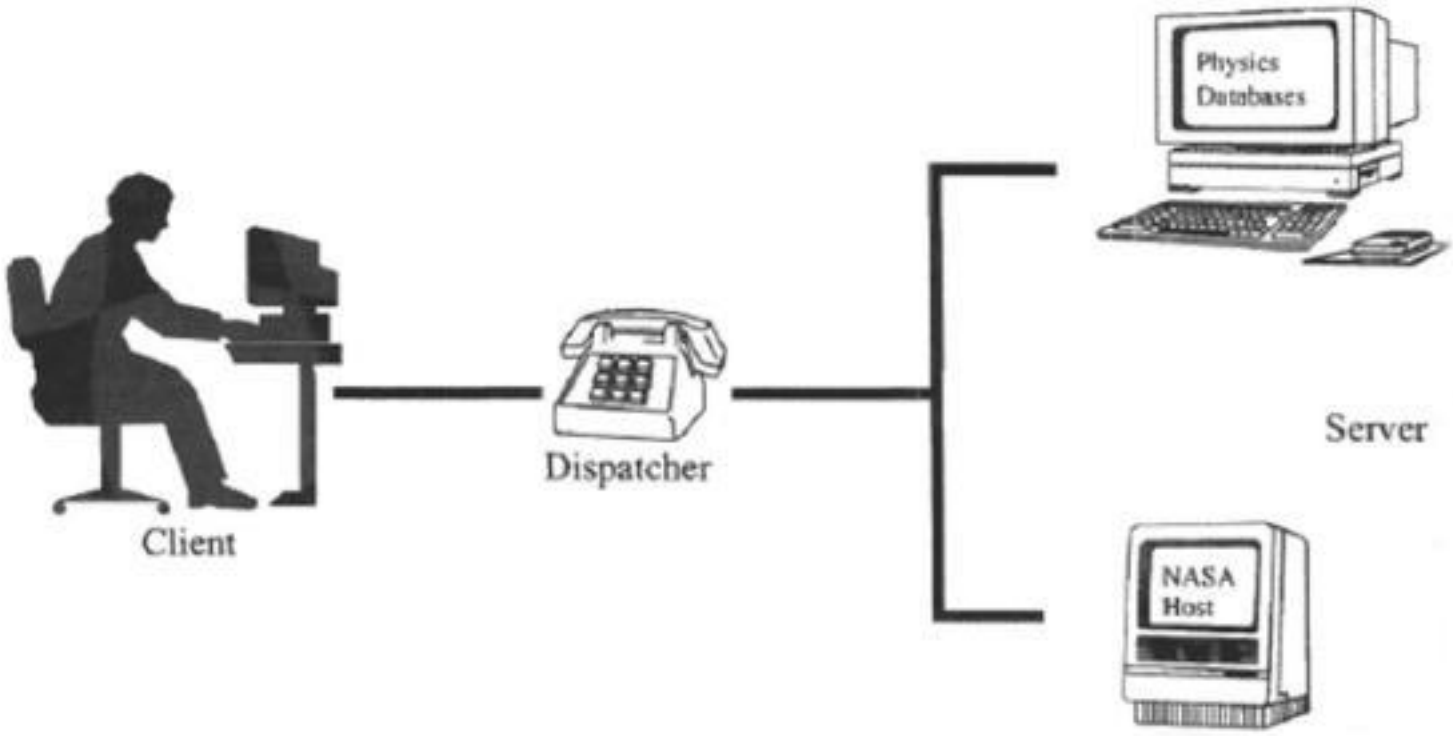
Patrones de Comunicación

- Necesidad de compartir y utilizar recursos de la red
- Maquinas rápidas y costosas que permiten alojar servicios centrales como administración de base de datos
- Maquinas livianas accediendo a servicios de otras maquinas pesadas de forma remota
- Encapsulamiento de la comunicación

Patrones de Comunicación

- Acoplamiento al mecanismo de comunicación
- Portabilidad
- Migración de subsistemas depende de los mecanismos de comunicación que ofrece

Client – Dispatcher - Server
Caso de Estudio



Contexto

- Integración de un cliente con un conjunto de servidores distribuidos
  - servidores corriendo localmente o distribuidos en la red.
- Se requiere una comunicación sincrónica entre las partes.

Problema

- Fuerzas a balancear
  - Un componente puede utilizar un servicio independiente de la ubicación del proveedor de servicios (servidor)
  - El código de la aplicación de la base funcional de un cliente debe ser independiente del código usado para establecer una conexión con el servidor

Solución

- Introduce una capa intermedia entre los clientes y los servidores, el componente despachador.
- El despachador proporciona transparencia de ubicación
  - Servicio de nombres
  - Oculta los detalles del establecimiento de la comunicación

Solución

- El despachador es responsable de establecer el canal de comunicación entre el cliente y el servidor.
- Se permiten reconfiguraciones de ubicación sin impactos en clientes y servidores

Estructura

Class Dispatcher	***Collaborators***
Responsibility • Establishes communication channels between clients and servers. • Locates servers. • (Un-)Registers servers. • Maintains a map of server locations.	• Client • Server

Estructura

Class Client	**Collaborators**
Responsibility • Implements a system task. • Requests server connections from the dispatcher, • Invokes services of servers,	• Dispatcher • Server

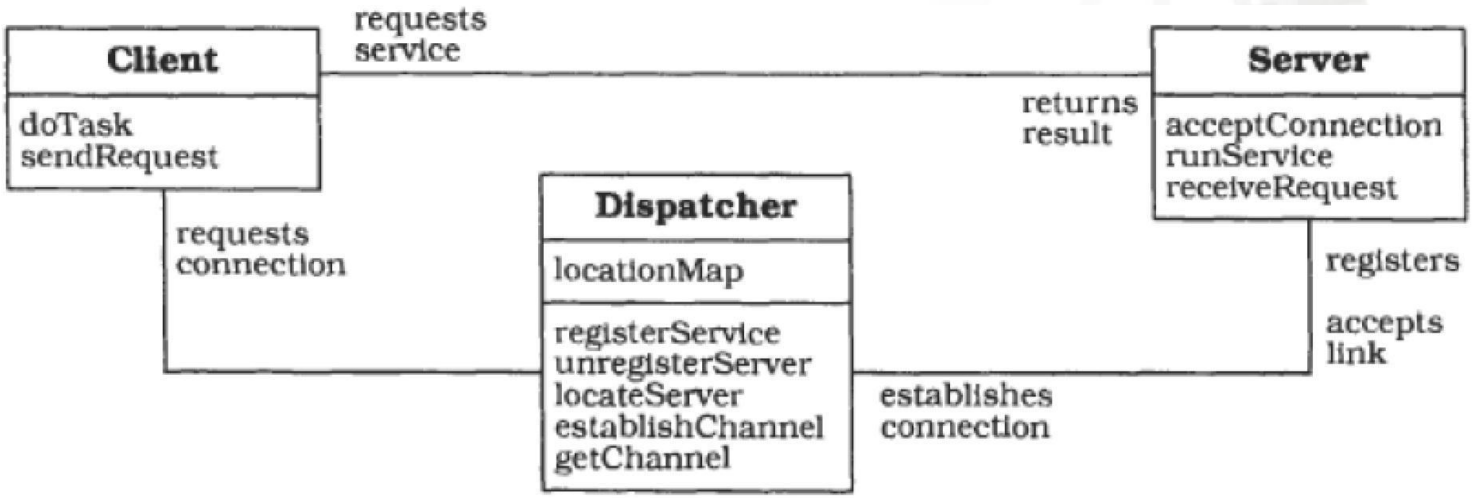
Class Server	**Collaborators**
Responsibility • Provides services to clients. • Registers itself with the dispatcher.	• Client • Dispatcher

Estructura

Class Client	**Collaborators**
Responsibility • Implements a system task. • Requests server connections from the dispatcher, • Invokes services of servers,	• Dispatcher • Server

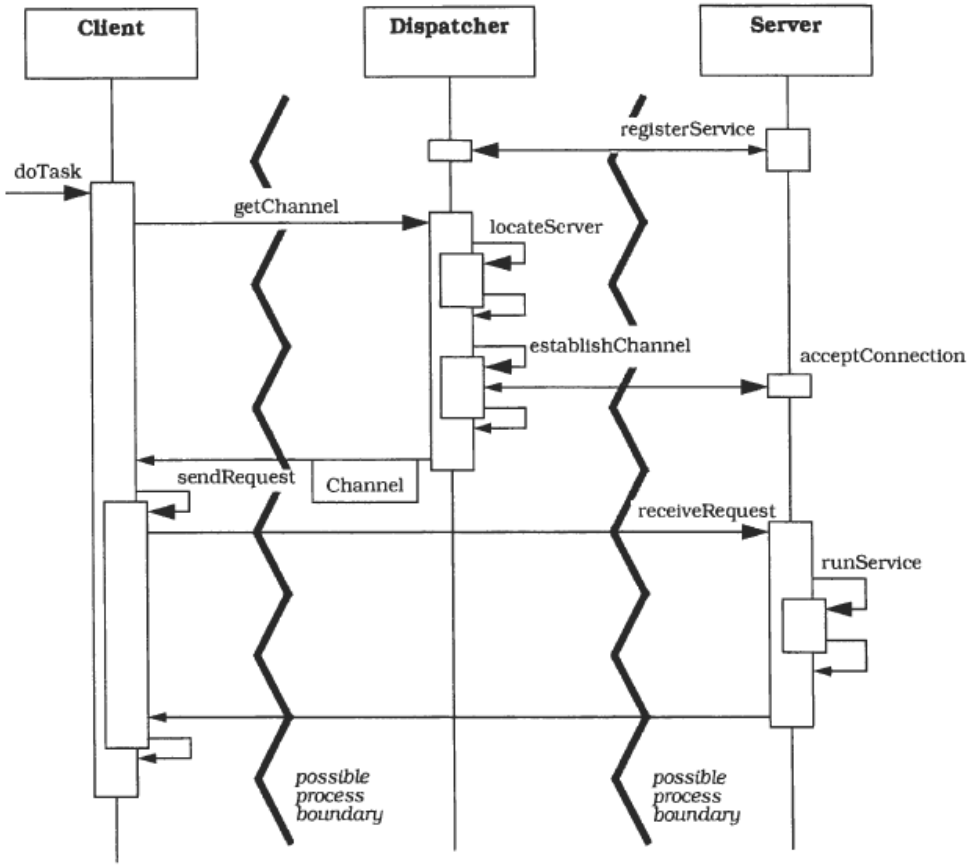
Class Server	**Collaborators**
Responsibility • Provides services to clients. • Registers itself with the dispatcher.	• Client • Dispatcher

Estructura



Estructura del patrón Client-Dispatcher-Server

Dinámica


Client
Dispatcher
Server
registerService
doTask
getChannel
locateServer
establishChannel
acceptConnection
sendRequest
Channel
receiveRequest
runService
possible process boundary
possible process boundary

Implementación

- Separe la aplicación en clientes y servidores.
- Seleccionar servicios de comunicación requeridos.
- Especifique los protocolos de comunicación entre los componentes
- Decida como denominar los servidores.
- Diseñe e implemente el despachador.
- Implemente los componentes cliente y servidor

Variantes

- Despachadores distribuidos.
- Comunicación manejada por los clientes
- Canales de Comunicación heterogéneos
- Client Dispatcher Services

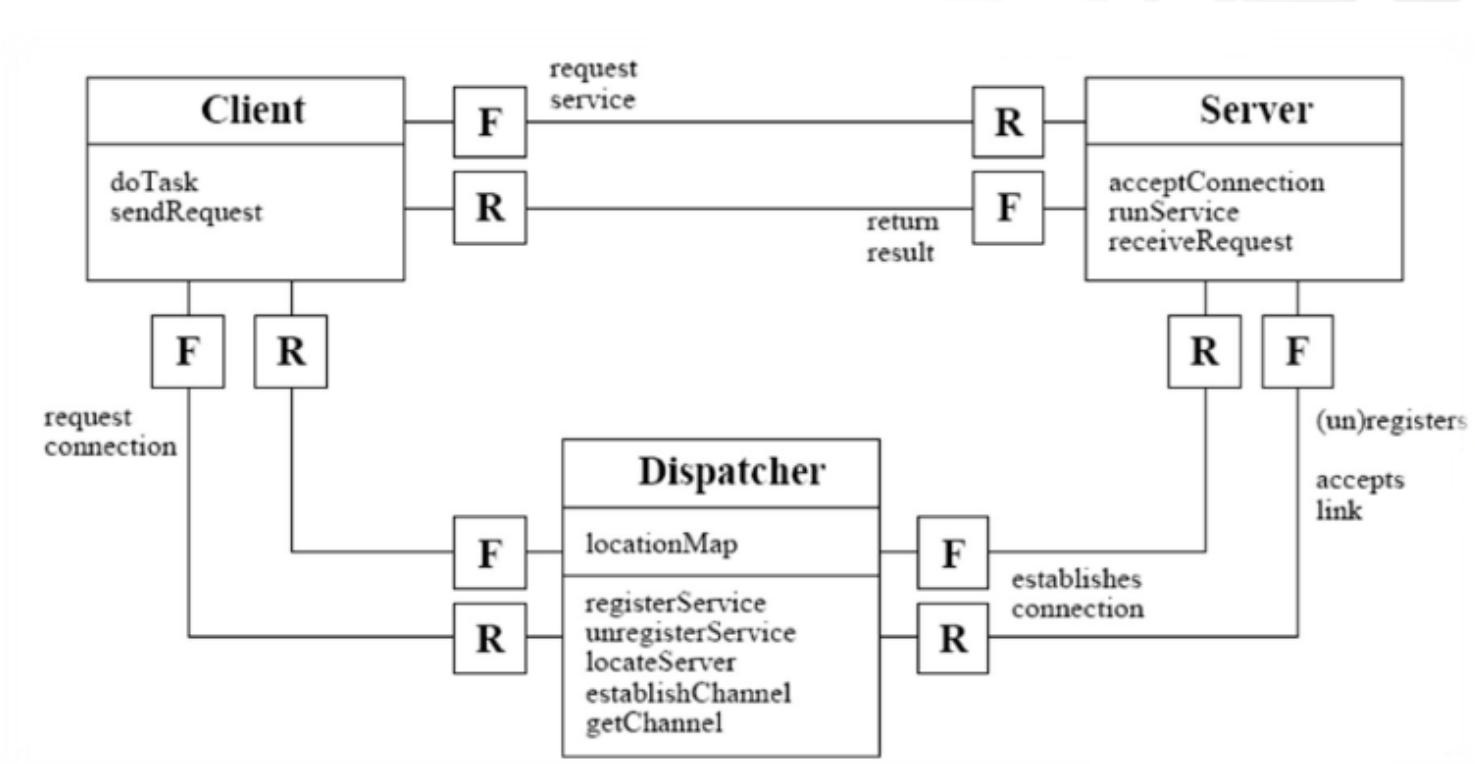
Consecuencias

- **Beneficios**
  - Intercambiabilidad de servidores.
  - Transparencia de ubicación y migración.
  - Reconfiguración
  - Tolerancia a fallas.

Restricciones

  - Menor eficiencia a causa de la indirección y el establecimiento explícito de conexiones.
  - Sensibilidad a cambios en las interfaces del despachador

Complementando Patrones…..



Trabajo en Clase

1. Formular un ejemplo que aplique para el estilo de arquitectura cliente servidor
2. Presentar un diseño de alto nivel
3. El ejemplo puede ser del trabajo o productos existentes en el mercado

Bibliografía

- Pattern-Oriented Software Architecture: A System of Patterns, F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, M. Stal. John Wiley & Sons, 1996
- Patterns of Enterprise Application Architecture: Martin Fowler, Addison-Wesley Professional, 1 edition ,November 15, 2002.