[1 Arquitectura](http://es.wikipedia.org/wiki/Arquitectura_de_software" \l "Arquitectura)

[3 Modelos o vistas](http://es.wikipedia.org/wiki/Arquitectura_de_software#Modelos_o_vistas)

[4 Arquitecturas más comunes](http://es.wikipedia.org/wiki/Arquitectura_de_software#Arquitecturas_m.C3.A1s_comunes)

**Arquitectura de software**

Se le llama arquitectura por la semejanza a los planos de un edificio en construcción, estas indican la estructura funcionamiento e interacción entre las partes del software. En el libro "An introduction to Software Architecture", David Garlan y Mary Shaw definen que la Arquitectura es un nivel de diseño que hace foco en aspectos "más allá de los algoritmos y estructuras de datos de la computación; el diseño y especificación de la estructura global del sistema es un nuevo tipo de problema".

**Arquitectura**

* La Arquitectura del Software es el diseño de más alto nivel de la estructura de un sistema.
* Una Arquitectura de [Software](http://es.wikipedia.org/wiki/Software), también denominada *Arquitectura* [*lógica*](http://es.wikipedia.org/wiki/L%C3%B3gica), consiste en un conjunto de patrones y abstracciones coherentes que proporcionan el marco
* Una arquitectura de software se selecciona y diseña con base en objetivos y restricciones. Los objetivos son aquellos prefijados para el sistema de información, pero no solamente los de tipo funcional, también otros objetivos como la mantenibilidad, auditabilidad, flexibilidad e interacción con otros sistemas de información. Las restricciones son aquellas limitaciones derivadas de las tecnologías disponibles para implementar sistemas de información. Unas arquitecturas son más recomendables de implementar con ciertas tecnologías mientras que otras tecnologías no son aptas para determinadas arquitecturas. Por ejemplo, no es viable emplear una arquitectura de software de tres capas para implementar sistemas en tiempo real.
* La arquitectura de software define, de manera abstracta, los componentes que llevan a cabo alguna tarea de computación, sus interfaces y la comunicación entre ellos. Toda arquitectura debe ser implementable en una [arquitectura](http://es.wikipedia.org/wiki/Arquitectura) física, que consiste simplemente en determinar qué [computadora](http://es.wikipedia.org/wiki/Computadora) tendrá asignada cada tarea.

La arquitectura de software, tiene que ver con el diseño y la implementación de estructuras de software de alto nivel. Es el resultado de ensamblar un cierto número de elementos arquitectónicos de forma adecuada para satisfacer la mayor funcionalidad y requerimientos de desempeño de un sistema, así como requerimientos no funcionales, como la [confiabilidad](http://es.wikipedia.org/wiki/Confiabilidad), [escalabilidad](http://es.wikipedia.org/wiki/Escalabilidad), [portabilidad](http://es.wikipedia.org/wiki/Portabilidad), y disponibilidad.

Kruchten, Philippe

**Modelos o vistas**

Toda arquitectura de software debe describir diversos aspectos del software. Generalmente, cada uno de estos aspectos se describe de una manera más comprensible si se utilizan distintos modelos o vistas. Es importante destacar que cada uno de ellos constituye una descripción parcial de una misma arquitectura y es deseable que exista cierto solapamiento entre ellos. Esto es así porque todas las vistas deben ser coherentes entre sí, evidente dado que describen la misma cosa.

Cada [paradigma](http://es.wikipedia.org/wiki/Paradigma) de desarrollo exige diferente número y tipo de vistas o modelos para describir una arquitectura. No obstante, existen al menos tres vistas absolutamente fundamentales en cualquier arquitectura:

* La visión **estática**: describe qué componentes tiene la arquitectura.
* La visión **funcional**: describe qué hace cada componente.
* La visión **dinámica**: describe cómo se comportan los componentes a lo largo del tiempo y como interactúan entre sí.

Las vistas o modelos de una arquitectura de software pueden expresarse mediante uno o varios [lenguajes](http://es.wikipedia.org/wiki/Lenguaje). El más obvio es el [lenguaje natural](http://es.wikipedia.org/wiki/Lenguaje_natural), pero existen otros lenguajes tales como los [diagramas de estado](http://es.wikipedia.org/wiki/Diagrama_de_estado), los [diagramas de flujo de datos](http://es.wikipedia.org/wiki/Diagrama_de_flujo_de_datos), etc. Estos lenguajes son apropiados únicamente para un modelo o vista. Afortunadamente existe cierto consenso en adoptar [UML](http://es.wikipedia.org/wiki/UML) (*Unified Modeling Language*, lenguaje unificado de modelado) como lenguaje único para todos los modelos o vistas. Sin embargo, un lenguaje generalista corre el peligro de no ser capaz de describir determinadas restricciones de un sistema de información (o expresarlas de manera incomprensible).

**Arquitecturas más comunes**

Generalmente, no es necesario inventar una nueva arquitectura de software para cada sistema de información. Lo habitual es adoptar una arquitectura conocida en función de sus ventajas e inconvenientes para cada caso en concreto. Así, las arquitecturas más universales son:

[Monolítica](http://es.wikipedia.org/w/index.php?title=Arquitectura_monol%C3%ADtica_%28inform%C3%A1tica%29&action=edit&redlink=1). Donde el software se estructura en grupos funcionales muy acoplados.

[Cliente-servidor](http://es.wikipedia.org/wiki/Cliente-servidor). Donde el software reparte su carga de cómputo en dos partes independientes pero sin reparto claro de funciones.

[Arquitectura de tres niveles](http://es.wikipedia.org/wiki/Arquitectura_de_tres_niveles). Especialización de la arquitectura cliente-servidor donde la carga se divide en tres capas con un reparto claro de funciones:

1. Para la interfaz de usuario,
2. Para el cálculo es donde se encuentra modelado el negocio
3. Para el almacenamiento.