En el ámbito del software cada vez es más común escuchar el término “arquitectura de software”, y encontrar oportunidades de empleo para “arquitectos de software”. Aún así, este concepto tiende a ser malentendido y la falta de comprensión al respecto de sus principios frecuentemente repercute de manera negativa en la construcción de sistemas de software.

El concepto de arquitectura de software se refiere a la estructuración del sistema que, idealmente, se crea en etapas tempranas del desarrollo. Esta estructuración representa un diseño de alto nivel del sistema que tiene dos propósitos primarios: satisfacer los atributos de calidad (desempeño, seguridad, modificabilidad), y servir como guía en el desarrollo. Al igual que en la ingeniería civil, las decisiones críticas relativas al diseño general de un sistema de software complejo deben de hacerse desde un principio. El no crear este diseño desde etapas tempranas del desarrollo puede limitar severamente el que el producto final satisfaga las necesidades de los clientes. Además, el costo de las correcciones relacionadas con problemas en la arquitectura es muy elevado. Es así que la arquitectura de software juega un papel fundamental dentro del desarrollo.

¿Qué es la arquitectura de software?

Antes de elaborar sobre el tema, es conveniente definir el concepto ya que hoy en día el término de arquitectura se usa para referirse a varios aspectos relacionados con las TI. De acuerdo al Software Engineering Institute (SEI), la Arquitectura de Software se refiere a “las estructuras de un sistema, compuestas de elementos con propiedades visibles de forma externa y las relaciones que existen entre ellos.”[1]

El término “elementos” dentro de la definición del SEI es vago a propósito, pues puede referirse a distintas entidades relacionadas con el sistema. Los elementos pueden ser entidades que existen en tiempo de ejecución (objetos, hilos), entidades lógicas que existen en tiempo de desarrollo (clases, componentes) y entidades físicas (nodos, directorios). Por otro lado, las relaciones entre elementos dependen de propiedades visibles (o públicas) de los elementos, quedando ocultos los detalles de implementación. Finalmente, cada conjunto de elementos relacionados de un tipo particular corresponde a una estructura distinta, de ahí que la arquitectura esta compuesta por distintas estructuras.

¿Por qué es importante la arquitectura de software?

La arquitectura de software es de especial importancia ya que la manera en que se estructura un sistema tiene un impacto directo sobre la capacidad de este para satisfacer lo que se conoce como los atributos de calidad del sistema. Ejemplos de atributos de calidad son el desempeño, que tiene que ver con el tiempo de respuesta del sistema a las peticiones que se le hacen, la usabilidad, que tiene que ver con qué tan sencillo les resulta a los usuarios realizar operaciones con el sistema, o bien la modificabilidad, que tiene que ver con qué tan simple resulta introducir cambios en el sistema. Los atributos de calidad son parte de los requerimientos (no funcionales) del sistema y son características que deben expresarse de forma cuantitativa. No tiene sentido, por ejemplo, decir que el sistema debe devolver una petición “de manera rápida”, o presentar una página “ligera”, ya que no es posible evaluar objetivamente si el sistema cubre o no esos requerimientos.

La manera en que se estructura un sistema permitirá o impedirá que se satisfagan los atributos de calidad. Por ejemplo, un sistema estructurado de tal manera que una petición deba transitar por muchos componentes antes de que se devuelva una respuesta podría tener un desempeño pobre. Por otro lado, un sistema estructurado de tal manera que los componentes estén altamente acoplados entre ellos limitará severamente la modificabilidad. Curiosamente, la estructuración tiene un impacto mucho menor respecto a los requerimientos funcionales del sistema. Por ejemplo, un sistema difícil de modificar puede satisfacer plenamente los requerimientos funcionales que se le imponen.

Además de los atributos de calidad, la arquitectura de software juega un papel fundamental para guiar el desarrollo. Una de las múltiples estructuras que la componen se enfoca en partir el sistema en componentes que serán desarrollados por individuos o grupos de individuos. La identificación de esta estructura de asignación de trabajo es esencial para apoyar las tareas de planeación del proyecto.

Finalmente, los diseños arquitectónicos que se crean en una organización pueden ser reutilizados para crear sistemas distintos. Esto permite reducir costos y aumentar la calidad, sobre todo si dichos diseños han resultado previamente en sistemas exitosos.

El ciclo de desarrollo de la arquitectura

Dentro de un proyecto de desarrollo, e independientemente de la metodología que se utilice, se puede hablar de “desarrollo de la arquitectura de software”. Este desarrollo, que precede a la construcción del sistema, esta dividido en las siguientes etapas: requerimientos, diseño, documentación y evaluación. Cabe señalar que las actividades relacionadas con el desarrollo de la arquitectura de software generalmente forman parte de las actividades definidas dentro de las metodologías de desarrollo.

A continuación se describen dichas etapas.

**Requerimientos**. La etapa de requerimientos se enfoca en la captura, documentación y priorización de requerimientos que influencian la arquitectura. Como se mencionó anteriormente, los atributos de calidad juegan un papel preponderante dentro de estos requerimientos, así que esta etapa hace énfasis en ellos. Otros requerimientos, sin embargo, son también relevantes para la arquitectura, estos son los requerimientos funcionales primarios y las restricciones.

**Diseño**. La etapa de diseño es la etapa central en relación con la arquitectura y probablemente la más compleja. Durante esta etapa se definen las estructuras que componen la arquitectura. La creación de estas estructuras se hace en base a patrones de diseño, tácticas de diseño y elecciones tecnológicas. El diseño que se realiza debe buscar ante todo satisfacer los requerimientos que influencian a la arquitectura, y no simplemente incorporar diversas tecnologías por que están “de moda”.

**Documentación.** Una vez creado el diseño de la arquitectura, es necesario poder comunicarlo a otros involucrados dentro del desarrollo. La comunicación exitosa del diseño muchas veces depende de que dicho diseño sea documentado de forma apropiada. La documentación de una arquitectura involucra la representación de varias de sus estructuras que son representadas a través de distintas vistas. Una vista generalmente contiene un diagrama, además de información adicional, que apoya en la comprensión de dicho diagrama.

**Evaluación.** Dado que la arquitectura de software juega un papel crítico en el desarrollo, es conveniente evaluar el diseño una vez que este ha sido documentado con el fin de identificar posibles problemas y riesgos. La ventaja de evaluar el diseño es que es una actividad que se puede realizar de manera temprana (aún antes de codificar), y que el costo de corrección de los defectos identificados a través de la evaluación es mucho menor al costo que tendría el corregir estos defectos una vez que el sistema ha sido construido.

El rol de arquitecto

Las actividades descritas anteriormente requieren de habilidades particulares que son la responsabilidad del arquitecto de software. El arquitecto es un líder técnico que debe conocer los principios relacionados con la arquitectura de software, tener un amplio conocimiento respecto a la tecnología, y tener excelentes habilidades de comunicación escrita y oral.

Desafortunadamente, en la actualidad pocos arquitectos de software que laboran en la industria han recibido una formación teórica respecto al tema. Esto se debe a que no es sino hasta épocas recientes que se han establecido de manera más formal los conceptos relacionados con la arquitectura de software, y que actualmente pocas instituciones ofrecen cursos enfocados en el tema. El desconocimiento de los principios relativos a la arquitectura de software frecuentemente impacta de manera negativa a los proyectos de desarrollo.

Apenas empezamos

A lo largo de las distintas entregas de esta columna que inicia se buscará dar una panorámica del tema de arquitectura de software y se discutirá de manera más detallada aspectos como:

* requerimientos que influyen en la arquitectura,
* el diseño, documentación y evaluación de la arquitectura,
* los retos relacionados con la introducción de prácticas de arquitectura de software en un contexto organizacional,
* la arquitectura de software dentro de las metodologías de desarrollo,
* el perfil del arquitecto de software.

En el ámbito del software cada vez es más común escuchar el término “arquitectura de software”, y encontrar oportunidades de empleo para “arquitectos de software”. Aún así, este concepto tiende a ser malentendido y la falta de comprensión al respecto de sus principios frecuentemente repercute de manera negativa en la construcción de sistemas de software.

El concepto de arquitectura de software se refiere a la estructuración del sistema que, idealmente, se crea en etapas tempranas del desarrollo. Esta estructuración representa un diseño de alto nivel del sistema que tiene dos propósitos primarios: satisfacer los atributos de calidad (desempeño, seguridad, modificabilidad), y servir como guía en el desarrollo. Al igual que en la ingeniería civil, las decisiones críticas relativas al diseño general de un sistema de software complejo deben de hacerse desde un principio. El no crear este diseño desde etapas tempranas del desarrollo puede limitar severamente el que el producto final satisfaga las necesidades de los clientes. Además, el costo de las correcciones relacionadas con problemas en la arquitectura es muy elevado. Es así que la arquitectura de software juega un papel fundamental dentro del desarrollo.

¿Qué es la arquitectura de software?

Antes de elaborar sobre el tema, es conveniente definir el concepto ya que hoy en día el término de arquitectura se usa para referirse a varios aspectos relacionados con las TI. De acuerdo al Software Engineering Institute (SEI), la Arquitectura de Software se refiere a “las estructuras de un sistema, compuestas de elementos con propiedades visibles de forma externa y las relaciones que existen entre ellos.”[1]

El término “elementos” dentro de la definición del SEI es vago a propósito, pues puede referirse a distintas entidades relacionadas con el sistema. Los elementos pueden ser entidades que existen en tiempo de ejecución (objetos, hilos), entidades lógicas que existen en tiempo de desarrollo (clases, componentes) y entidades físicas (nodos, directorios). Por otro lado, las relaciones entre elementos dependen de propiedades visibles (o públicas) de los elementos, quedando ocultos los detalles de implementación. Finalmente, cada conjunto de elementos relacionados de un tipo particular corresponde a una estructura distinta, de ahí que la arquitectura esta compuesta por distintas estructuras.

¿Por qué es importante la arquitectura de software?

La arquitectura de software es de especial importancia ya que la manera en que se estructura un sistema tiene un impacto directo sobre la capacidad de este para satisfacer lo que se conoce como los atributos de calidad del sistema. Ejemplos de atributos de calidad son el desempeño, que tiene que ver con el tiempo de respuesta del sistema a las peticiones que se le hacen, la usabilidad, que tiene que ver con qué tan sencillo les resulta a los usuarios realizar operaciones con el sistema, o bien la modificabilidad, que tiene que ver con qué tan simple resulta introducir cambios en el sistema. Los atributos de calidad son parte de los requerimientos (no funcionales) del sistema y son características que deben expresarse de forma cuantitativa. No tiene sentido, por ejemplo, decir que el sistema debe devolver una petición “de manera rápida”, o presentar una página “ligera”, ya que no es posible evaluar objetivamente si el sistema cubre o no esos requerimientos.

La manera en que se estructura un sistema permitirá o impedirá que se satisfagan los atributos de calidad. Por ejemplo, un sistema estructurado de tal manera que una petición deba transitar por muchos componentes antes de que se devuelva una respuesta podría tener un desempeño pobre. Por otro lado, un sistema estructurado de tal manera que los componentes estén altamente acoplados entre ellos limitará severamente la modificabilidad. Curiosamente, la estructuración tiene un impacto mucho menor respecto a los requerimientos funcionales del sistema. Por ejemplo, un sistema difícil de modificar puede satisfacer plenamente los requerimientos funcionales que se le imponen.

Además de los atributos de calidad, la arquitectura de software juega un papel fundamental para guiar el desarrollo. Una de las múltiples estructuras que la componen se enfoca en partir el sistema en componentes que serán desarrollados por individuos o grupos de individuos. La identificación de esta estructura de asignación de trabajo es esencial para apoyar las tareas de planeación del proyecto.

Finalmente, los diseños arquitectónicos que se crean en una organización pueden ser reutilizados para crear sistemas distintos. Esto permite reducir costos y aumentar la calidad, sobre todo si dichos diseños han resultado previamente en sistemas exitosos.

El ciclo de desarrollo de la arquitectura

Dentro de un proyecto de desarrollo, e independientemente de la metodología que se utilice, se puede hablar de “desarrollo de la arquitectura de software”. Este desarrollo, que precede a la construcción del sistema, esta dividido en las siguientes etapas: requerimientos, diseño, documentación y evaluación. Cabe señalar que las actividades relacionadas con el desarrollo de la arquitectura de software generalmente forman parte de las actividades definidas dentro de las metodologías de desarrollo.

A continuación se describen dichas etapas.

**Requerimientos**. La etapa de requerimientos se enfoca en la captura, documentación y priorización de requerimientos que influencian la arquitectura. Como se mencionó anteriormente, los atributos de calidad juegan un papel preponderante dentro de estos requerimientos, así que esta etapa hace énfasis en ellos. Otros requerimientos, sin embargo, son también relevantes para la arquitectura, estos son los requerimientos funcionales primarios y las restricciones.

**Diseño**. La etapa de diseño es la etapa central en relación con la arquitectura y probablemente la más compleja. Durante esta etapa se definen las estructuras que componen la arquitectura. La creación de estas estructuras se hace en base a patrones de diseño, tácticas de diseño y elecciones tecnológicas. El diseño que se realiza debe buscar ante todo satisfacer los requerimientos que influencian a la arquitectura, y no simplemente incorporar diversas tecnologías por que están “de moda”.

**Documentación.** Una vez creado el diseño de la arquitectura, es necesario poder comunicarlo a otros involucrados dentro del desarrollo. La comunicación exitosa del diseño muchas veces depende de que dicho diseño sea documentado de forma apropiada. La documentación de una arquitectura involucra la representación de varias de sus estructuras que son representadas a través de distintas vistas. Una vista generalmente contiene un diagrama, además de información adicional, que apoya en la comprensión de dicho diagrama.

**Evaluación.** Dado que la arquitectura de software juega un papel crítico en el desarrollo, es conveniente evaluar el diseño una vez que este ha sido documentado con el fin de identificar posibles problemas y riesgos. La ventaja de evaluar el diseño es que es una actividad que se puede realizar de manera temprana (aún antes de codificar), y que el costo de corrección de los defectos identificados a través de la evaluación es mucho menor al costo que tendría el corregir estos defectos una vez que el sistema ha sido construido.

El rol de arquitecto

Las actividades descritas anteriormente requieren de habilidades particulares que son la responsabilidad del arquitecto de software. El arquitecto es un líder técnico que debe conocer los principios relacionados con la arquitectura de software, tener un amplio conocimiento respecto a la tecnología, y tener excelentes habilidades de comunicación escrita y oral.

Desafortunadamente, en la actualidad pocos arquitectos de software que laboran en la industria han recibido una formación teórica respecto al tema. Esto se debe a que no es sino hasta épocas recientes que se han establecido de manera más formal los conceptos relacionados con la arquitectura de software, y que actualmente pocas instituciones ofrecen cursos enfocados en el tema. El desconocimiento de los principios relativos a la arquitectura de software frecuentemente impacta de manera negativa a los proyectos de desarrollo.

Apenas empezamos

A lo largo de las distintas entregas de esta columna que inicia se buscará dar una panorámica del tema de arquitectura de software y se discutirá de manera más detallada aspectos como:

* requerimientos que influyen en la arquitectura,
* el diseño, documentación y evaluación de la arquitectura,
* los retos relacionados con la introducción de prácticas de arquitectura de software en un contexto organizacional,
* la arquitectura de software dentro de las metodologías de desarrollo,
* el perfil del arquitecto de software.

1. DISEÑO DE LA ARQUITECTURA Vistas La arquitectura de software describe cómo un sistema es descompuesto en componentes, cómo éstos son interconectados, y la manera en que éstos se comunican e interactúan entre sí. Varias alternativas para documentar una arquitectura de software, a través de un conjunto de vistas. Cada vista representa un comportamiento particular del sistema. Dos artículos de mayor relevancia para el uso de vistas: “Modelo de 4+1 vistas de la arquitectura de software” de Philippe B. Kruchten, y “La arquitectura de software en aplicaciones industriales” de Robert L. Nord.
2. [2.](http://image.slidesharecdn.com/21vistasarquitectonicas-131021100719-phpapp01/95/2-1-vistas-arquitectonicas-2-638.jpg?cb=1382350143)DISEÑO DE LA ARQUITECTURA Vistas Modelo 4+1. Se proponen 5 vistas:
3. [3.](http://image.slidesharecdn.com/21vistasarquitectonicas-131021100719-phpapp01/95/2-1-vistas-arquitectonicas-3-638.jpg?cb=1382350143)DISEÑO DE LA ARQUITECTURA Vistas Vista lógica. Apoya principalmente los requisitos funcionales, lo que el sistema debe brindar en términos de servicios a sus usuarios. El sistema se descompone en una serie de abstracciones primarias, tomadas principalmente del dominio del problema en la forma de objetos o clases de objetos. Aquí se aplican los principios de abstracción, encapsulación y herencia. Esta descomposición no sólo se hace para potenciar el análisis funcional, sino también sirve para identificar mecanismos y elementos de diseño comunes a diversas partes del sistema.
4. [4.](http://image.slidesharecdn.com/21vistasarquitectonicas-131021100719-phpapp01/95/2-1-vistas-arquitectonicas-4-638.jpg?cb=1382350143)DISEÑO DE LA ARQUITECTURA Vistas Vista lógica.
5. [5.](http://image.slidesharecdn.com/21vistasarquitectonicas-131021100719-phpapp01/95/2-1-vistas-arquitectonicas-5-638.jpg?cb=1382350143)DISEÑO DE LA ARQUITECTURA Vistas Vista lógica.
6. [6.](http://image.slidesharecdn.com/21vistasarquitectonicas-131021100719-phpapp01/95/2-1-vistas-arquitectonicas-6-638.jpg?cb=1382350143)DISEÑO DE LA ARQUITECTURA Vistas Vista de procesos. Se tratan los aspectos de concurrencia y distribución, integridad del sistema, y tolerancia a fallos. Se especifica en cuál hilo de control se ejecuta efectivamente una operación de una clase identificada en la vista lógica. Puede ser descrita como un conjunto de redes lógicas de procesos que son ejecutados de forma independiente, y distribuidos a lo largo de varios recursos de hardware conectados mediante un bus o a una red de datos.
7. [7.](http://image.slidesharecdn.com/21vistasarquitectonicas-131021100719-phpapp01/95/2-1-vistas-arquitectonicas-7-638.jpg?cb=1382350143)DISEÑO DE LA ARQUITECTURA Vistas Vista de procesos.
8. [8.](http://image.slidesharecdn.com/21vistasarquitectonicas-131021100719-phpapp01/95/2-1-vistas-arquitectonicas-8-638.jpg?cb=1382350143)DISEÑO DE LA ARQUITECTURA Vistas Vista de desarrollo. Se centra en la organización real de los módulos de software en el ambiente de desarrollo. El software se empaqueta en partes pequeñas que pueden ser bibliotecas o subsistemas que son desarrollados por uno o un grupo de desarrolladores. Los subsistemas se organizan en una jerarquía de capas, cada una brinda una interfaz estrecha y bien definida hacia las capas superiores.
9. [9.](http://image.slidesharecdn.com/21vistasarquitectonicas-131021100719-phpapp01/95/2-1-vistas-arquitectonicas-9-638.jpg?cb=1382350143)DISEÑO DE LA ARQUITECTURA Vistas Vista de desarrollo.
10. [10.](http://image.slidesharecdn.com/21vistasarquitectonicas-131021100719-phpapp01/95/2-1-vistas-arquitectonicas-10-638.jpg?cb=1382350143)DISEÑO DE LA ARQUITECTURA Vistas Vista de desarrollo.
11. [11.](http://image.slidesharecdn.com/21vistasarquitectonicas-131021100719-phpapp01/95/2-1-vistas-arquitectonicas-11-638.jpg?cb=1382350143)DISEÑO DE LA ARQUITECTURA Vistas Vista física. Se toma en cuenta los requisitos no funcionales del sistema tales como, disponibilidad, confiabilidad, desempeño entre otras más. El sistema se ejecuta sobre varios nodos de procesamiento (hardware). Estos nodos son relacionados con los elementos identificados de las vistas anteriores. En esta vista se especifican varias configuraciones físicas. Por ejemplo, una para el desarrollo y las pruebas, o para el despliegue del sistema en plataformas distintas.
12. [12.](http://image.slidesharecdn.com/21vistasarquitectonicas-131021100719-phpapp01/95/2-1-vistas-arquitectonicas-12-638.jpg?cb=1382350143)DISEÑO DE LA ARQUITECTURA Vistas Vista física.
13. [13.](http://image.slidesharecdn.com/21vistasarquitectonicas-131021100719-phpapp01/95/2-1-vistas-arquitectonicas-13-638.jpg?cb=1382350143)DISEÑO DE LA ARQUITECTURA Vistas Kruchten define una última vista. Propone el uso de un pequeño subconjunto de escenarios que son instancias de casos de uso. La función de los escenarios es relacionar las cuatro vistas entre sí, de esta forma se cuenta con una perspectiva general del sistema, que ayuda a descubrir nuevos elementos o validar la arquitectura.
14. [14.](http://image.slidesharecdn.com/21vistasarquitectonicas-131021100719-phpapp01/95/2-1-vistas-arquitectonicas-14-638.jpg?cb=1382350143)DISEÑO DE LA ARQUITECTURA Vistas Ultima vista.
15. [15.](http://image.slidesharecdn.com/21vistasarquitectonicas-131021100719-phpapp01/95/2-1-vistas-arquitectonicas-15-638.jpg?cb=1382350143)DISEÑO DE LA ARQUITECTURA Vistas Nord realza las de mayor importancia y su uso. Este estudio se realizó en sistemas industriales como sistemas de procesamiento de señales e imágenes, sistemas operativos en tiempo real, sistemas de comunicaciones, sistemas de control de instrumentación. Se propusieron 4 vistas : Vista conceptual. Se describe el sistema en términos de sus elementos principales de diseño y las relaciones entre éstos, dentro de un dominio determinado. Esta vista es independiente de las decisiones de implementación y enfatiza en los protocolos de interacción entre los elementos de diseño. Vista de módulos. Se captura la descomposición funcional y las capas del sistema. El sistema es descompuesto lógicamente en subsistemas, módulos, y unidades abstractas. Cada capa representa las distintas interfaces de comunicación permitidas entre los módulos.
16. [16.](http://image.slidesharecdn.com/21vistasarquitectonicas-131021100719-phpapp01/95/2-1-vistas-arquitectonicas-16-638.jpg?cb=1382350143)DISEÑO DE LA ARQUITECTURA Vistas Vista de ejecución. Se describe la estructura dinámica del sistema en términos de sus elementos en tiempo de ejecución. Por ejemplo, se modela las tareas operativas del sistema, procesos, mecanismos de comunicación y asignación de recursos. Algunos de los aspectos que se consideran en esta vista son, el desempeño y el entorno de ejecución. Vista de código. Se organiza el código fuente en directorios, archivos y bibliotecas. Algunos de los aspectos que se incluyen son, los lenguajes de programación a utilizar, herramientas de desarrollo, la administración de la configuración y, la estructura y organización del proyecto.
17. [17.](http://image.slidesharecdn.com/21vistasarquitectonicas-131021100719-phpapp01/95/2-1-vistas-arquitectonicas-17-638.jpg?cb=1382350143)DISEÑO DE LA ARQUITECTURA Vistas Recomendaciones: ⎫ Documentar las vistas que sean de mayor utilidad ⎫ Documentar tomando en cuenta los intereses de los involucrados. ⎫ En sistemas muy grandes utilizar paquetes de vistas. Para seleccionar las vistas: 1. Elaborar una lista de las vistas candidatas. 2. Combinar las vistas 3. Priorizar las vistas
18. [18.](http://image.slidesharecdn.com/21vistasarquitectonicas-131021100719-phpapp01/95/2-1-vistas-arquitectonicas-18-638.jpg?cb=1382350143)DISEÑO DE LA ARQUITECTURA Vistas Una vez que las vistas se han seleccionado y priorizado, se inicia su documentación . De acuerdo al SEI: • Presentación primaria. Elementos y sus relaciones entre sí • Catálogo de elementos. Detalles de éstos, sus propiedades e interfaces. • Diagrama de contexto. Relación entre el sistema o porción de éste y su entorno. • Guía de variabilidad. Posibles puntos de variación en caso de que las vistas sean modificadas. • Antecedentes de la arquitectura. Justificación de la arquitectura así como los supuestos, y los resultados de los análisis realizados. • Otra información. Prácticas y políticas de la organización. • Paquetes de vista relacionados. Relaciones entre los distintos paquetes de vista.
19. [19.](http://image.slidesharecdn.com/21vistasarquitectonicas-131021100719-phpapp01/95/2-1-vistas-arquitectonicas-19-638.jpg?cb=1382350143)DISEÑO DE LA ARQUITECTURA Vistas ¨Vistas y más allá de estas¨ del SEI
20. [20.](http://image.slidesharecdn.com/21vistasarquitectonicas-131021100719-phpapp01/95/2-1-vistas-arquitectonicas-20-638.jpg?cb=1382350143)DISEÑO DE LA ARQUITECTURA Vistas El IEEE Software propone el estándar IEEE 1471 que define un conjunto de recomendaciones centradas en dos ideas, un marco conceptual para describir arquitecturas, y un conjunto de prácticas a seguir. θ Identificación e información general. Control de versiones del documento, fecha, histórico de revisiones, estado, contexto del sistema, declaración del alcance, entre otros más. θ Identificación del personal involucrado y sus intereses. Personas involucradas en el proyecto: administradores, diseñadores, desarrolladores, usuarios, patrocinadores. Se incluye la diversidad de intereses que la arquitectura debe satisfacer.
21. [21.](http://image.slidesharecdn.com/21vistasarquitectonicas-131021100719-phpapp01/95/2-1-vistas-arquitectonicas-21-638.jpg?cb=1382350143)DISEÑO DE LA ARQUITECTURA Vistas θ Puntos de vista. Cada de uno debe contener un nombre, personal involucrado, intereses que satisface, lenguaje o técnicas de modelado utilizados durante la construcción de la vista, algún método analítico para analizar de manera cualitativa o cuantitativa los atributos de calidad que satisface el punto de vista, y la justificación de éste. θ Vistas. Cada vista debe tener un identificador, una breve introducción y la representación del sistema con respecto a un punto de vista en particular. θ Consistencia entre vistas. Se registran las inconsistencias entre las vistas, así como algún tipo de procedimiento que indique la consistencia entre éstas. θ Justificación. Se debe incluir la justificación del tipo de arquitectura seleccionada, y de los puntos de vista utilizados.
22. [22.](http://image.slidesharecdn.com/21vistasarquitectonicas-131021100719-phpapp01/95/2-1-vistas-arquitectonicas-22-638.jpg?cb=1382350143)DISEÑO DE LA ARQUITECTURA Vistas
23. [23.](http://image.slidesharecdn.com/21vistasarquitectonicas-131021100719-phpapp01/95/2-1-vistas-arquitectonicas-23-638.jpg?cb=1382350143)DISEÑO DE LA ARQUITECTURA Vistas Conclusiones. Uno de los principales beneficios de la documentación de arquitecturas es el poder efectuar evaluaciones sobre la arquitectura documentada , para determinar si se están cumpliendo o no los intereses del personal involucrado. • Documentar tomando en cuenta las necesidades e intereses de quienes forman parte del proyecto. • Acompañar la documentación de las vistas con un modelo analítico que ayude a predecir el comportamiento de los atributos de calidad. • UML no es el único lenguaje para documentar la arquitectura. Existen diferentes notaciones y lenguajes para este propósito (ADLS) • Mantener una relación consistente entre las vistas. • Elaborar plantillas de estilos para promover la reutilización de artefactos dentro de la organización. • Mantener actualizada la matriz de trazabilidad entre los requisitos y los elementos de la arquitectura. • Tener bajo una línea base el documento de la arquitectura.

# **Arquitectura de software**

En los inicios de la informática, la [programación](https://es.wikipedia.org/wiki/Programaci%C3%B3n) se consideraba un arte y se desarrollaba como tal, debido a la dificultad que entrañaba para la mayoría de las personas, pero con el tiempo se han ido descubriendo y desarrollando formas y guías generales, con base a las cuales se puedan resolver los problemas. A estas, se les ha denominado Arquitectura de Software, porque, a semejanza de los planos de un edificio o 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".

## Índice

  [[ocultar](https://es.wikipedia.org/wiki/Arquitectura_de_software)]

* [1 Arquitectura](https://es.wikipedia.org/wiki/Arquitectura_de_software#Arquitectura)
* [2 Breve reseña histórica](https://es.wikipedia.org/wiki/Arquitectura_de_software#Breve_rese.C3.B1a_hist.C3.B3rica)
* [3 Modelos o vistas](https://es.wikipedia.org/wiki/Arquitectura_de_software#Modelos_o_vistas)
* [4 Arquitecturas más comunes](https://es.wikipedia.org/wiki/Arquitectura_de_software#Arquitecturas_m.C3.A1s_comunes)
* [5 Bibliografía](https://es.wikipedia.org/wiki/Arquitectura_de_software#Bibliograf.C3.ADa)
* [6 Véase también](https://es.wikipedia.org/wiki/Arquitectura_de_software#V.C3.A9ase_tambi.C3.A9n)
* [7 Enlaces externos](https://es.wikipedia.org/wiki/Arquitectura_de_software#Enlaces_externos)

## **Arquitectura[[editar](https://es.wikipedia.org/w/index.php?title=Arquitectura_de_software&action=edit&section=1" \o "Editar sección: Arquitectura)]**

* La Arquitectura del Software es el diseño de más alto nivel de la estructura de un sistema.
* Una Arquitectura de [Software](https://es.wikipedia.org/wiki/Software), también denominada *Arquitectura*[*lógica*](https://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 (requerimientos) 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](https://es.wikipedia.org/wiki/Arquitectura) física, que consiste simplemente en determinar qué [computadora](https://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](https://es.wikipedia.org/wiki/Confiabilidad).

## **Breve reseña histórica[[editar](https://es.wikipedia.org/w/index.php?title=Arquitectura_de_software&action=edit&section=2" \o "Editar sección: Breve reseña histórica)]**

En los [años 1960](https://es.wikipedia.org/wiki/A%C3%B1os_1960) ya se acercaba el concepto de arquitectura de software en los círculos de investigación (por ejemplo, por[Edsger Dijkstra](https://es.wikipedia.org/wiki/Edsger_Dijkstra)). No obstante, toma popularidad en los [años 1990](https://es.wikipedia.org/wiki/A%C3%B1os_1990) tras reconocerse la denominada [*crisis del software*](https://es.wikipedia.org/wiki/Crisis_del_software) y como tema de interés de la incipiente disciplina de la [ingeniería del software](https://es.wikipedia.org/wiki/Ingenier%C3%ADa_del_software).

## **Modelos o vistas[[editar](https://es.wikipedia.org/w/index.php?title=Arquitectura_de_software&action=edit&section=3" \o "Editar sección: 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](https://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](https://es.wikipedia.org/wiki/Lenguaje). El más obvio es el [lenguaje natural](https://es.wikipedia.org/wiki/Lenguaje_natural), pero existen otros lenguajes tales como los [diagramas de estado](https://es.wikipedia.org/w/index.php?title=Diagrama_de_estado&action=edit&redlink=1), los [diagramas de flujo de datos](https://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](https://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[**[**editar**](https://es.wikipedia.org/w/index.php?title=Arquitectura_de_software&action=edit&section=4)**]**

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:

* [Descomposición Modular](https://es.wikipedia.org/w/index.php?title=Dise%C3%B1o_Estructurado&action=edit&redlink=1). Donde el software se estructura en grupos funcionales muy acoplados.
* [Cliente-servidor](https://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](https://es.wikipedia.org/wiki/Arquitectura_de_tres_niveles). Especialización de la arquitectura cliente-servidor donde la carga se divide en tres partes (o capas) con un reparto claro de funciones: una capa para la presentación (interfaz de usuario), otra para el cálculo (donde se encuentra modelado el negocio) y otra para el almacenamiento (persistencia). Una capa solamente tiene relación con la siguiente.

Otras arquitecturas afines menos conocidas son:

* [Modelo Vista Controlador](https://es.wikipedia.org/wiki/Modelo_Vista_Controlador).
* [En *pipeline*](https://es.wikipedia.org/wiki/Arquitectura_en_pipeline_(inform%C3%A1tica)).
* [Entre pares](https://es.wikipedia.org/w/index.php?title=Arquitectura_peer-to-peer_(inform%C3%A1tica)&action=edit&redlink=1).
* [En pizarra](https://es.wikipedia.org/wiki/Arquitectura_en_pizarra_(inform%C3%A1tica)).
* [Orientada a servicios](https://es.wikipedia.org/wiki/Arquitectura_orientada_a_servicios_(inform%C3%A1tica)).
* [Dirigida por eventos](https://es.wikipedia.org/wiki/Arquitectura_dirigida_por_eventos).
* [Máquinas virtuales](https://es.wikipedia.org/w/index.php?title=Arquitectura_de_m%C3%A1quinas_virtuales_(inform%C3%A1tica)&action=edit&redlink=1)