Especialidad Análisis, diseño y desarrollo de software

D1.7 Vistas arquitectónicas 4+1

Asesor: M.T.I.C. Leonardo Enriquez

Ingeniero electrónico, sistemas digitales

Decisiones en el diseño arquitectónico

El diseño arquitectónico es un proceso creativo en el cual se diseña una organización del sistema que cubrirá los **requerimientos funcionales y no funcionales de este.**

Durante el proceso de diseño arquitectónico, los arquitectos del sistema deben tomar algunas decisiones estructurales que afectaran el sistema y su proceso de desarrollo. Cada decisión incorporada en una arquitectura de software puede afectar potencialmente los atributos de calidad.**, por lo que considerar lo siguiente:**

1. ¿Existe alguna arquitectura de aplicación genérica que actúe como plantilla para el sistema que se está diseñando?

2. ¿Cómo se distribuirá el sistema a través de algunos núcleos o procesadores?

3. ¿Qué patrones o estilos arquitectónicos pueden usarse?

4. ¿Cuál será el enfoque fundamental usado para estructurar el sistema?

5. ¿Cómo los componentes estructurales en el sistema se separarán en subcomponentes?

6. ¿Qué estrategia se usará para controlar la operación de los componentes en el sistema?

7. ¿Cuál organización arquitectónica es mejor para entregar los requerimientos no funcionales del sistema?

8. ¿Cómo se evaluará el diseño arquitectónico?

9. ¿Cómo se documentará la arquitectura del sistema?

La arquitectura de un sistema de software puede basarse en un patrón o un estilo arquitectónico particular, tal como una organización cliente-servidor o una arquitectura por capas.

Decisiones en el diseño arquitectónico

1. ¿Existe alguna arquitectura de aplicación genérica que actúe como plantilla para el sistema que se está diseñando?

- Aunque cada sistema de software es único, los sistemas en el mismo dominio de aplicación tienen normalmente arquitectura similares que reflejan los conceptos fundamentales del dominio.
- Por ejemplo, las líneas de producto de aplicación son aplicaciones que se construyen en torno a una arquitectura central con variantes que cubren requerimientos específicos del cliente. Cuando se diseña una arquitectura de sistema, debe decidirse que tienen en común el sistema y las clases de aplicación mas amplias, con la finalidad de determinar cuanto conocimiento se puede reutilizar de dichas arquitecturas de aplicación. Los patrones arquitectónicos son medios para reutilizar el conocimiento las arquitecturas de sistemas genéricos. Describen la arquitectura, explican cuando debe usarse y exponen sus ventajas y desventajas.
- Los patrones arquitectónicos usados comúnmente incluyen el modelo de vista del controlador, arquitectura en capas, repositorio, cliente-servidor, y tubería y filtro.

La arquitectura de un sistema de software puede basarse en un patrón o un estilo arquitectónico particular, tal como una organización cliente-servidor o una arquitectura por capas.

Decisiones en el diseño arquitectónico

1. ¿Existe alguna arquitectura de aplicación genérica que actúe como plantilla para el sistema que se está diseñando? (continuación pregunta 1…)

Tipos de sistemas de aplicación: (Ingeniería de Software, Somerville, Capitulo 6, sección 4 y Capitulo 18)

- Los **modelos genéricos** de las arquitectura de sistemas de aplicación ayudan a entender la operación de las aplicaciones, compara aplicaciones del mismo tipo, validar diseños del sistema de aplicación y valorar componentes para reutilización a gran escala.
- Los **sistemas de procesamiento de transacción** son sistemas interactivos que permiten el acceso y la modificación remota de la información, en una base de datos por parte de varios usuarios. Los sistemas de información y los sistemas de gestión de recursos son ejemplos de sistema de procesamiento de transacciones.
- Los **sistemas de procesamiento de lenguaje** se usan para traducir textos de un lenguaje a otro y para realizar las instrucciones especificadas en el lenguaje de entrada. Incluyen un traductor y una maquina abstracta que ejecuta el lenguaje generado.

Decisiones en el diseño arquitectónico

2. ¿Cómo se distribuirá el sistema a través de algunos núcleos o procesadores?

- Para sistemas embebidos y sistemas diseñados para computadores personales, por lo general, hay un solo procesador y no tendrá que diseñar una arquitectura distribuida para el sistema. Sin embargo, los sistemas mas grandes ahora son sistemas distribuidos donde el software de sistema se distribuye a través de muchas y diferentes computadoras. La elección de arquitectura de distribución es una decisión clave que afecta el rendimiento y la fiabilidad del sistema. (Ingeniería de Software, Somerville, Capitulo 18)

3. ¿Qué patrones o estilos arquitectónicos pueden usarse?

- La arquitectura de un sistema de software puede basarse en un patrón o un estilo arquitectónico particular. Un patrón arquitectónico es una descripción de una organización del sistema, tal como una organización cliente-servidor o una arquitectura por capas. Los patrones arquitectónicos captan la esencia de una arquitectura que se uso en diferentes sistemas de software, por lo que se tiene que conocer tanto los patrones comunes, en que estos se usen, como sus fortalezas y debilidad cuando se tomen decisiones sobre la arquitectura de un sistema. (Ingeniería de Software, Somerville, Sección 6.3)

Decisiones en el diseño arquitectónico

4. ¿Cuál será el enfoque fundamente usado para estructura el sistema?

5. ¿Cómo los componentes estructurales en el sistema se separarán en subcomponentes?

6. ¿Qué estrategia se usará para controlar la operación de los componentes en el sistema?

- Los enfoques que se puede usarse es permitir la implementación de diferentes tipos de arquitectura como cliente-servidor o estructura en capas que le permita satisface los requerimientos del sistema, o se puede descomponer las unidades del sistema estructura optando por la estrategia de separar los componentes en subcomponentes (pregunta 5) y finalmente en el proceso de modelado de control, se toman decisiones sobre como se controla la ejecución de componentes (pregunta 6).

Decisiones en el diseño arquitectónico

7. ¿Cuál organización arquitectónica es mejor para entregar los requerimientos no funcionales del sistema?

- Debido a la estrecha relación entre los requerimientos funcionales y la arquitectura de software, el estilo y la arquitectura arquitectónicas particulares que se elijan para un sistema dependerán de los requerimientos de sistema no funcionales, por lo que si el:
    - **El rendimiento es critico**, la arquitectura debe diseñarse para localizar operaciones criticas dentro de un pequeño numero de componentes, con todos estos componentes desplegados en la misma computado en vez de distribuirlo por la red. Esto significaría usar algunos **componentes** relativamente grandes, en vez de pequeños componentes de grano fino, lo cual reduce el numero de comunicaciones entre componentes. También puede considerar organización del sistema en tiempo de operación que permitan a este ser replicable y ejecutable en diferentes procesadores.
    - **La seguridad es critica**, será necesario usar una **estructura en capas para la arquitectura**, con los activos mas críticos protegidos en las capas mas internas, y con un alto nivel de validación de seguridad aplicado a dichas capas.
    - **La protección es critica**, la arquitectura debe diseñarse de modo que las operaciones relacionadas con la protección se ubiquen en algún componente individual o en un pequeño numero de **componentes**. Esto reduce los costos y problemas de validación de la protección, y hace posible ofrecer sistema de protección relacionados que, en caso de falla, desactiven con seguridad el sistema.
    - **La disponibilidad es critica**, la arquitectura tiene que ser diseñada para incluir componentes redundantes de manera que sea posible sustituir y actualizar componentes sin detener el sistema. (Ingeniería de Software, Somerville, capitulo 13 Sistemas tolerantes a fallas en sistema de alta disponibilidad), por ejemplo Arquitecturas de *auto monitorización y programación de n-versión*)
    - **La mantenibilidad es critica**, la arquitectura del sistema debe diseñarse usando componentes autocontenidos de grano fino que puedan cambiarse con facilidad. Los productores de datos tienen que separarse de los consumidores y hay que evitar compartir las estructuras de datos.

Decisiones en el diseño arquitectónico

8. ¿Cómo se evaluará el diseño arquitectónico?

- De acuerdo con la pregunta anterior es evidente que hay un conflicto potencial entre algunas de las arquitecturas mencionadas, por ejemplo utilizar componentes grandes mejora el rendimiento, y utilizar componentes pequeños de grano fino aumenta la mantenibilidad. Si tanto el rendimiento como la mantenibilidad son requerimientos importantes del sistema, entonces se debe encontrar algún compromiso. Esto en ocasiones se logra usando diferentes patrones o estilos arquitectónicos para distintas partes del sistema.
- Evaluar un diseño arquitectónico es difícil porque la verdadera prueba de una arquitectura es que tan bien el sistema cubra sus requerimientos funcionales y no funcionales cuando esta en uso. Sin embargo, es posible hacer cierta evaluación al compara el diseño contra arquitecturas de referencias o patrones arquitectónicos genéricos. Para ayudar con la evaluación arquitectónica, también puede usarse la descripción de Bosch (2000) de las características no funcionales de los patrones arquitectónicos para los atributos de calidad.

Atributo de Calidad	Descripción
Disponibilidad *(Availability)*	Es la medida de disponibilidad del sistema para el uso (Barbacci et al., 1995).
Confidencialidad *(Confidentiality)*	Es la ausencia de acceso no autorizado a la información (Barbacci et al., 1995).
Funcionalidad *(Functionality)*	Habilidad del sistema para realizar el trabajo para el cual fue concebido (Kazman et al., 2001).
Desempeño *(Performance)*	Es el grado en el cual un sistema o componente cumple con sus funciones designadas, dentro de ciertas restricciones dadas, como velocidad, exactitud o uso de memoria. (IEEE 610.12). Según Smith (1993), el desempeño de un sistema se refiere a aspectos temporales del comportamiento del mismo. Se refiere a capacidad de respuesta, ya sea el tiempo requerido para responder a aspectos específicos o el número de eventos procesados en un intervalo de tiempo. Según Bass et al. (1998), se refiere además a la cantidad de comunicación e interacción existente entre los componentes del sistema.
Confiabilidad *(Reliability)*	Es la medida de la habilidad de un sistema a mantenerse operativo a lo largo del tiempo (Barbacci et al., 1995).
Seguridad externa *(Safety)*	Ausencia de consecuencias catastróficas en el ambiente. Es la medida de ausencia de errores que generan pérdidas de información (Barbacci et al., 1995).
Seguridad interna *(Security)*	Es la medida de la habilidad del sistema para resistir a intentos de uso no autorizados y negación del servicio, mientras se sirve a usuarios legítimos (Kazman et al., 2001).

Decisiones en el diseño arquitectónico

9. ¿Cómo se documentara la arquitectura del sistema?

- Los modelos arquitectónico de un sistema de software sirven para enfocar la discusión sobre los requerimientos o el diseño del software. De manera alternativa, pueden emplearse para documentar un diseño, de modo que se usen como base en el diseño y la implementación mas detallados, así como en la evolución futura del sistema.
- Es imposible representar toda la información relevante sobre la arquitectura de un sistema en un solo modelo arquitectónico, ya que cada uno presenta una vista o perspectiva del sistema. Esta puede mostrar como un sistema se descompone en módulos, como interactúan los procesos de tiempo de operación o las diferentes formas en que los componentes del sistema se distribuyen a través de una red.
- Krutchen (1995), en su modelo de vista 4+1 de la arquitectura de software, sugiere que deben existir cuatro vistas arquitectónicas fundamente, que se relacionan usando casos de uso o escenarios. Las vistas que el sugiere son:

    1. **Una vista lógica**.
    2. **Una vista de proceso**
    3. **Una vista de desarrollo**.
    4. **Una vista física**.

Vistas arquitectónicas

Los estilos y patrones ayudan al arquitecto a definir la composición y el comportamiento del sistema de software, y una combinación adecuada de ellos permite alcanzar los requerimientos de calidad. Ahora bien la organización de un sistema debe estar disponible para todos los involucrados, estableciendo una comunicación entre los mismos. Tal objetivo se logra mediante la representación de la arquitectura a través de vistas arquitectónicas, las notaciones como UML y los lenguajes de descripción arquitectónica (Bengtsson, 1999).

De acuerdo al nivel de responsabilidad dentro del desarrollo de un sistema y la relación que se establezca con el mismo, son muchas las partes involucradas e interesados en la arquitectura de software (Kruchten, 1999).

- El analista del sistema, quien la utiliza para organizar y expresar los requerimientos y entender las restricciones de tecnología y los riesgos.
- Usuarios finales y clientes, que necesitan conocer el sistema que están adquiriendo.
- El gerente del proyecto, que la utiliza para organizar el equipo y planificar el desarrollo
- Los diseñadores, que lo utilizan para entender los principios subyacentes y localizar los limites de su propio diseño.
- Otras organizaciones desarrolladoras si el sistema es abierto.
- Las compañías subcontratadas que la utilizan para entender los limites de su sección de desarrollo
- Los arquitectos, quien velan por la evolución del sistema y la reutilización de componentes.

Vistas arquitectónicas 4+1 por Krutchen

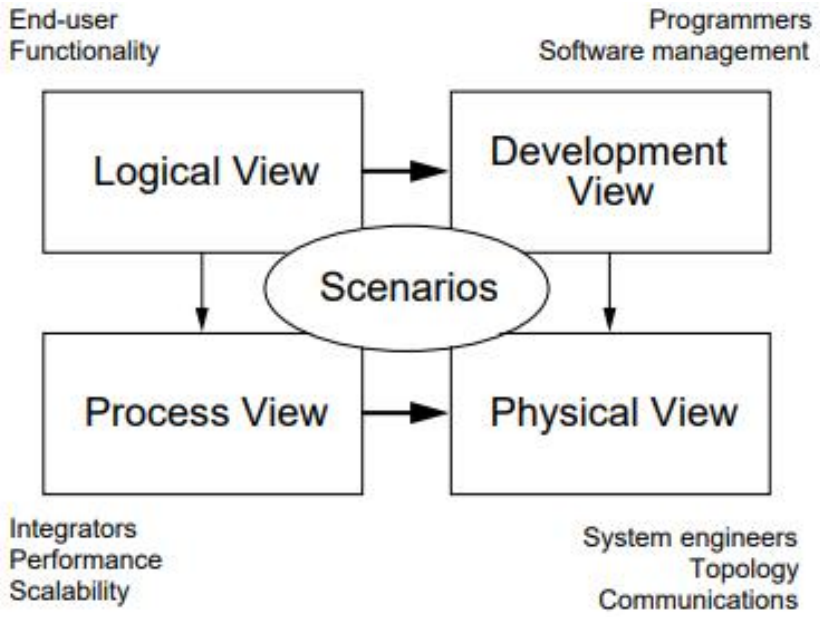
Krutchen (1999) define una vista arquitectónica como una descripción simplificada o abstracción de un sistema desde una perspectiva especifica, que cubre intereses particulares y omite entidades no relevantes a esta perspectiva.

Los modelos arquitectónico de un sistema de software además de servir para enfocar la discusión sobre los requerimientos o el diseño del software, pueden emplearse para documentar un diseño, de modo que se usen como base en el diseño y la implementación, así como en la evolución futura del sistema.

Krutchen propone un sistema de software se debe documentar y mostrar (tal como se propone en el estándar IEEE 1471-2000) con 4 vistas diferenciadas y estas 4 se han de relacionar entre si con una vista mas, que es la denominada la vista “+1”.

Estas 4 vista las denomino Krutchen como:

- Vista lógica,
- Vista de proceso,
- Vista de desarrollo,
- Vista física,
- Y la vista escenarios “+1”



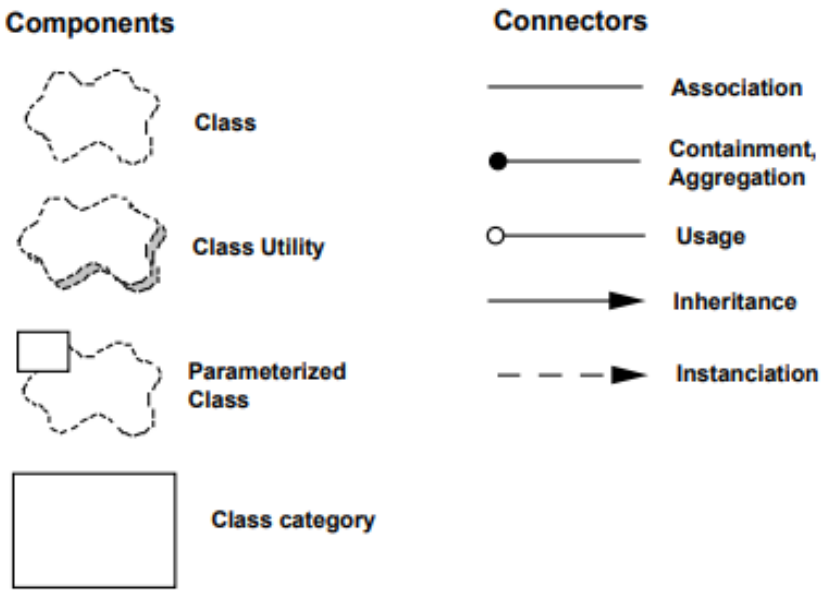
Modelo de vista “4+1”

Vistas arquitectónicas 4+1 (vista lógica)

La arquitectura lógica (descomposición de la orientación a objetos)

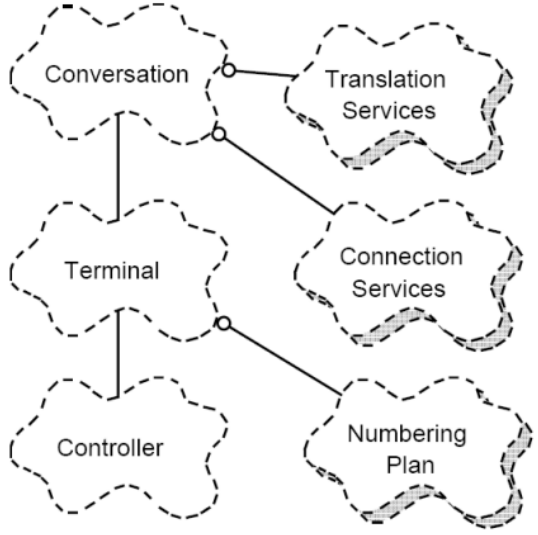
- La arquitectura **lógica** apoya los requisitos funcionales (lo que el sistema debe brindar en términos de servicios a sus usuarios).
- El sistema se descompone en una serie de abstracciones clave, tomadas del dominio del problema en la forma de objetos o clases de objetos.
- Se aplican los principios de abstracción, encapsulamiento y herencia.
- Vista **Lógica**, representa la funcionalidad que el sistema proporcionara a los ***usuarios finales***. Se ha de representar lo que el sistema debe hacer, y las funciones y servicios que ofrece.

Notaciones para la vista lógica



Estilos para la vista lógica

- Orientación a objetos
- Esta puede representa con UML utilizando diagrama de clase, comunicación, y secuencia.



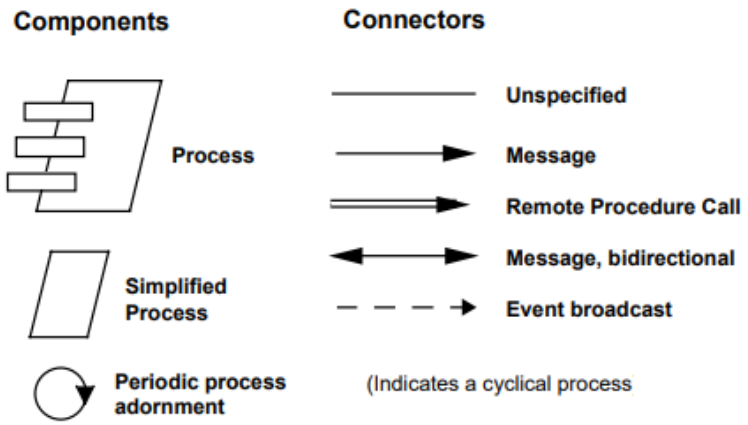
Ejemplo de la notación

Vistas arquitectónicas 4+1 (vista de proceso)

La arquitectura del proceso (descomposición del proceso)

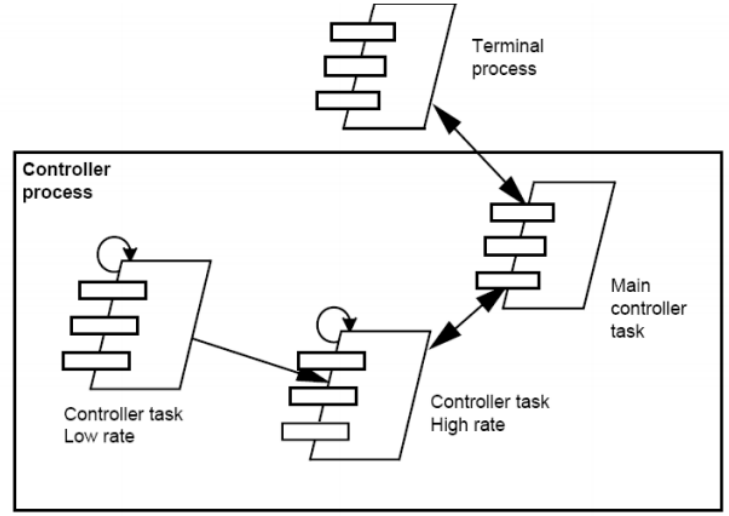
- **La arquitectura de procesos** toma en cuenta algunos requisitos **no funcionales** tales por el rendimiento y disponibilidad.
- Se enfoca en asuntos de concurrencia y distribución, integridad del sistema, de tolerancia a fallas.
- **Una vista de proceso,** que muestre como, en el tiempo de operación, el sistema esta compuesto de procesos en interacciones.
- La arquitectura de procesos se describe en varios niveles de abstracción, donde el nivel mas alto puede verse como un conjunto de redes lógicas de programas comunicantes ejecutándose en forma independiente, y distribuidos a lo largo de un conjunto de recursos de hardware conectados mediante un bus, una LAN, o WAN.
- En esta se representa la perspectiva de un ***integrador de sistemas***, el flujo de trabajo paso a paso de negocio y operacionales de los componentes que conforman el sistema.

Notaciones para la vista de proceso



Estilos para la vista proceso

- Pipes y filters, cliente-Servidor, y variantes a múltiples clientes simples y múltiples cliente / múltiples servidores.
- Esta puede representa con UML utilizando diagrama de Actividad



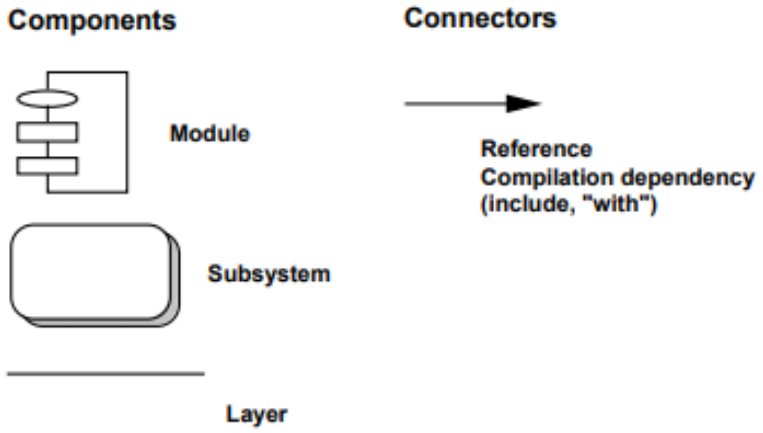
Ejemplo de la notación

Vistas arquitectónicas 4+1 (vista de desarrollo)

La arquitectura de desarrollo (descomposición de subsistemas)

- **Una vista de desarrollo,** que muestre como el software esta descompuesto para su desarrollo, indicando la descomposición del software en elementos que se implementen mediante un solo **desarrollador o equipo de desarrollo**. Esta vista es útil para administradores y programadores de software.
- En esta vista se muestra el sistema desde la perspectiva de *un programador* y se ocupa de la gestión del software; muestra como esta dividido el sistema software en componentes y las dependencias que hay entre esos componentes.

Notaciones para la vista de desarrollo



Estilos para la vista desarrollo

- Capas
- Esta puede representa con UML utilizando diagrama de componentes y diagrama de paquetes

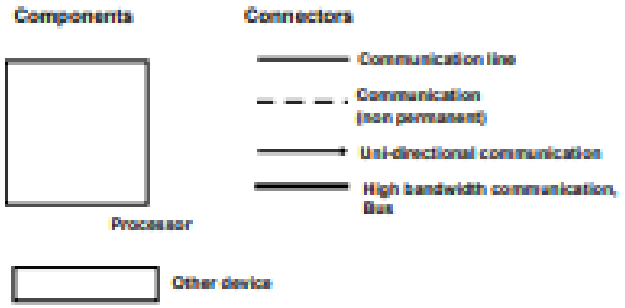
Vistas arquitectónicas 4+1 (arquitectura física)

La arquitectura física (mapeo del software al hardware)

- Esta vista toma los requerimientos funciones del sistema tales como disponibilidad, tolerancia al fallo, escalabilidad, …

- Una vista física, expone el hardware del sistema así como los componentes de software que se distribuyen a través de los procesadores en el sistema. Esta vista es útil para los ingenieros de desarrollo y pruebas que plantean una implementación de sistema.

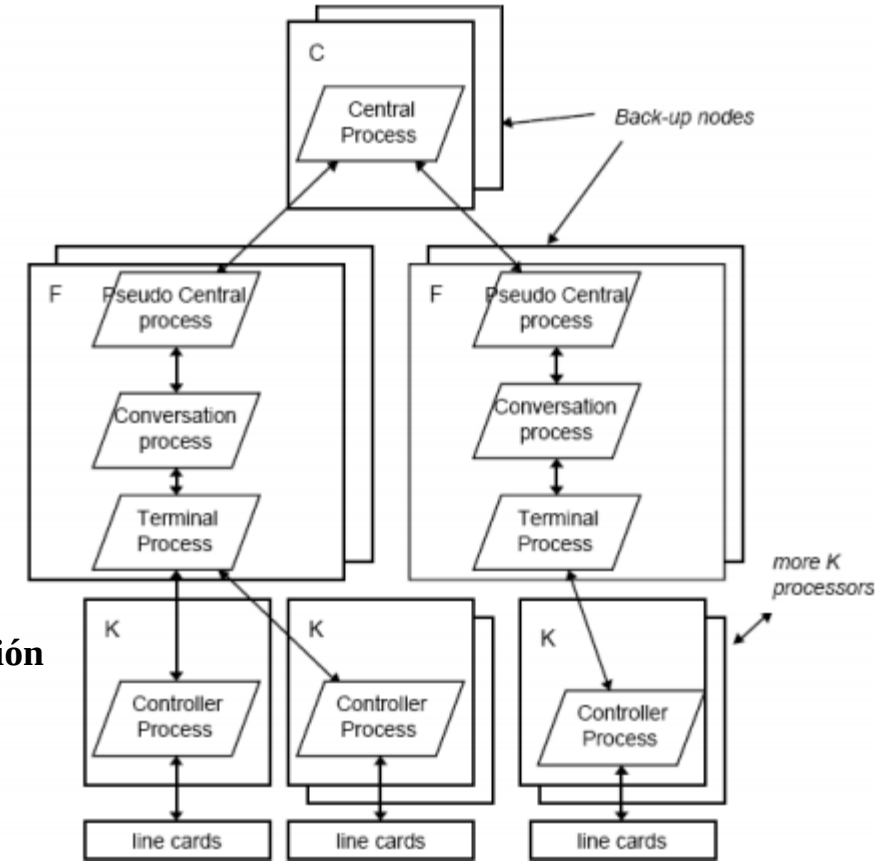
Notaciones para la vista de desarrollo



Estilos para la vista física

- Esta puede representa con UML utilizando diagrama de despliegue

Ejemplo de la notación

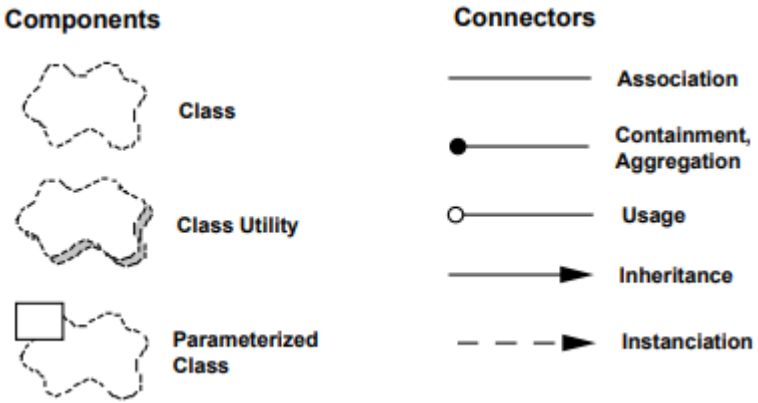


Vistas arquitectónicas 4+1 (arquitectura de escenarios)

“+1” Vista de Escenarios:

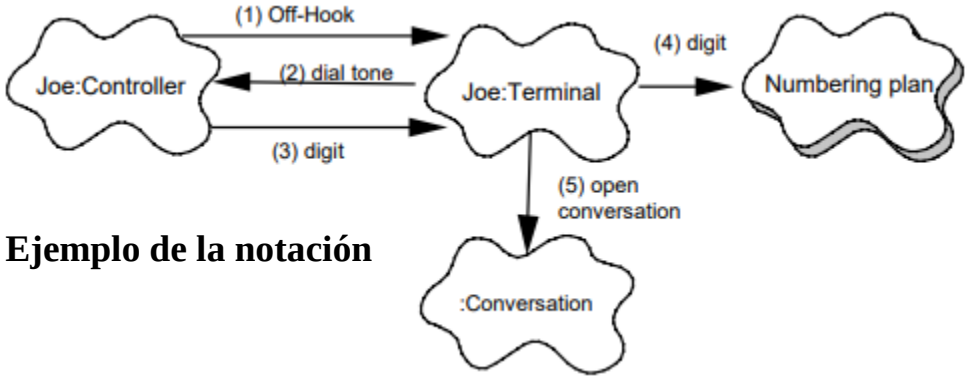
- Esta vista va a ser representada por los casos de uso software y va a tener la función de unir y relacionar las otras 4 vistas, esto quiere decir que desde un caso de uso podemos ver como se van ligando las otras 4 vistas, con lo que tendremos una trazabilidad de componentes, clases, equipos, paquetes, etc.

Notaciones para la vista de desarrollo



Estilos para la vista de escenarios

- Esta puede representa con UML utilizando diagrama de casos de uso



Ejemplo de la notación

Somerville, en su libro Ingeniería de Software, indica que **UML no es útil** durante el proceso de diseño en si y prefiere notaciones informales que sean rápidas de dibujar. El UML se diseño para describir sistemas orientados a objetos, y en la etapa de diseño arquitectónico, uno quiere describir con frecuencia sistema en un nivel superior de abstracciones.

Vistas arquitectónicas relacionadas y representadas con UML

Conceptual / Logical

Physical / Operational

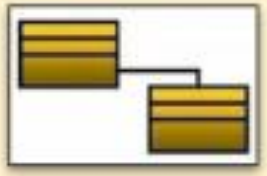
Functional

Non-functional

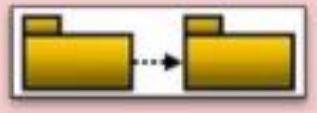
Logical / Structural view

Perspective: Analysts, Designers
Stage: Requirement analysis
Focus: Object oriented decomposition
Concerns: Functionality
Artefacts:

- Class diagram
- Object diagram
- Composite structure diagram



Implementation / Developer view



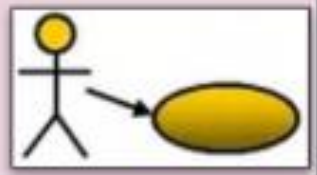
Perspective: Developers, Proj. mngs.
Stage: Design
Focus: Subsystem decomposition
Concerns: Software management
Artefacts:

- Component diagram
- Package diagram

Use Case/Scenario view

Perspective: End users
Stage: Putting it alltogether
Concerns: Understandability, usability
Focus: Feature decomposition
Artefacts:

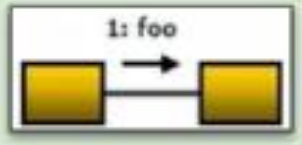
- Use-case diagram
- User stories



Process / Behaviour view

Perspective: System Integrators
Stage: Design
Focus: Process decomposition
Concerns: Performance, scalability, throughput
Artefacts:

- Sequence diagram
- Communication diagram
- Activity diagram
- State (machine) diagram
- Interaction overview diagram
- Timing diagram



Deployment / Physical view

Perspective: System Engineers
Stage: Design
Focus: Map software to hardware
Concerns: System topology, delivery, installation, communication
Artefacts:

- Deployment diagram
- Network topology (not UML)

Comparación de vistas arquitectónicas

Comparación de vistas arquitectónicas en función de las perspectivas del sistema

Perspectiva	Kazman, et al. (2001)	Kruchten (1999)	Hofmeister, et al. (2000)	Bass et al. (1998)	Parte Interesada	Atributo de Calidad
Abstracción de requerimientos funcionales del sistema	Vista Funcional	Vista Lógica	Vista Conceptual	Vista Conceptual o Lógica	Cliente Usuario final Analista	Modificabilidad Reusabilidad Dependencia Seguridad Externa
Creación de procesos e hilos de ejecución, comunicación entre ellos y recursos compartidos.	Vista de Concurrencia	Vista de Proceso	Vista de Ejecución	Vista de Procesos o Coordinación + Vista de Llamadas	Arquitectos Desarrolladores Equipo de Pruebas Mantenimiento	Desempeño Disponibilidad
Organización de los elementos Arquitectónicas implementados.	Vista de Desarrollo	Vista de Implantación	Vista de Código	Vista Física + Vista de Módulos	Programadores Mantenimiento Gerentes de Configuración Gerentes de Desarrollo	Mantenibilidad Modificabilidad Capacidad de Prueba
Distribución de procesos en la plataforma	Vista Física + Vista de Concurrencia	Vista de Desarrollo	Vista de Módulos y Vista de Ejecución	Vista de Flujo de Control	Arquitectos Desarrolladores Equipo de Pruebas Mantenimiento Ing. Hardware	Desempeño Escalabilidad Disponibilidad Seguridad Interna
Escenarios y casos de uso	-	Vista de Casos de Uso	Vista Conceptual	Vista de Usos	Cliente Usuario final Analista	Reusabilidad Disponibilidad Modificabilidad
Especificación abstracta de clases, objetos, funciones y procedimientos.	Vista de Código	-	-	Vista de Clases + Vista de Flujo de Datos	Diseñadores Desarrolladores	Modificabilidad Portabilidad Mantenibilidad

Bibliografía

Pressman, R. S. (2010). Ingeniería de Software, Un enfoque practico Séptima Edición. Ciudad de México: Mc Graw Hill.

Sommerville. (2011). Ingeniería de Software 9 Edición. Estado de México: Pearson.

UNID Universidad Interamericana para el desarrollo. (2018). Ingeniería de software. Ciudad de México: UNID.

Philippe Kruchten, Rational Software Corp, Papel published in IEEE, Noviembre 1995, pp. 42-50