ESTILOS Y PATRONES ARQUITECTONICOS

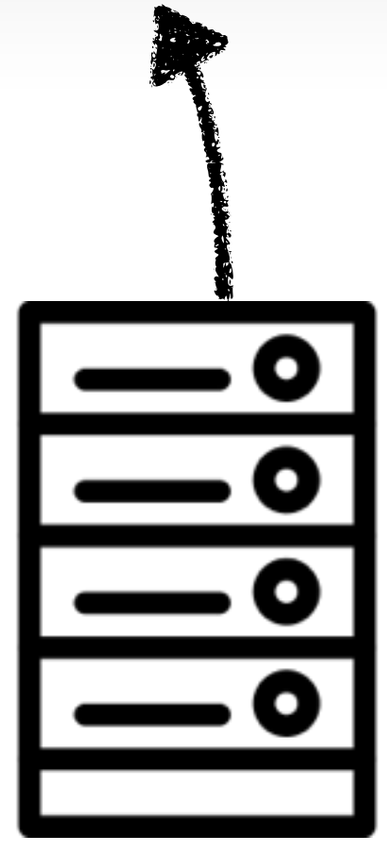
ISIS 3710

Aplicaciones Web

Web browser/cliente http

Cliente

Servidor/Contenedor web



Servidor

Componentes del lado del cliente: HTML, CSS, JS

Componentes del lado del servidor: JSP, JSF, PHP, EJBs

Aplicaciones Web

Web browser/cliente http

Cliente

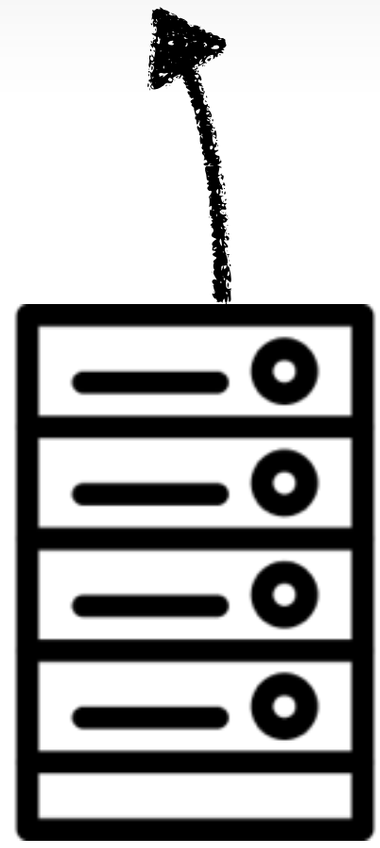
- Lógica de presentación
- Clientes ligeros
- Componentes se ejecutan en el browser
- Interacción con el usuario

Componentes del lado del cliente: HTML, CSS, JS

Aplicaciones Web

- Lógica de negocio
- Procesamiento computacionalmente costoso
- Bases de datos SQL/NoSQL
- CDNs: content deliver network
- Motores de minería de datos, BI

Servidor/Contenedor web



Servidor

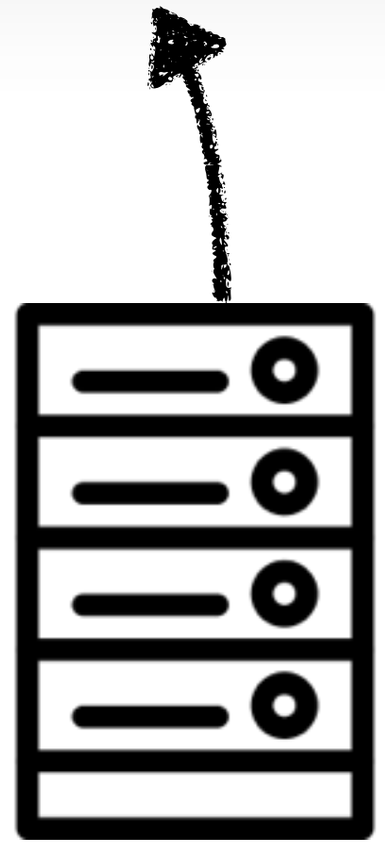
Componentes del lado del servidor: JSP, JSF, PHP, EJBs

Aplicaciones Web

Web browser/cliente http

Cliente

Servidor/Contenedor web



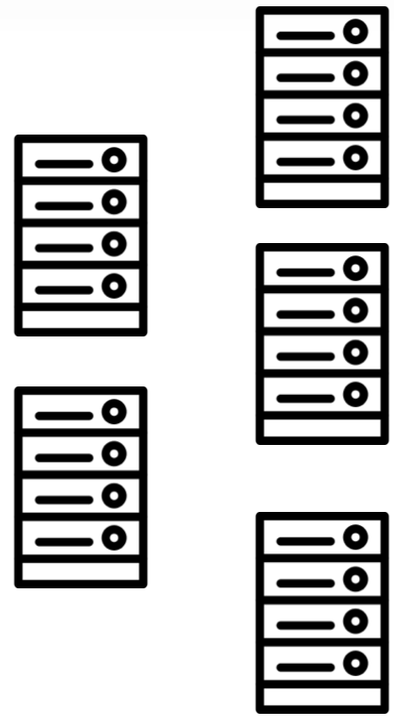
Servidor

Aplicaciones Web

Web browser/cliente http

Cliente

Servidor/Contenedor web



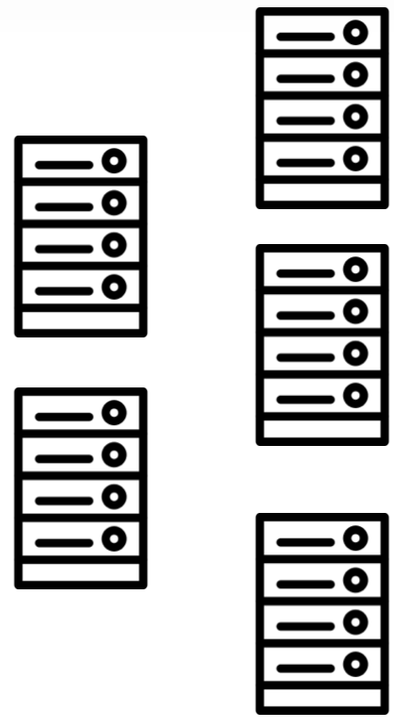
Servidores, Servicios virtualizados, Servicios en cloud

Aplicaciones Web

Cliente rico
que accede a servicios
vía http

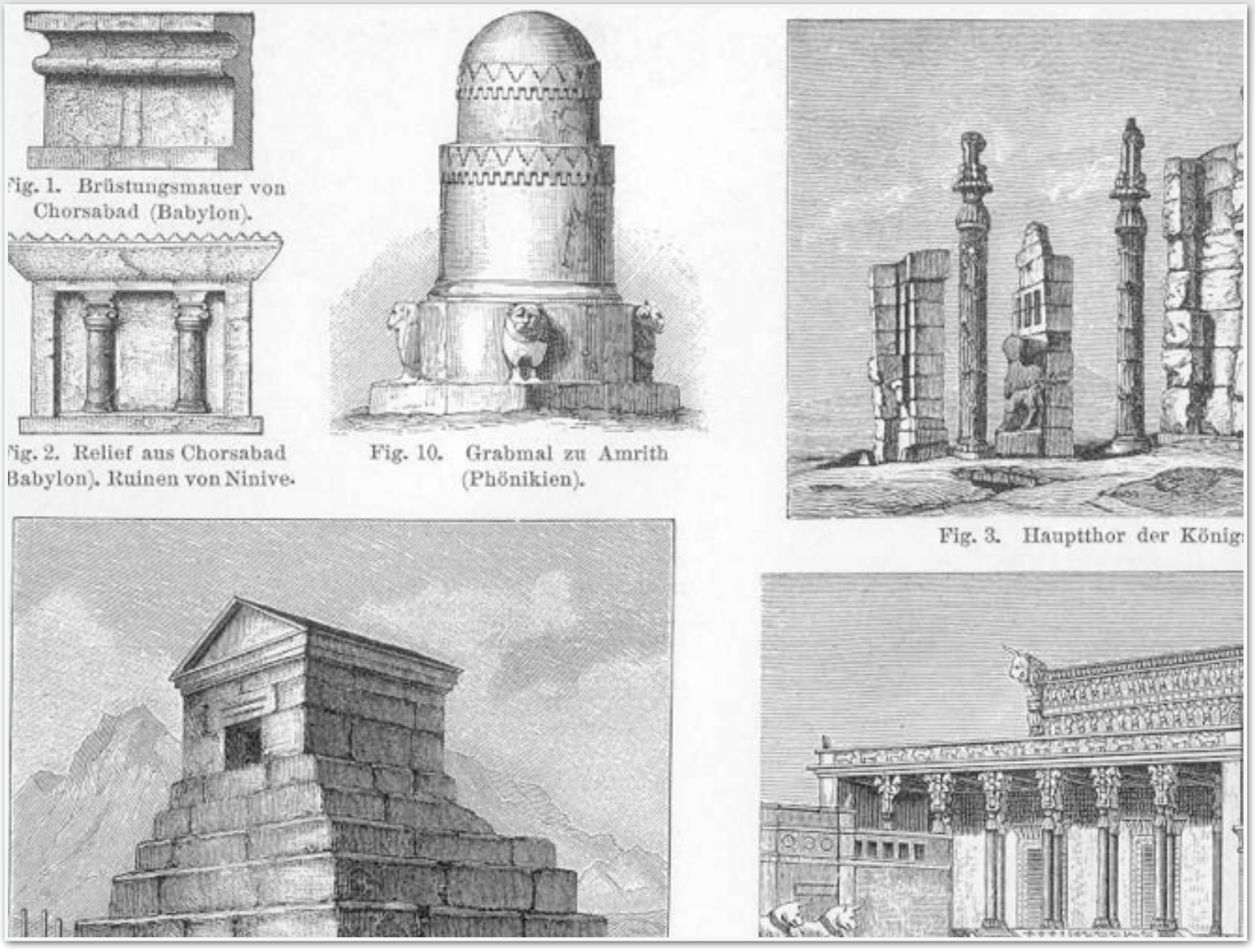
Cliente

Servidor/Contenedor web



Servidores, Servicios
virtualizados, Servicios en
cloud

Estilos arquitectónicos



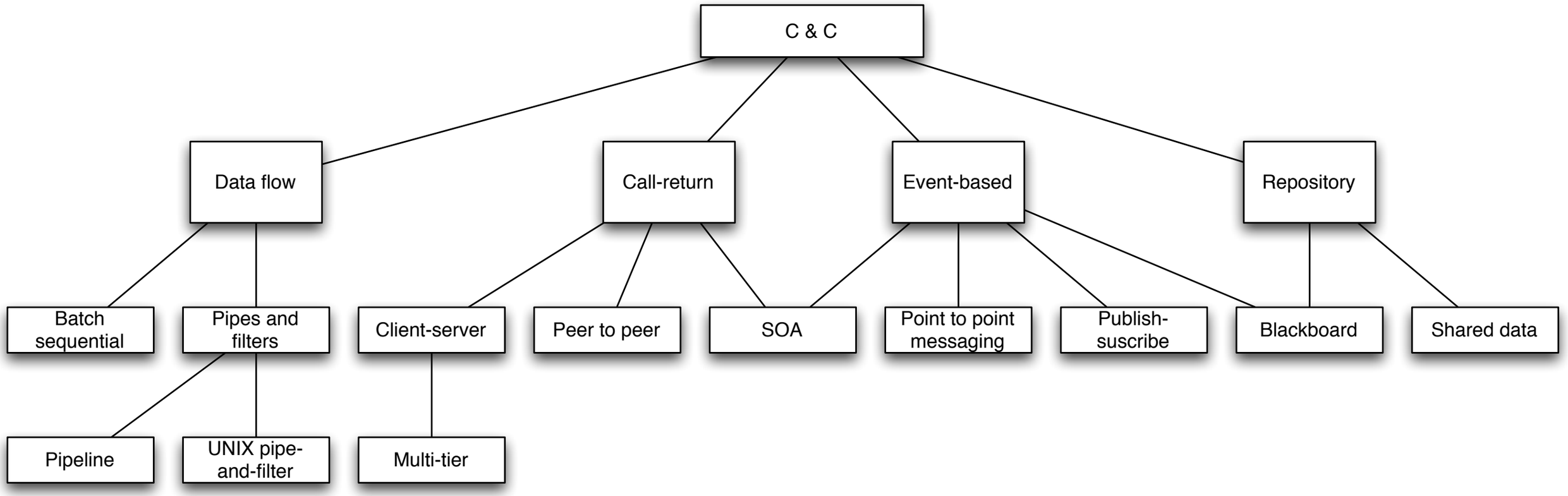
http://www.vintage-views.com/architectural-styles/

Estilos arquitectónicos

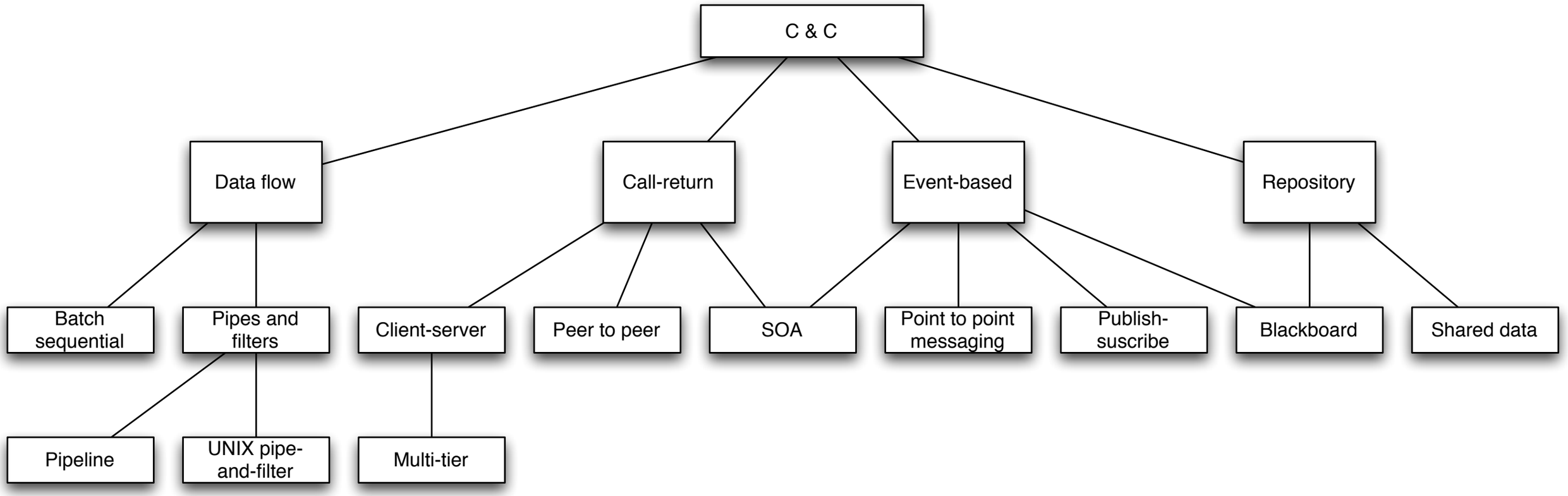
Conjunto de decisiones que explican una aproximación genérica para el diseño de un sistema de software:

- Componentes
- Conectores (interacciones entre componentes)
- Restricciones acerca de cómo deben ser combinados los componentes y conectores
- Modelo computacional asociado al estilo

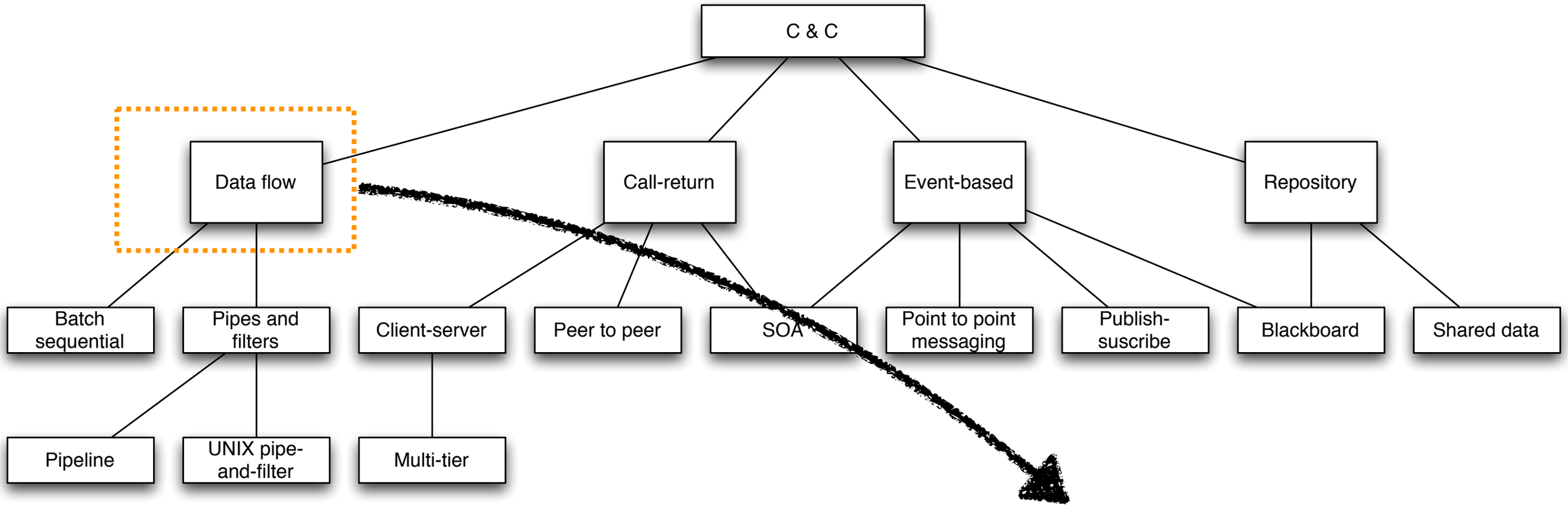
Estilos basados en componentes y conectores



Estilos basados en componentes y conectores

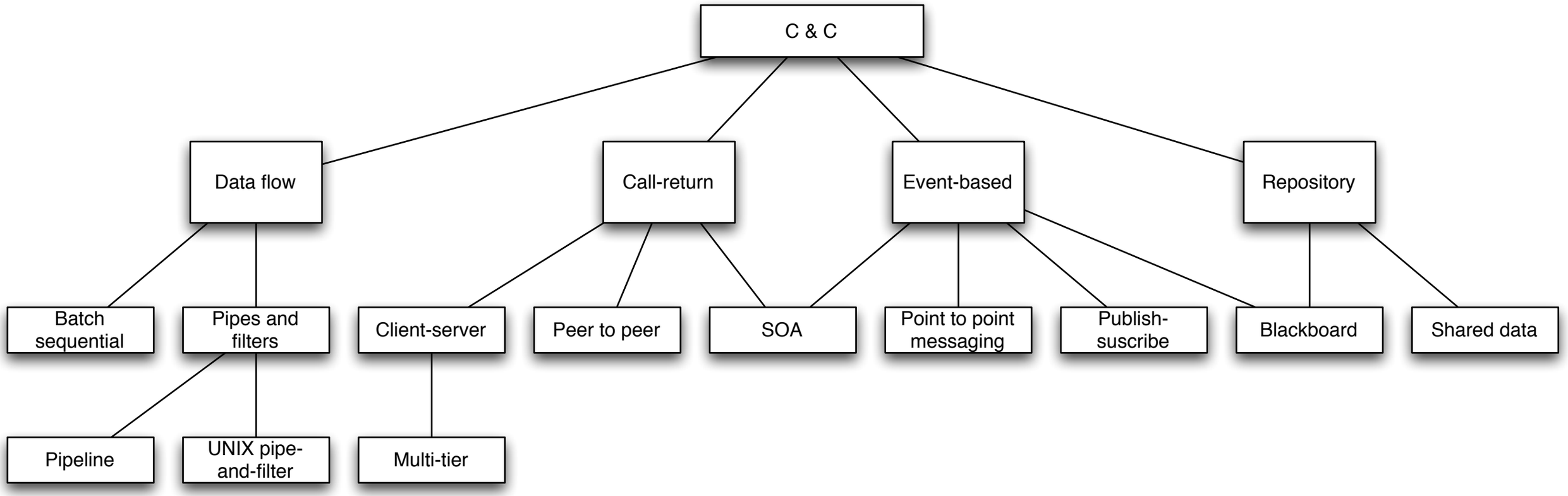


Estilos basados en componentes y conectores

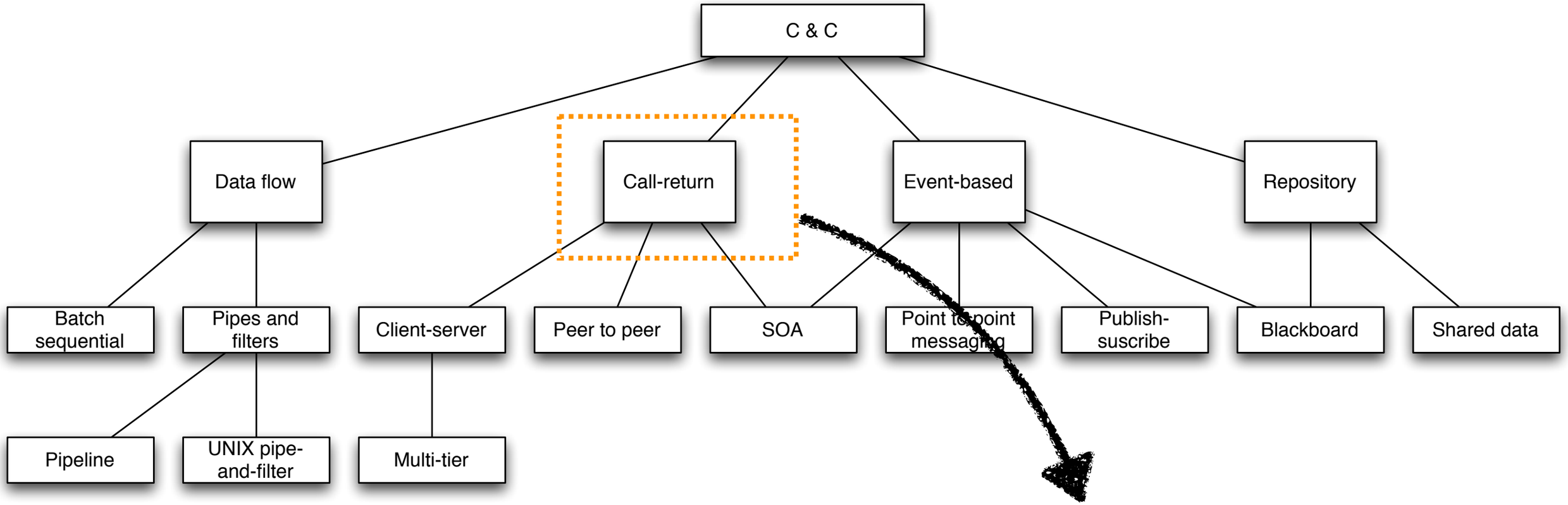


La computación/ procesamiento es dirigido por el flujo de datos en el sistema

Estilos basados en componentes y conectores

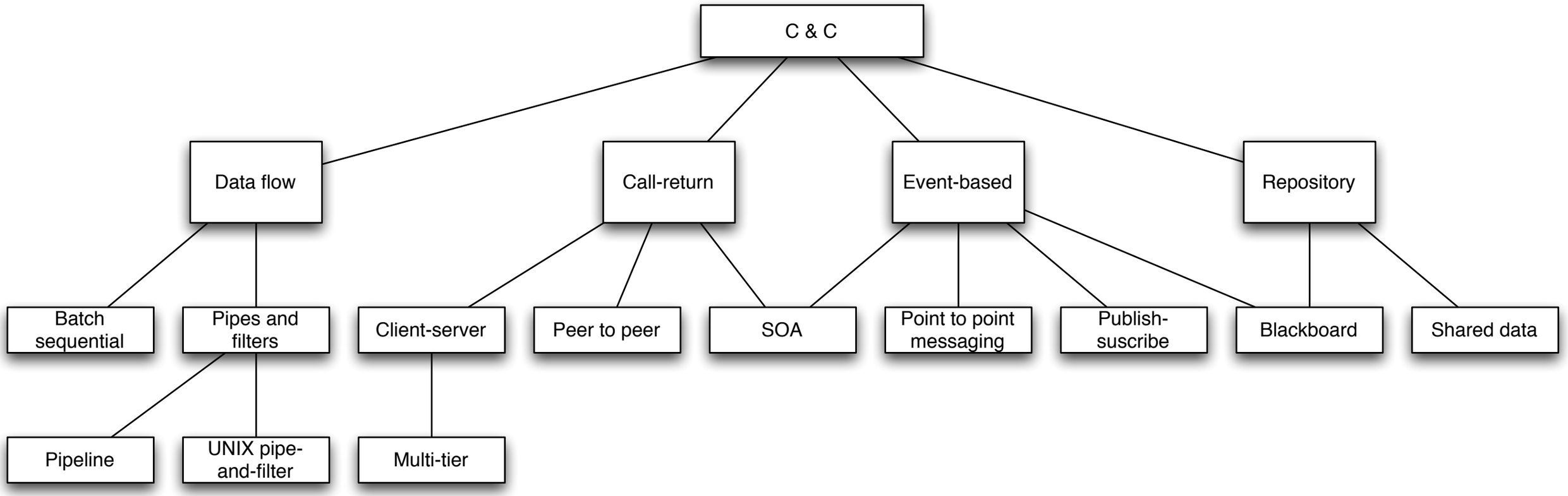


Estilos basados en componentes y conectores

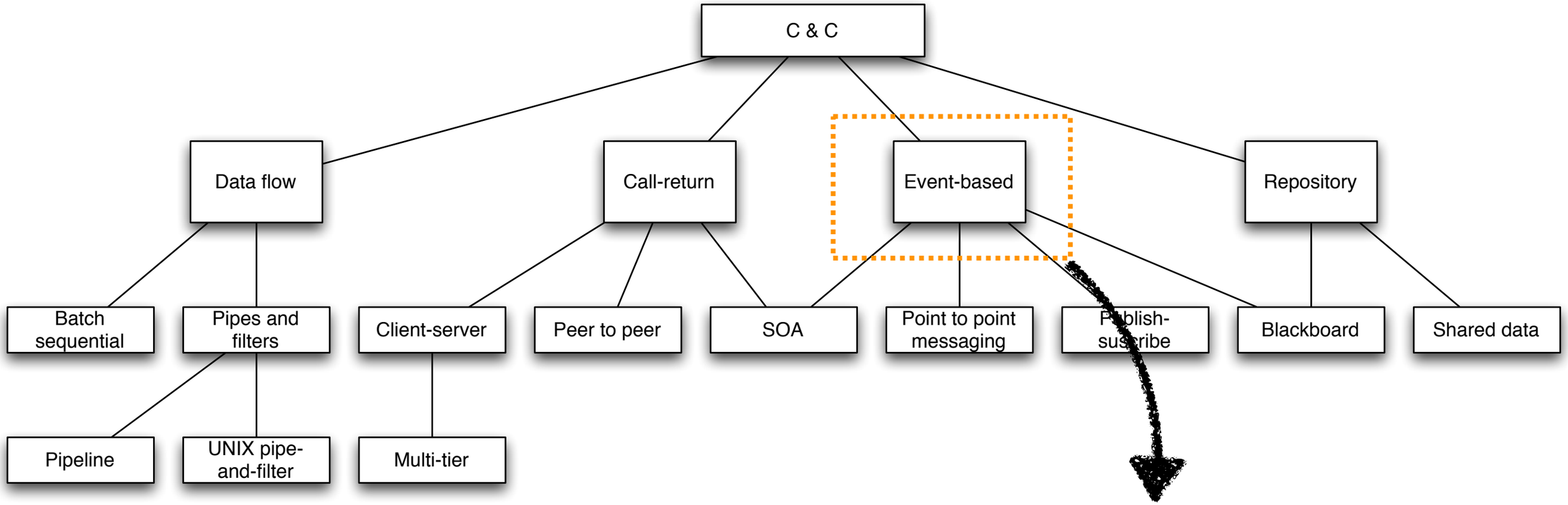


Interacción por invocación síncrona de capacidades proporcionadas por los componentes

Estilos basados en componentes y conectores

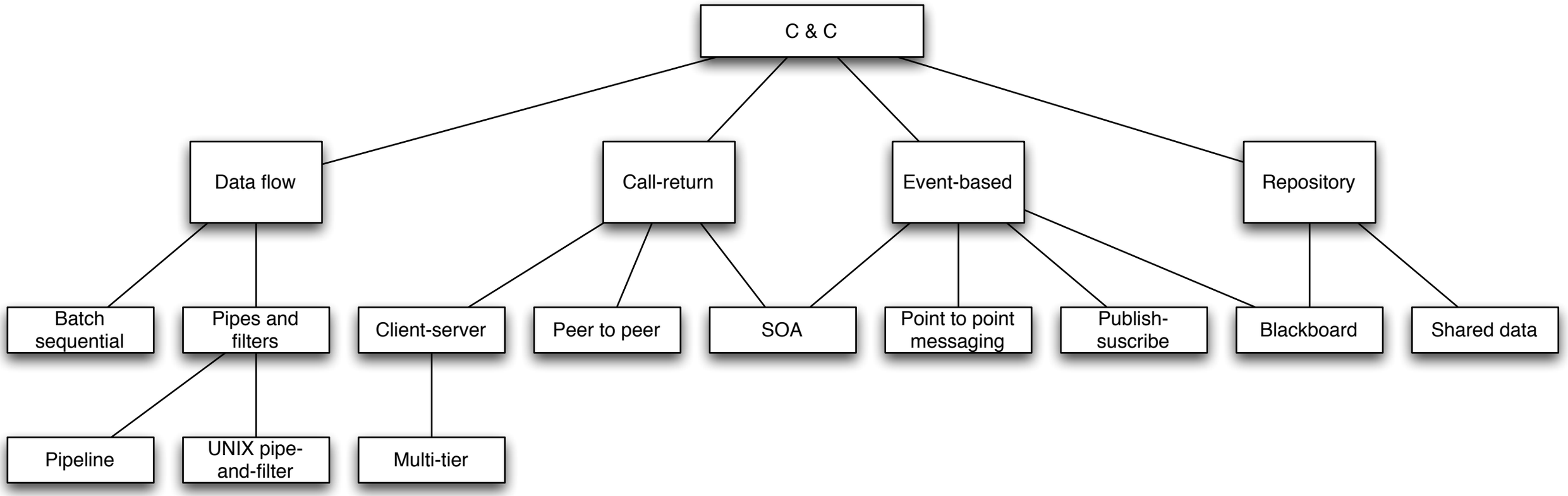


Estilos basados en componentes y conectores

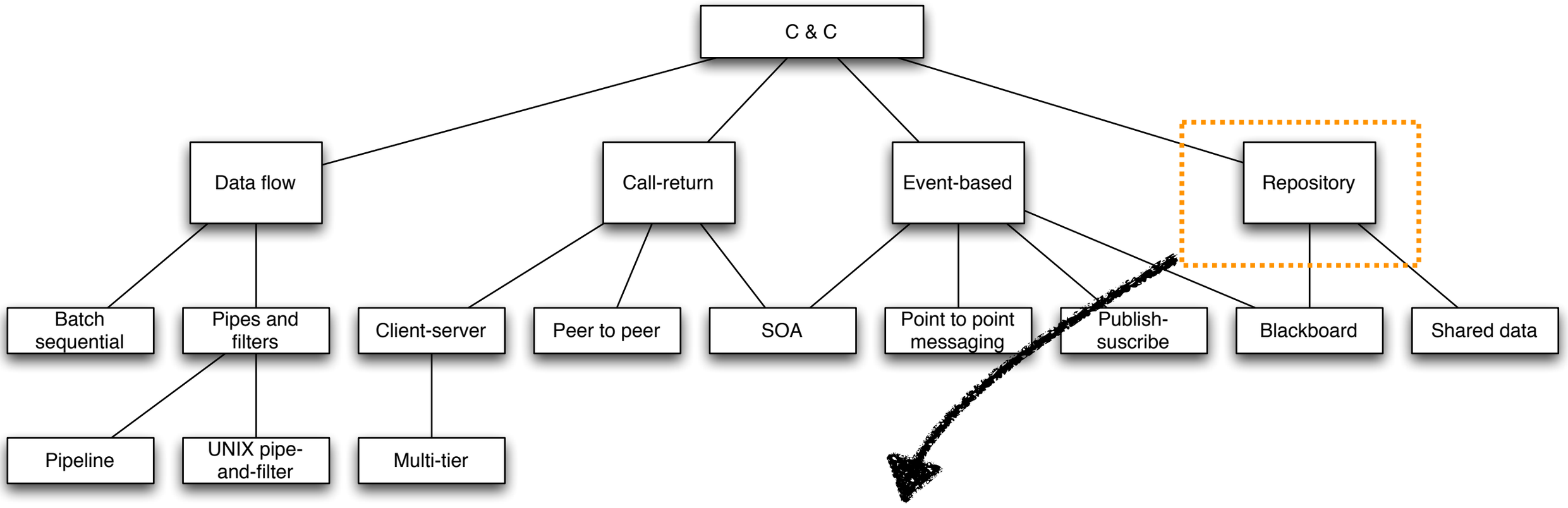


Los componentes interactúan mediante eventos/mensajes asíncronos

Estilos basados en componentes y conectores

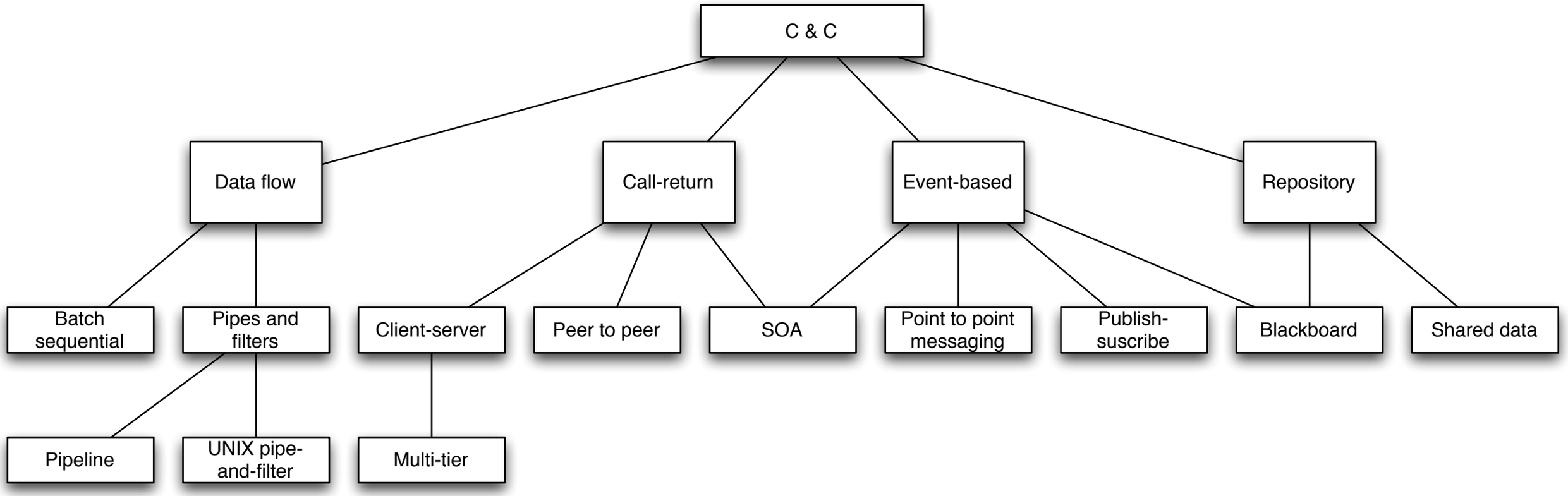


Estilos basados en componentes y conectores



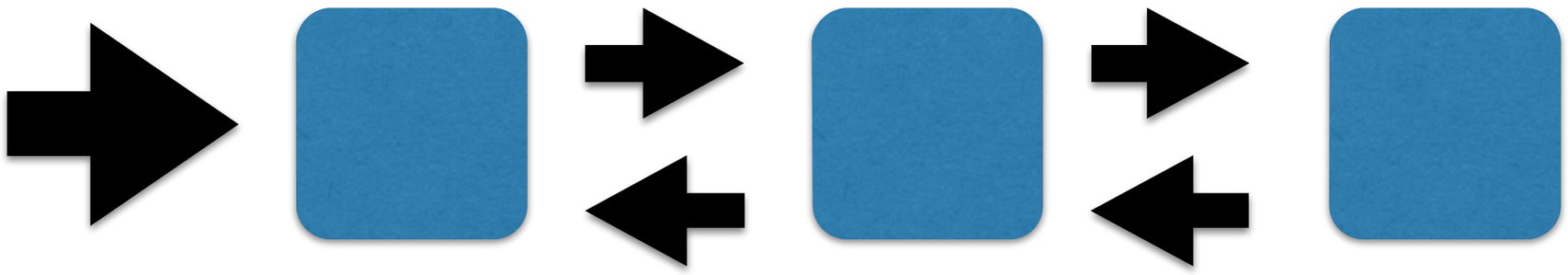
Los componentes interactúan mediante colecciones de datos compartidos y persistentes

Estilos basados en componentes y conectores



Call return

- Representa un modelo computacional en el cual los componentes proporcionan servicios que pueden ser invocados por otros de forma síncrona.
- Un componente que invoca un servicio permanece en estado de bloqueo mientras el servicio termina.

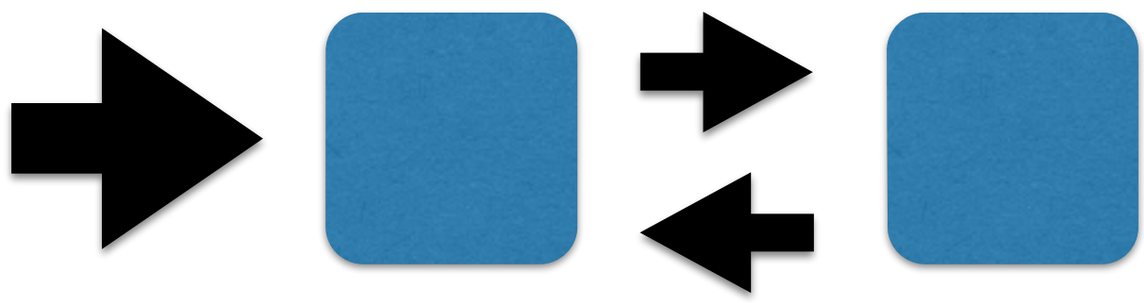


Call return: cliente servidor

- Clientes invocan servicios proporcionados por un servidor.

- La interacción se hace a través de un protocolo basado en un esquema de request/reply (por ejemplo HTTP)

- El cliente requiere conocer el servidor y el servicio a invocar; los servidores no conocen los clientes.

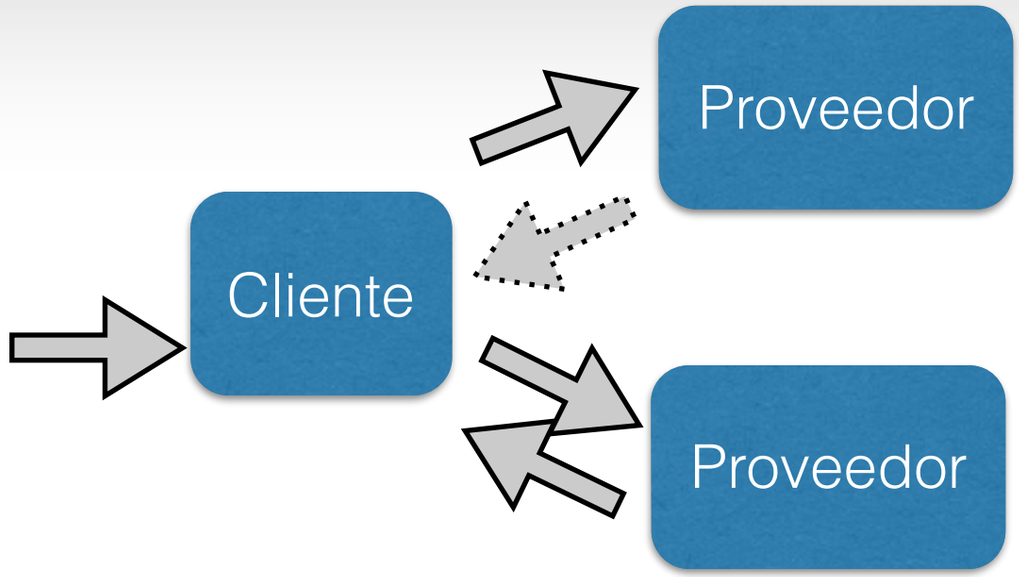


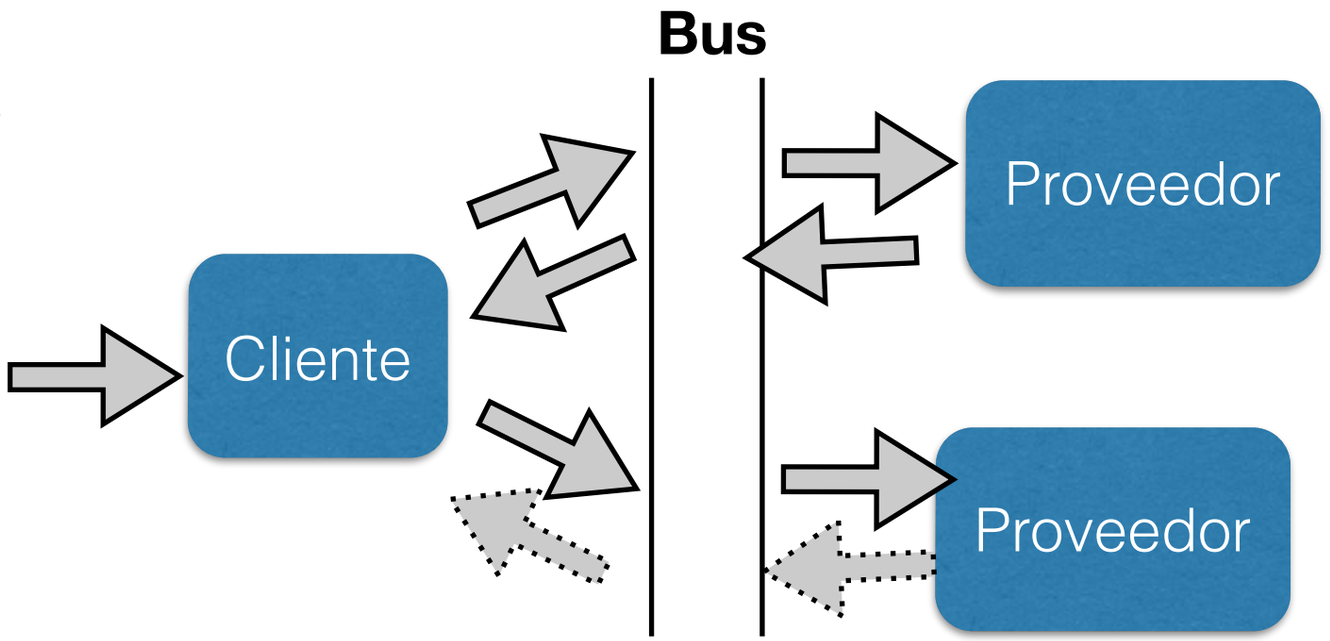
Call return: SOA

- Clientes invocan servicios proporcionados por los servidores, en un ambiente heterogéneo.

- La interacción se hace mediante invocación directa o mediante el uso de un bus empresarial (ESB).

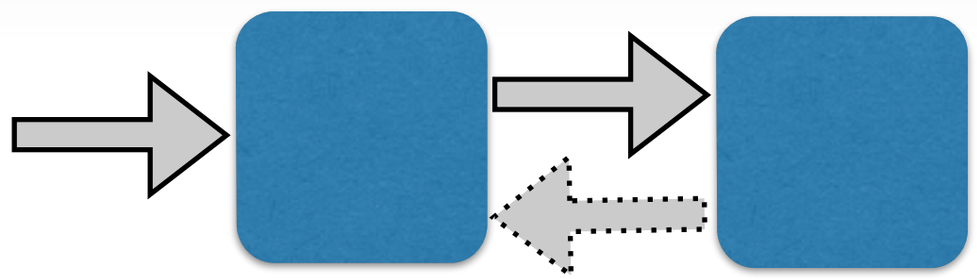
- Es event-based porque puede usar motores de orquestación/ coreografía e invocar servicios de forma asíncrona



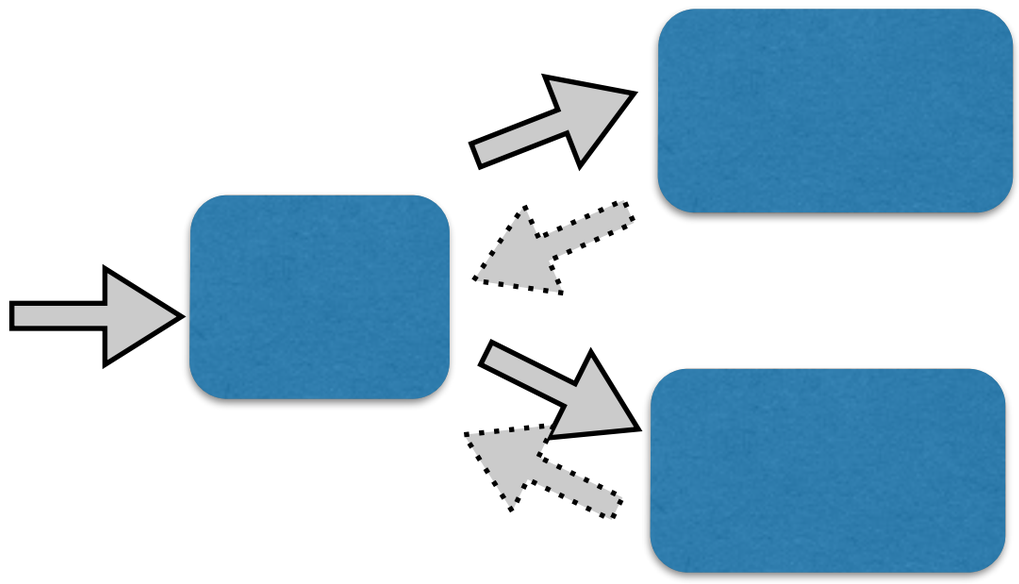


Event based

- Se caracteriza por una federación bajamente acoplada de componentes que disparan comportamientos en otros a través de eventos.

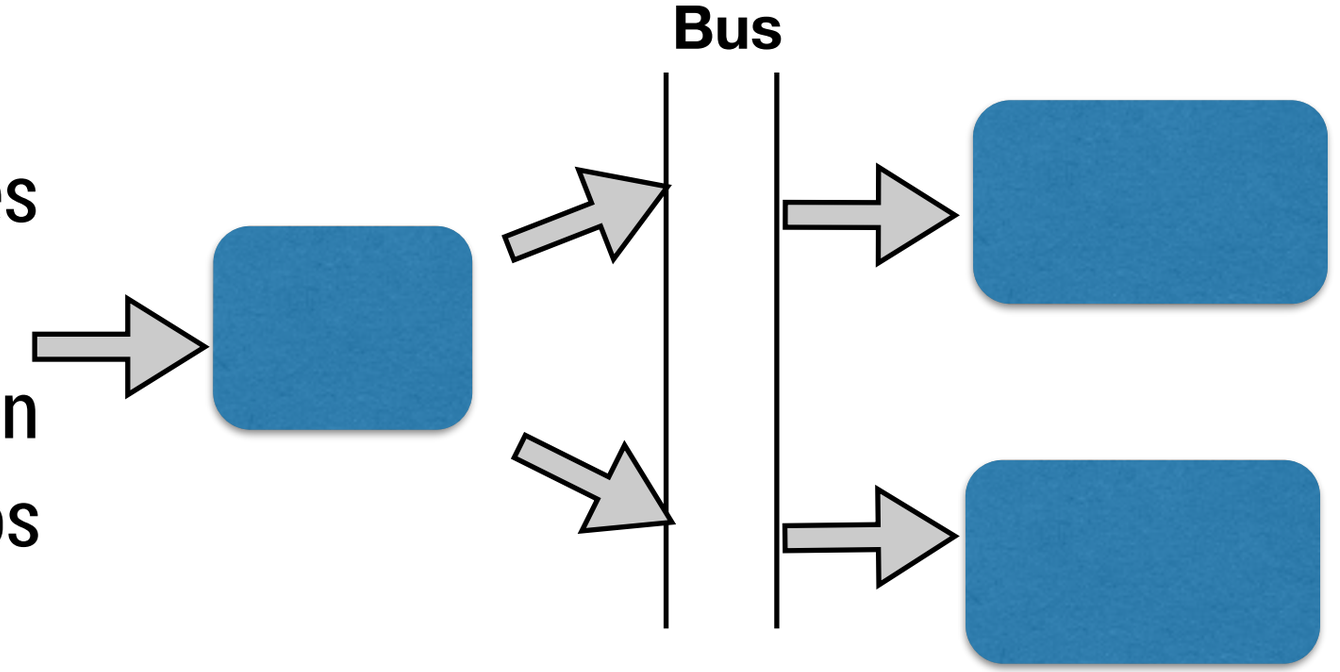


- Pueden ser punto a punto u operar en un modelo de publicación-suscripción.



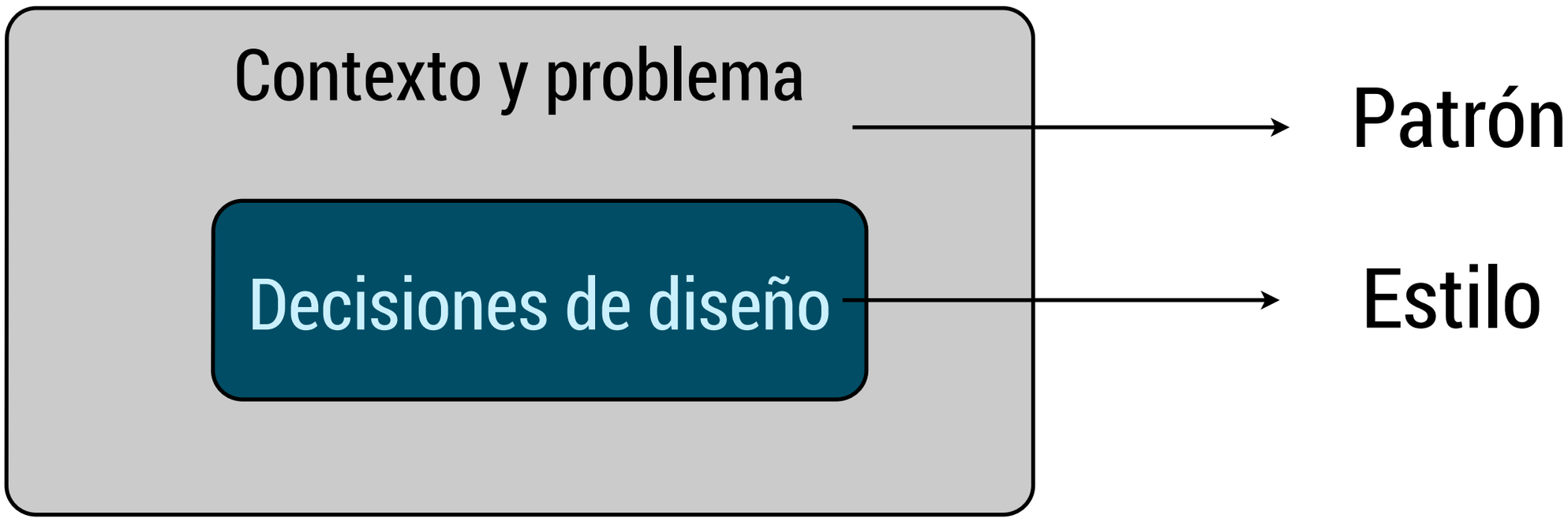
Event based: Publish-suscribe

- Los conectores se suscriben a un conjunto de eventos
- El conector entre los componentes es un bus de eventos: los componentes anuncian eventos en el bus, y el bus entrega los eventos a los componentes respectivos (suscriptores)



Patrón vs Estilo

El patrón incluye la descripción del contexto y problema a solucionar



Model View Controller (MVC)

The original MVC reports

Trygve Reenskaug
Dept. of Informatics
University of Oslo

I made the first implementation and wrote the original MVC reports while I was a visiting scientist at Xerox Palo Alto Research Laboratory (PARC) in 1978/79. MVC was created as an obvious solution to the general problem of giving users control over their information as seen from multiple perspectives. MVC has created a surprising amount of interest. Some texts even use perverted variants for the opposite purpose of making the computer control the user.

MVC was conceived as a general solution to the problem of users controlling a large and complex data set. The hardest part was to hit upon good names for the different architectural components. Model-View-Editor was the first set. They are described in my first note of May 12, 1979: *THING-MODEL-VIEW-EDITOR -.an Example from a planningsystem*. A scanned version is included below.

After long discussions, particularly with Adele Goldberg, I ended with the terms Model-View-Controller as described in my second note of December 10, 1979: *MODELS - VIEWS – CONTROLLERS*. A scanned version is included at the end of this document.

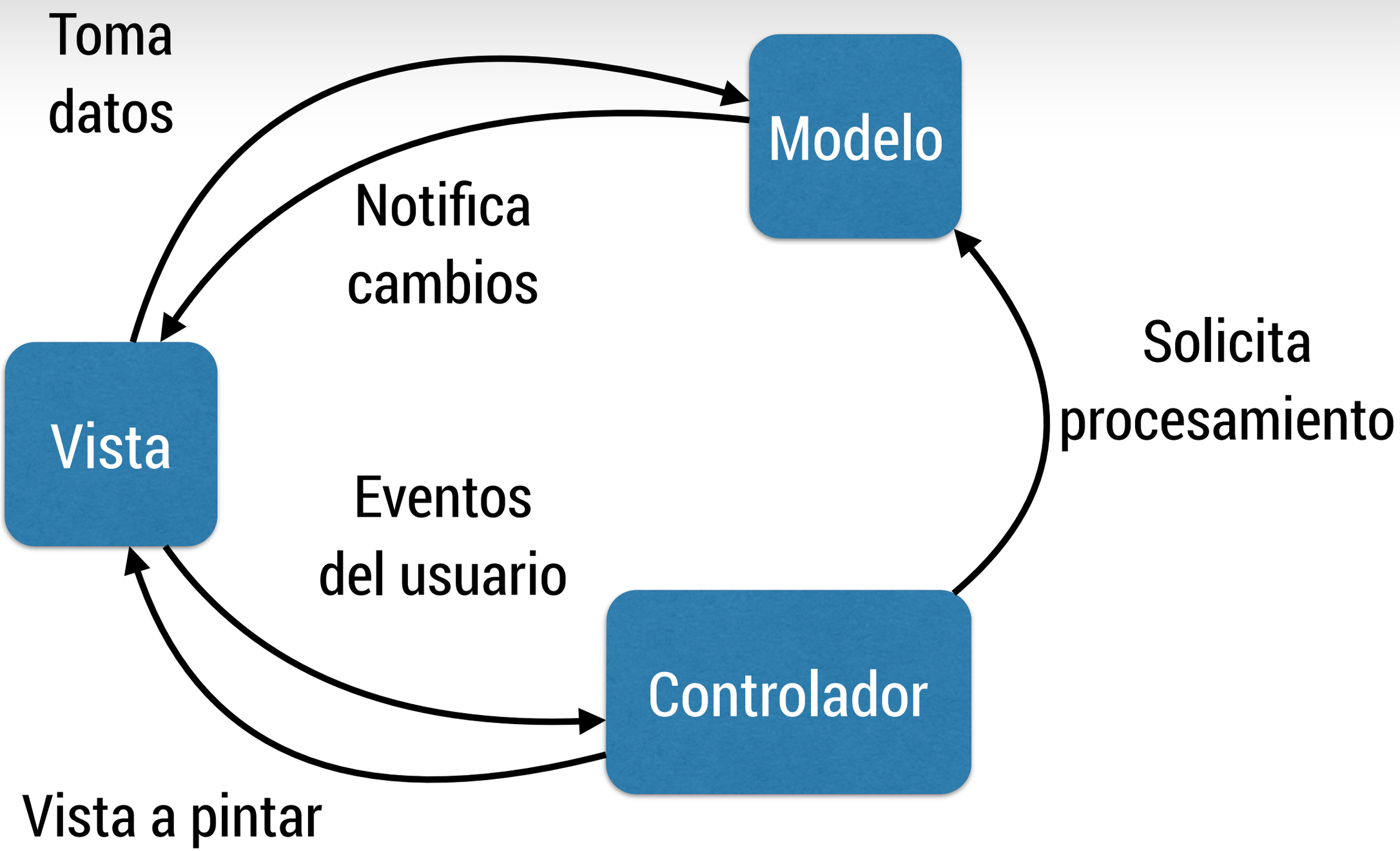
Jim Althoff and others implemented a version of MVC for the Smalltalk-80 class library after I had left Xerox PARC; I was in not involved in this work.

Oslo, February 12, 2007
Trygve Reenskaug

http://heim.ifi.uio.no/~trygver/themes/mvc/mvc-index.html

http://heim.ifi.uio.no/~trygver/2007/MVC_Originals.pdf

Model View Controller (MVC)



Model View Controller (MVC)

Controlador: acepta solicitudes del usuario, invoca procesamiento en los componentes del modelo, y determina cual vista debe ser desplegada. Define el flujo de la aplicación. Realiza el mapeo entre las acciones del usuario y la modificación del modelo.

Vista: envía eventos al controlador, y pinta los datos contenidos en los componentes del modelo.

Modelo: contiene los datos, reglas y lógica de negocio. No debe tener ningún detalle acerca de la interfaz gráfica.

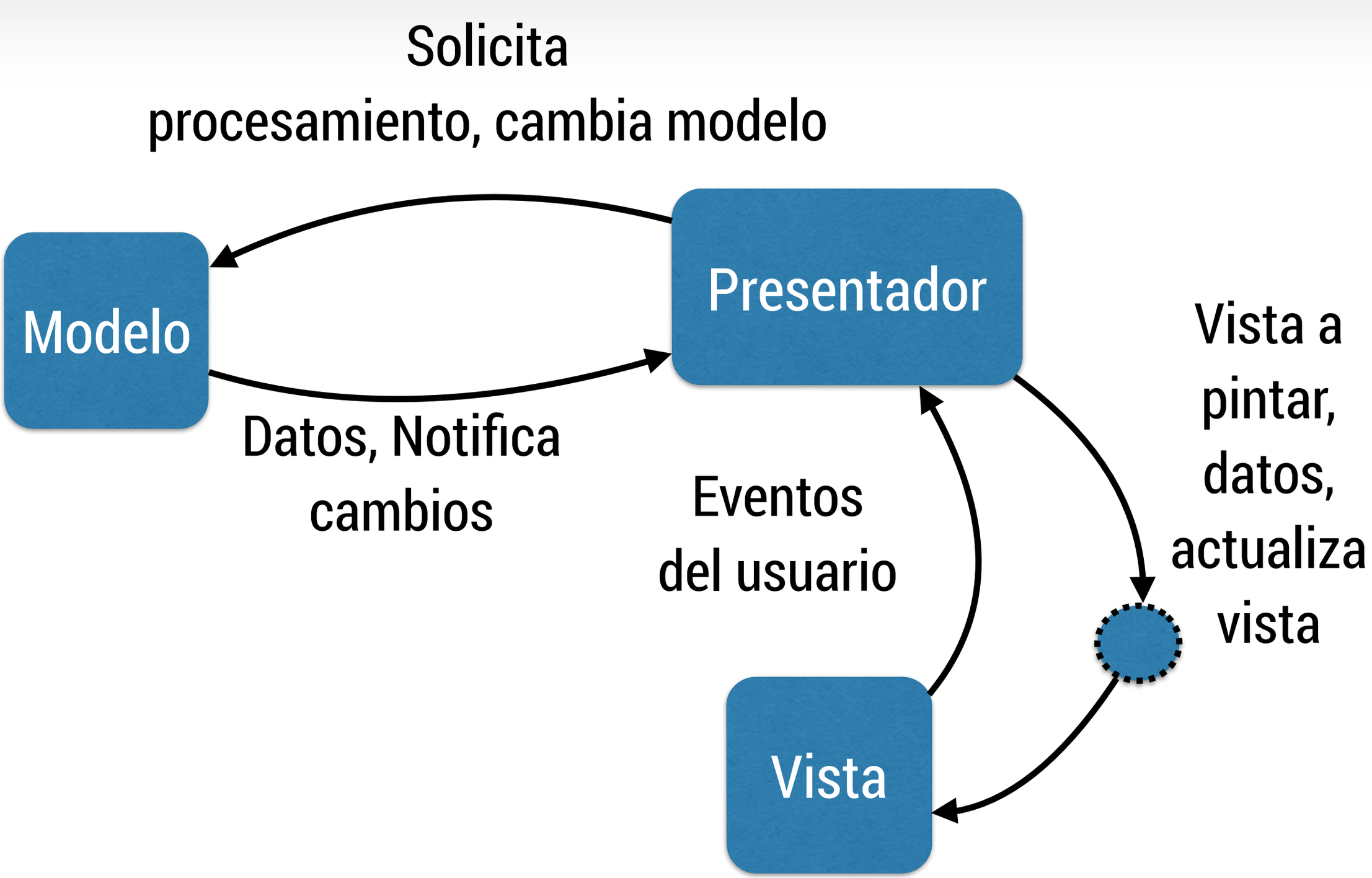
Model View Presenter (MVP)

MVP: Model-View-Presenter
The Taligent Programming Model for C++ and Java

Mike Potel
VP & CTO
Taligent, Inc.

Taligent, a wholly-owned subsidiary of IBM, is developing a next generation programming model for the C++ and Java programming languages, called Model-View-Presenter or MVP, based on a generalization of the classic MVC programming model of Smalltalk. MVP provides a powerful yet easy to understand design methodology for a broad range of application and component development tasks. Taligent's framework-based implementation of these concepts adds great value to developer programs that employ MVP. MVP also is adaptable across multiple client/server and multi-tier application architectures. MVP will enable IBM to deliver a unified conceptual programming model across all its major object-oriented language environments.

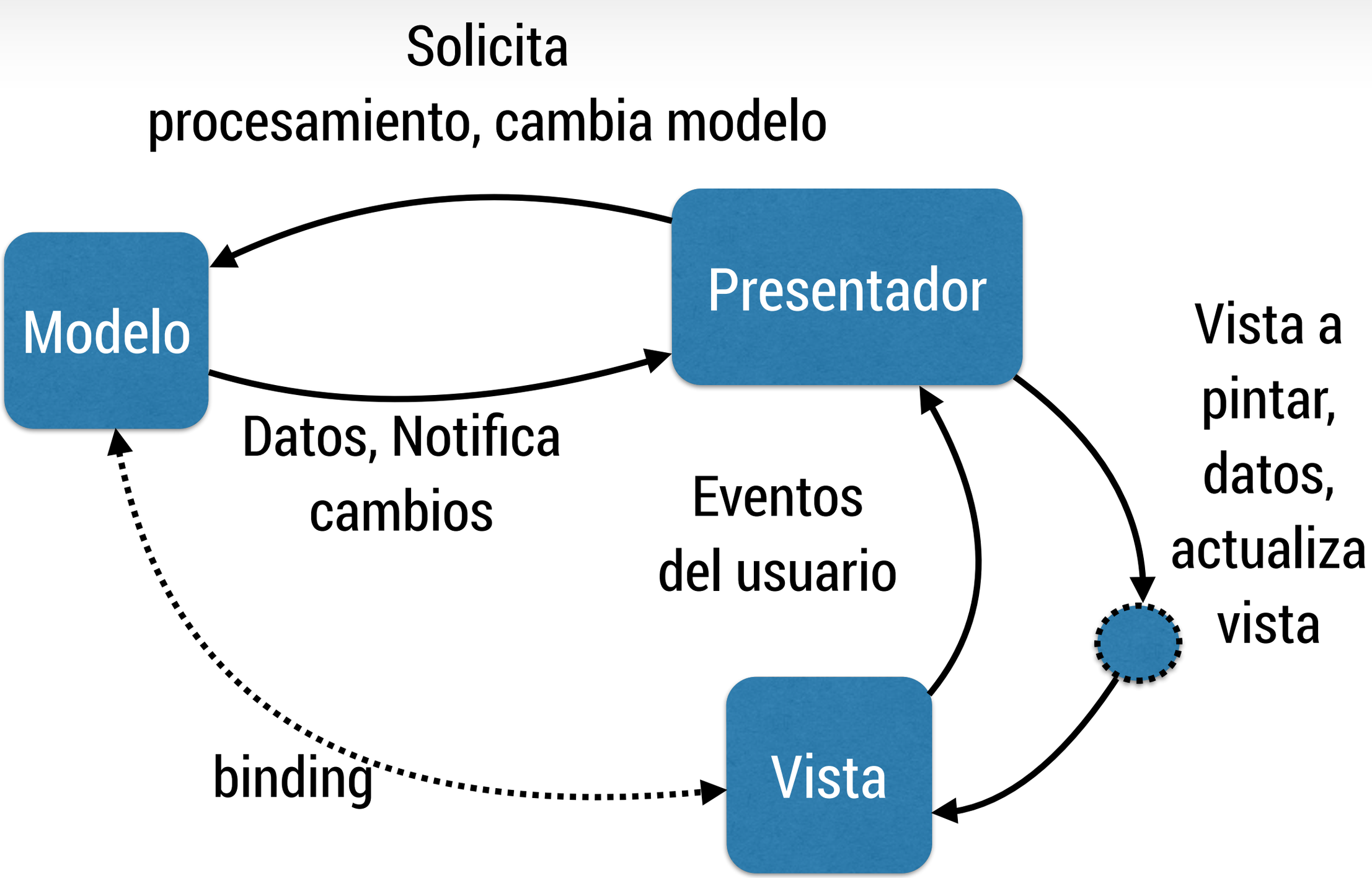
Model View Presenter 1 (MVP - Passive view)



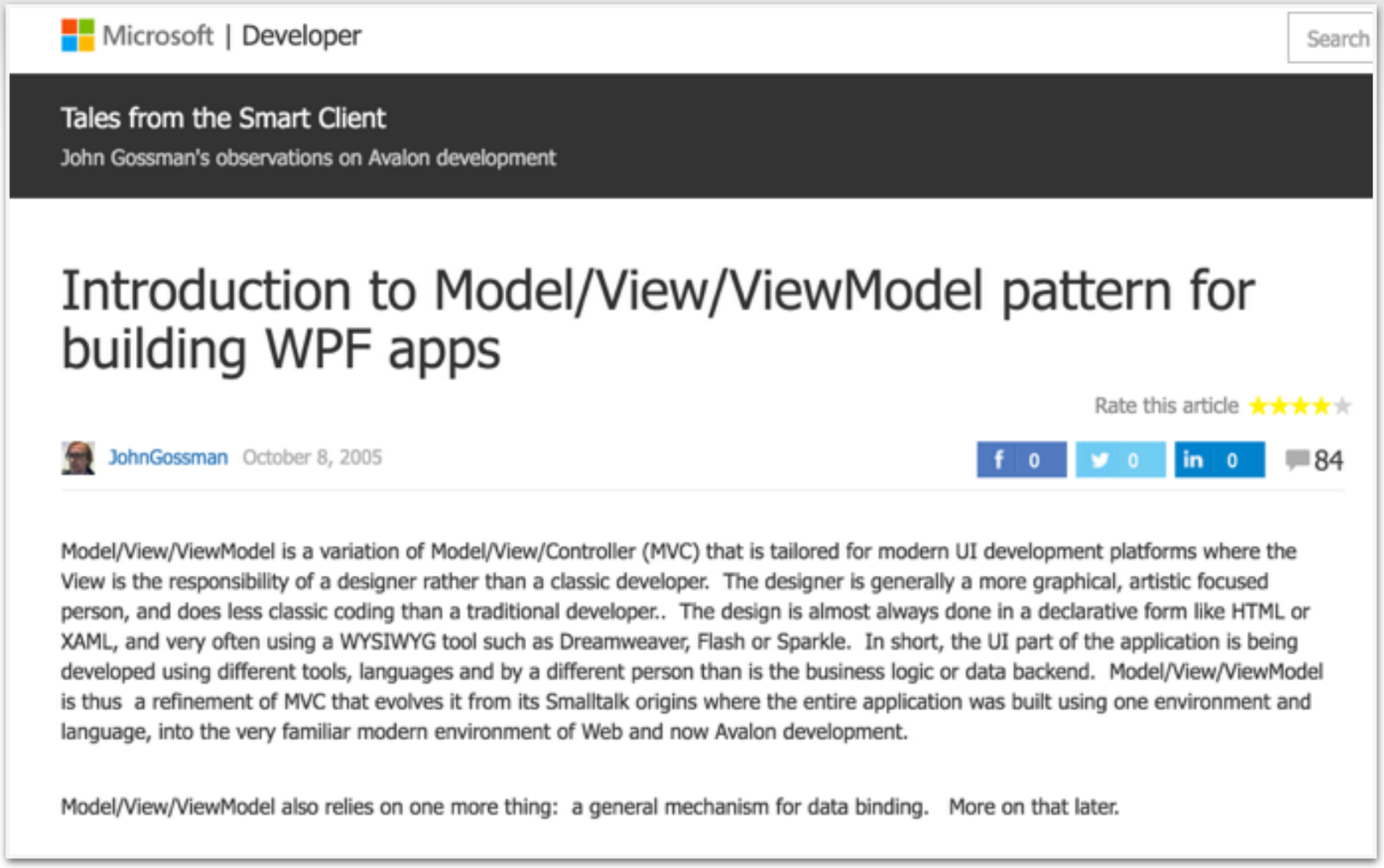
Model View Presenter 1 (MVP - Passive view)

Interfaz de la vista, útil para mock testing

Model View Presenter 2 (MVP - Supervising Controller)



Model View ViewModel (MVVM)



Microsoft | Developer

Search

Tales from the Smart Client

John Gossman's observations on Avalon development

Introduction to Model/View/ViewModel pattern for building WPF apps

Rate this article ★★★★☆

JohnGossman October 8, 2005

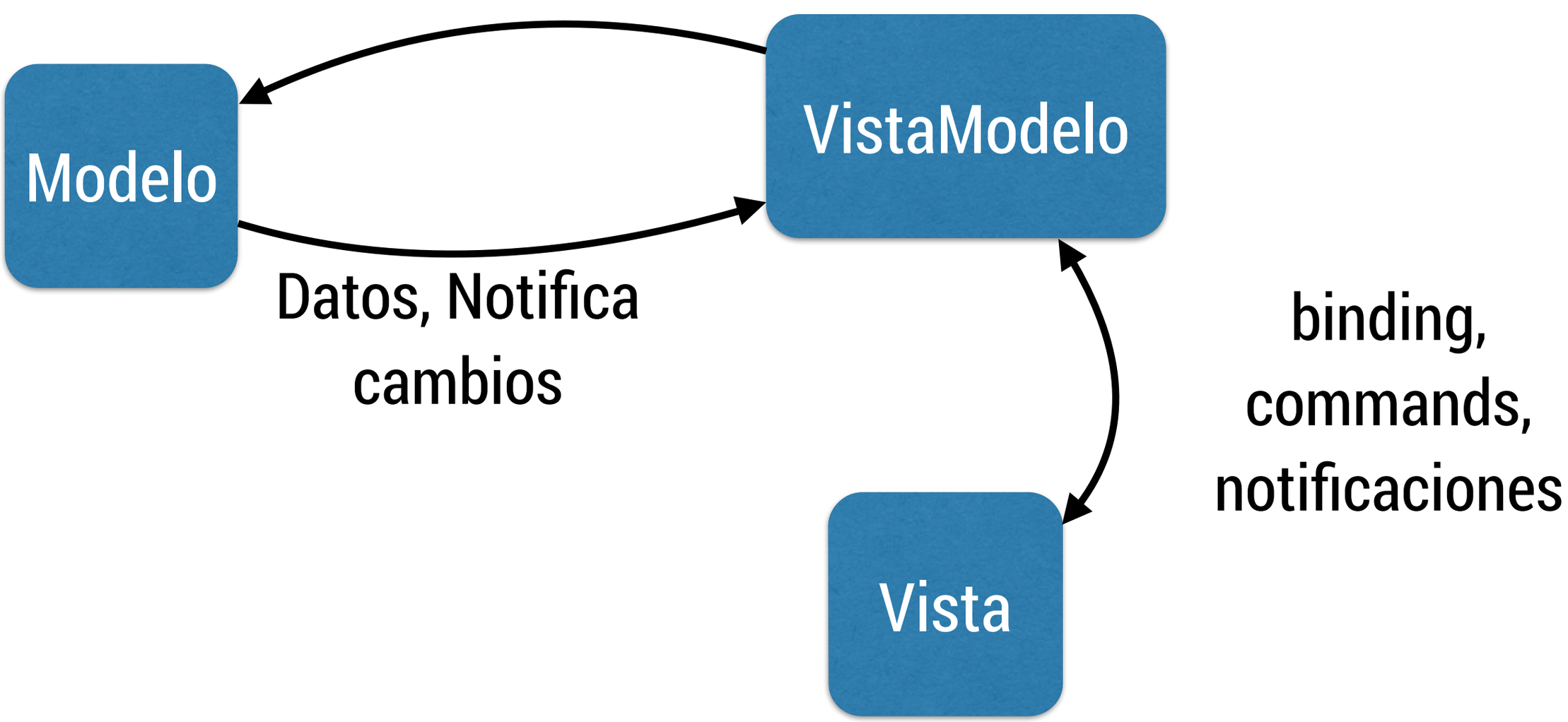
f 0 0 in 0 84

Model/View/ViewModel is a variation of Model/View/Controller (MVC) that is tailored for modern UI development platforms where the View is the responsibility of a designer rather than a classic developer. The designer is generally a more graphical, artistic focused person, and does less classic coding than a traditional developer.. The design is almost always done in a declarative form like HTML or XAML, and very often using a WYSIWYG tool such as Dreamweaver, Flash or Sparkle. In short, the UI part of the application is being developed using different tools, languages and by a different person than is the business logic or data backend. Model/View/ViewModel is thus a refinement of MVC that evolves it from its Smalltalk origins where the entire application was built using one environment and language, into the very familiar modern environment of Web and now Avalon development.

Model/View/ViewModel also relies on one more thing: a general mechanism for data binding. More on that later.

https://blogs.msdn.microsoft.com/johngossman/2005/10/08/introduction-to-modelviewviewmodel-pattern-for-building-wpf-apps/

Model View ViewModel (MVVM)



Model View Whatever (MV*, MVW)

AngularJS ▸ Public Jul 19, 2012

Originally shared by Igor Minar

MVC vs MVVM vs MVP. What a controversial topic that many developers can spend hours and hours debating and arguing about.

For several years **+AngularJS was closer to MVC** (or rather one of its client-side variants), but over time and thanks to many refactorings and api improvements, it's **now closer to MVVM** – the $scope object could be considered the ViewModel that is being decorated by a function that we call a Controller.

Being able to categorize a framework and put it into one of the MV* buckets has some advantages. It can help developers get more comfortable with its apis by making it easier to create a mental model that represents the application that is being built with the framework. It can also help to establish terminology that is used by developers.

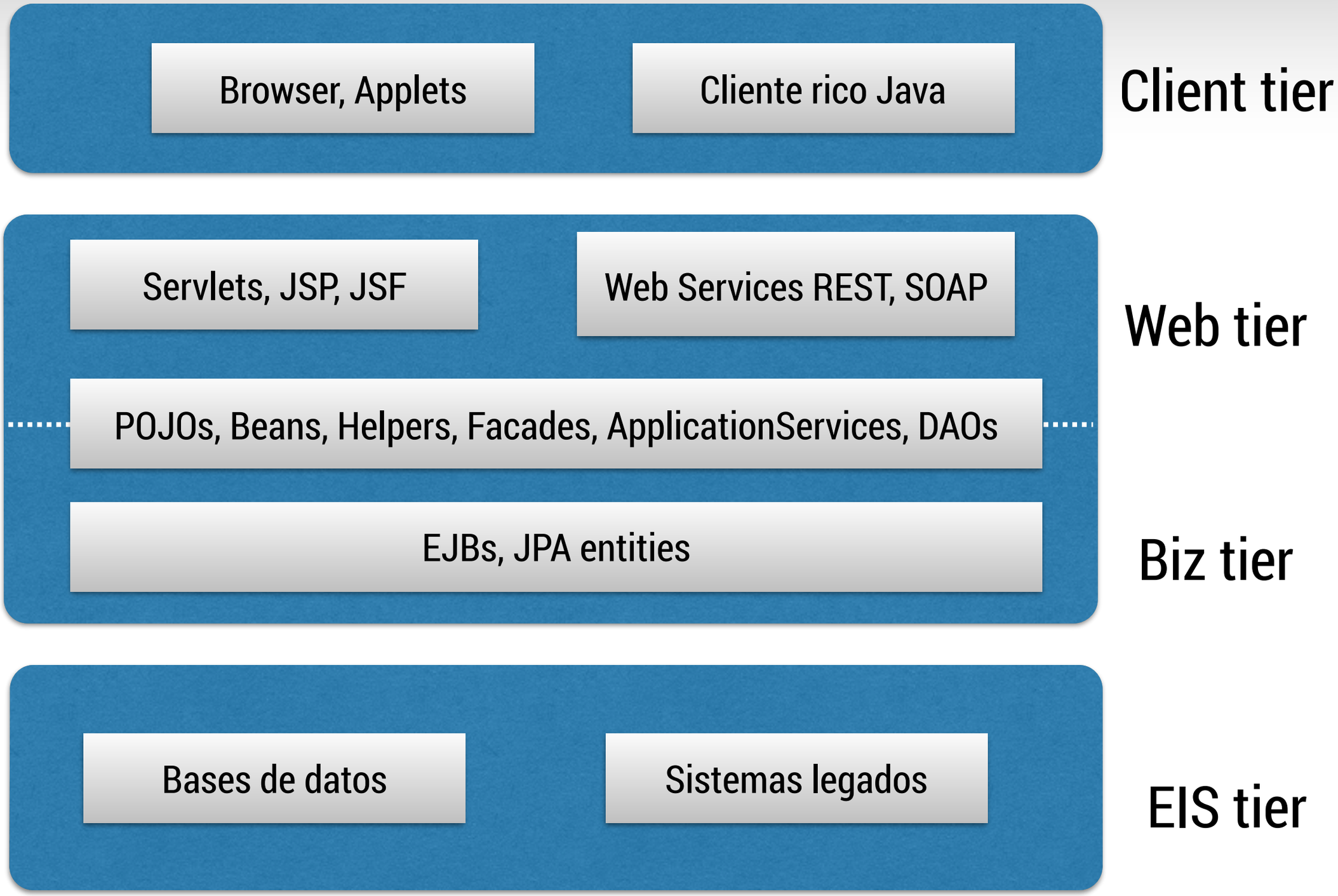
Having said, **I'd rather see developers build kick-ass apps** that are well-designed and follow separation of concerns, than see them waste time arguing about MV* nonsense. And for this reason, **I hereby declare AngularJS to be MVW framework - Model-View-Whatever**. Where Whatever stands for "whatever works for you".

Angular gives you a lot of flexibility to nicely separate presentation logic from business logic and presentation state. Please use it fuel your productivity and application maintainability rather than heated discussions about things that at the end of the day don't matter that much.

https://plus.google.com/+AngularJS/posts/aZNVhj355G2

EN LOS PATRONES MVC, MVP, MVVM, DONDE SE LOCALIZAN LA LOGICA DE NEGOCIO Y LA LOGICA DE PRESENTACION ?

Arquitectura JEE (multi-tier)



LA ARQUITECTURA JEE ES MVC, MVP O MVVM ?