DOCUMENTACION DE ARQUITECTURA – VISTA DE COMPONENTES Y CONECTORES VISTA DE ASIGNACION

ELSA ESTEVEZ
UNIVERSIDAD NACIONAL DEL SUR
DEPARTAMENTO DE CIENCIAS E INGENIERIA DE LA COMPUTACION

CONTENIDO

VISTA DE COMPONENTES Y CONECTORES – INTRODUCCION

COMPONENTES

Elementos que tienen alguna presencia en ejecución:

- procesos
- objetos
- clientes
- servidores
- almacenamiento de datos

CONECTORES

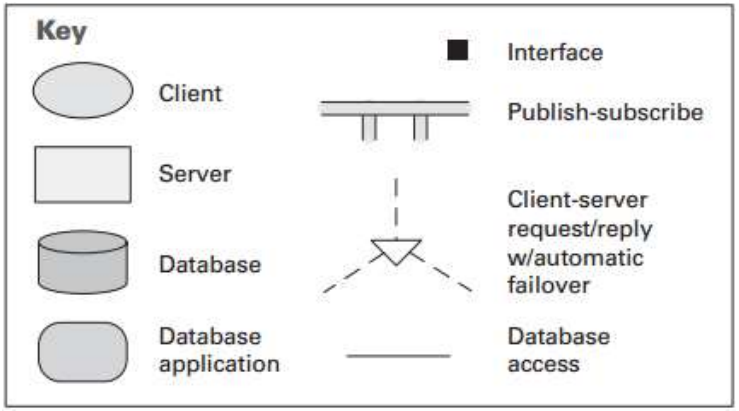
Caminos de interacción entre los componentes:

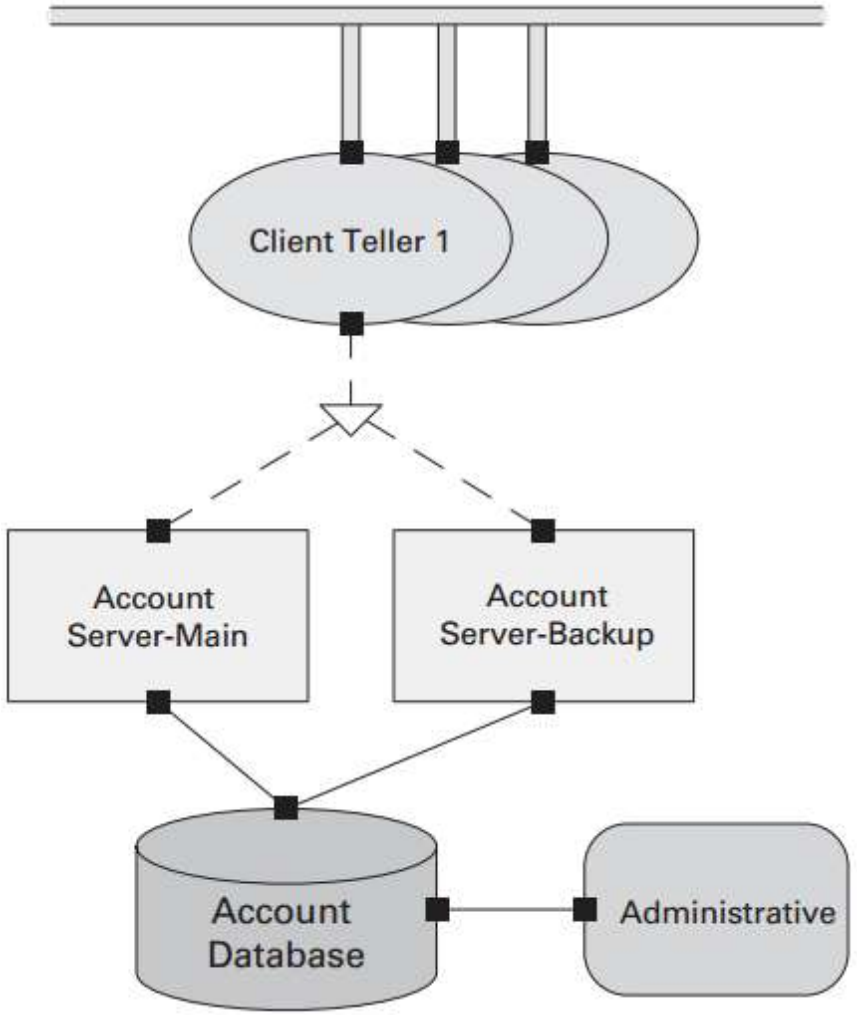
- enlaces de comunicación
- protocolos de comunicación
- flujos de información
- accesos a almacenamientos compartido

VISTA DE COMPONENTES Y CONECTORES – EJEMPLO 1

Diagrama de alto nivel de
un sistema en ejecución

Como interpretamos el diagrama?



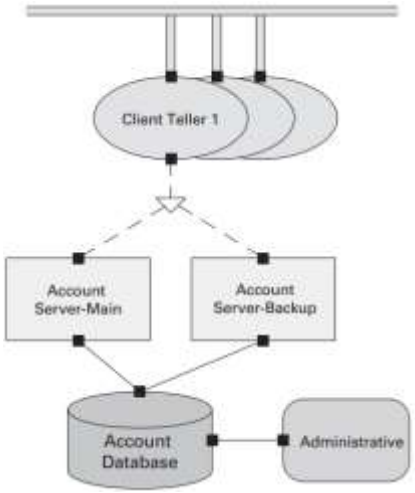


VISTA DE COMPONENTES Y CONECTORES – EJEMPLO 2

Se presentan 3 tipos de conectores diferentes:

- Client-server - permiten que un conjunto de clientes concurrentes recuperen datos sincrónicamente a través de requerimientos de servicios. Soporta failover transparente a un servidor de backup
- Acceso a la base de datos - soporta acceso transaccional y autenticado para leer, escribir y monitorear la base de datos
- Publish-subscribe - soporta envío de eventos y notificaciones asincrónicas.

La implementación de cada conector tiene cierta complejidad asociada.



DOCUMENTACION DE LA VISTA

Documentación gráfica

- presenta detalles acerca de los elementos, relaciones y propiedades
- se restringe a la información que puede ser presentada en un único diagrama
- indica la cantidad y tipo de interfaces sobre sus componentes y conectores
- usa abstracciones de componentes y conectores que pueden tener semánticas ricas e implementaciones complejas

Documentación de soporte

- elabora sobre los elementos mostrados en el diagrama, proveyendo detalles sobre su implementación y su aporte a los atributos de calidad requeridos.

  Ejemplo: el Account Server-Backup mejora la disponibilidad del sistema

DOCUMENTACION – PROPOSITO

La combinación de diagramas de componentes y conectores y su documentación de soporte provee un vehículo esencial para comunicar un intento de diseño de arquitectura:

- facilita el razonamiento sobre el comportamiento del sistema en ejecución
- justifica las decisiones de diseño en términos del impacto sobre los atributos de calidad relevantes.

ELEMENTOS

Cada elemento de una vista de Componentes y Conectores tiene su correspondencia en tiempo de ejecución, consumiendo recursos y contribuyendo al comportamiento del sistema en ejecución.

ELEMENTOS:

1) Componentes

2) Conectores

COMPONENTES

Un componente representa a los principales elementos computacionales o de almacenamiento presentes en ejecución.

Tiene un nombre que sugiere la función que cumple y permite buscar su descripción en la documentación de soporte.

Un componente en una vista de componentes y conectores puede representar a un subsistema complejo, cuya sub-arquitectura puede ser mostrada en el mismo diagrama (anidada al componente) o en otro.

Los componente tienen interfaces llamadas **puertos**.

PUERTOS

Un puerto es una interface de un componente.

Características:

- define un punto específico de interacción potencial de un componente con su ambiente
- tiene un tipo explícito que define el tipo de comportamiento que tiene lugar
- un componente puede tener muchos puertos del mismo tipo.
- los puerto pueden anotarse con un número (o rango de números) para indicar replicación.
- los puertos deben ser documentados explícitamente, mostrándolos en el diagrama y definiéndolos en la documentación de soporte.

CONECTORES

Es el camino en ejecución para la interacción entre dos o más componentes.

EJEMPLOS SIMPLES

- invocaciones a servicios
- colas de mensajes asincrónicos
- multicast de eventos
- pipes que representan streams de datos asincrónicos que preservan el orden

EJEMPLOS MAS COMPLEJOS

- canales de comunicación orientados a transacciones entre un servidor de base de datos y un cliente.
- Enterprise Service Bus (ESB) que media en la interacción entre un conjunto de proveedores y consumidores de servicios

CONECTORES – ROLES

Los conectores tienen roles.

El rol define la forma en la cual un conector puede ser utilizado por los componentes para llevar adelante la interacción.

Ejemplos:

client-server → dos roles: *invoca-servicio* y *provee-servicio*

pipe → dos roles: *escribe* y lee

El rol define las expectativas de un participante en la interacción.

Ejemplo: el rol *invoca-servicio* podría requerir que el invocador inicialice la conexión antes de enviar un requerimiento.

CONECTORES – SEMANTICA

- La semántica de la interacción representada por un conector es documentada como una especificación de protocolo.

- Los conectores pueden representar formas complejas de interacción. Por ejemplo, lo que parece una llamada a procedimiento simple, podría incluir:
  - protocolos para el manejo de time-outs
  - manejo de errores
  - serialización de datos
  - localización del proveedor de servicios

- Los conectores complejos pueden descomponerse en componentes y conectores que describen la sub-arquitectura del conector. Ejemplo: conector de client-server con fail-over

- Si el principal propósito de un componente es mediar en la interacción entre un conjunto de componentes, podría ser definido como un conector.

VISTAS E INSTANCIAS – TIPOS

- Cuando distintos componentes o conectores de una vista comparten la misma forma y comportamiento, se podría definir un tipo específico para ellos.

- Definir tipos específicos de la aplicación de manera centralizada, le permite al lector tener un único lugar para los detalles de todos los componentes y conectores.

- Los tipos de componentes y conectores instanciados en una vista de componentes y conectores particular, deben ser explicados referenciando al estilo/patrón que los guía, o a través de un catálogo de tipos específicos a la aplicación definidos como parte de la arquitectura.

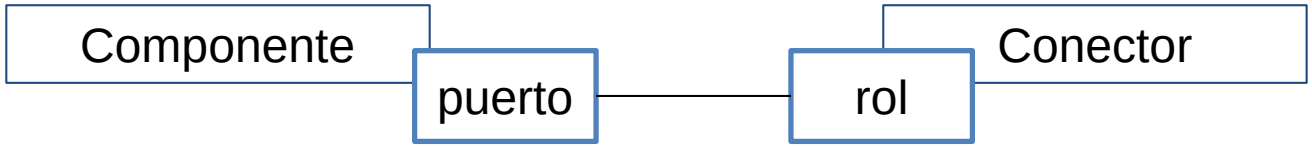
VISTAS E INSTANCIAS – TIPOS

- La definición de un tipo debería caracterizar la cantidad y tipo de sus interfaces (puertos y roles)
- El diagrama inicial de una vista de componentes y conectores muestra solamente instancias de componentes y conectores; no se deberían mostrar tipos.
- Cuando se mapean entre vistas, los módulos se mapean con tipos (no instancias) de componentes y conectores.

RELACIONES

1) *attachment (asociado)*

- indica qué conectores están asociados a qué componentes, definiendo al sistema como un grafo de componentes y conectores.
- específicamente, un *attachment* se especifica asociando un puerto de un componente a un rol del conector.
- un attachment válido se da cuando el protocolo del puerto del componente es consistente con el comportamiento esperado por el rol del conector.



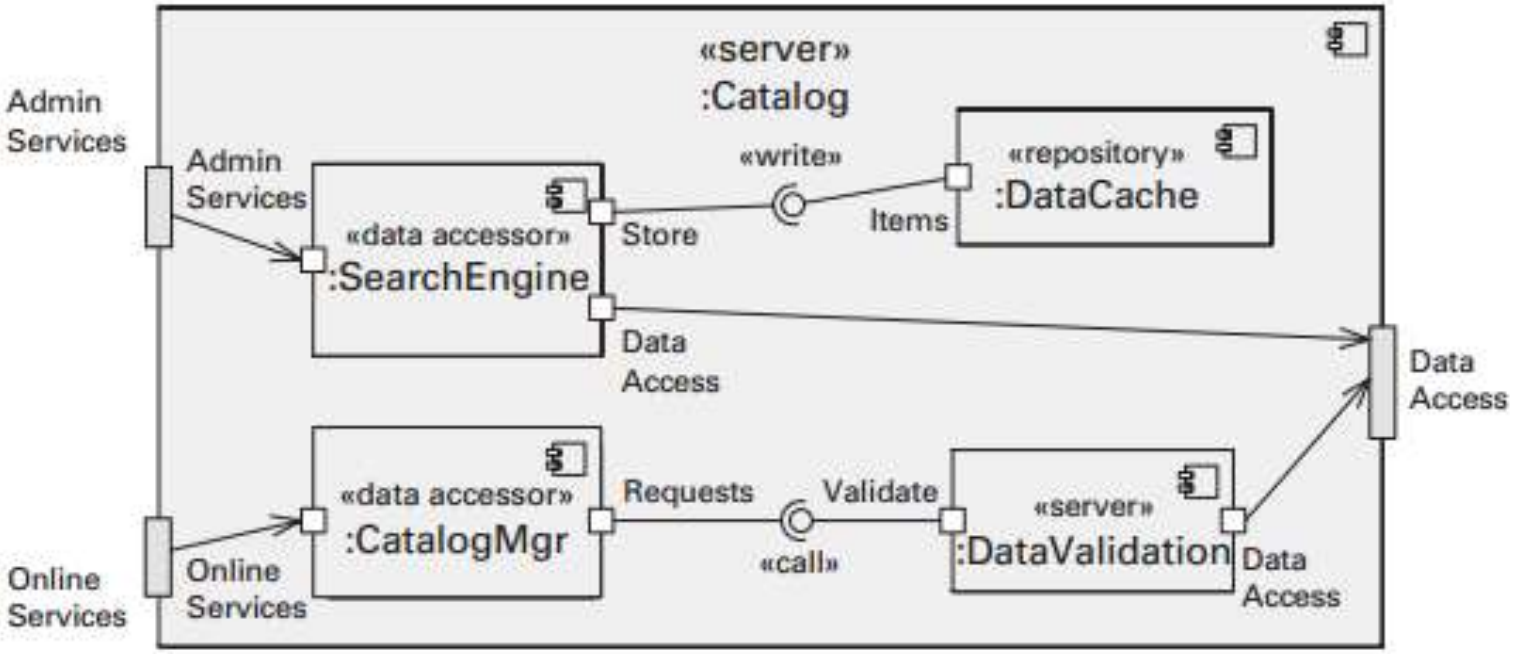
2) *delegacion-de-interface*

- cuando un componente o conector tiene una sub-arquitectura, es importante documentar la relación entre la estructura interna y las interfaces externas de aquel componente o conector. Esta es una *relación de interface*.

  - Componentes - mapean puertos internos a puertos externos
  - Conectores - mapean roles internos a roles externos

RELACIONES – EJEMPLOS

Ejemplo de delegacion de interface



El diagrama de componentes muestra la sub-arquitectura del componente `Catalog`. Los puertos externos de `Catalog` están asociados a puertos internos.

PROPIEDADES

Todos los elementos tendrán un nombre y tipo.

Propiedades adicionales dependen del tipo de componente o conector.

Las propiedades guían la implementación y configuración de los componentes y conectores.

El arquitecto también debe definir valores para las propiedades que soportan el análisis previsto para una vista de componentes y conectores particular.

Ejemplo - Si la vista se usa para análisis de performance, será valioso indicar latencias, capacidades de las colas, prioridades de los threads, etc.

PROPIEDADES - EJEMPLOS

CONFIABILIDAD	¿Cuál es la probabilidad de falla de un componente o conector?
PERFORMANCE	¿Qué tiempos de respuesta provee el componente bajo distinta carga? ¿Qué latencia y throughput se puede esperar para un conector?
REQUERIMIENTO DE RECURSOS	¿Cuáles son las necesidades de procesamiento y almacenamiento de un componente o conector?
FUNCIONALIDAD	¿Qué funciones realiza un elemento?
SEGURIDAD	¿Qué características de seguridad como encriptación, auditoría o autenticación provee o fuerza un componente o conector?
CONCURRENCIA	¿Qué proceso(s) o thread(s) ejecuta el componente?
CAPA	Para una topología en capas, ¿en qué capa reside el componente?

VISTA DE COMPONENTES Y CONECTORES – USOS 1

- mostrar a los desarrolladores y otros interesados cómo funciona el sistema:
  - ¿Cuáles son los principales componentes del sistema y cómo interactúan?
  - ¿Cuáles son los principales almacenamientos compartidos?
  - ¿Qué partes del sistema están replicadas? ¿Cuántas veces?
  - ¿Cómo fluyen los datos en el sistema?
  - ¿Qué partes del sistema corren en paralelo?
- guiar el desarrollo especificando la estructura y comportamiento de los elementos en ejecución
- razonar acerca de los atributos de calidad en tiempo de ejecución:
  - performance, confiabilidad, disponibilidad

VISTA DE COMPONENTES Y CONECTORES – USOS 2

Vistas de componentes y conectores bien documentadas permiten predecir las propiedades del sistema total, a partir de las estimaciones o mediciones de las propiedades de elementos e interacciones individuales.

Ejemplos:

- Para determinar el cumplimiento de los requerimientos de planificación en tiempo real, se necesita conocer el tiempo de ejecución de cada proceso componente.

- La confiabilidad de los elementos individuales y de los canales de comunicación ayudan al arquitecto para calcular la confiabilidad total del sistema.

RESTRICCIONES

- Los componentes pueden ser asociados solamente a conectores, no a otros componentes
- Los conectores pueden ser asociados solamente a componentes, no a otros conectores.
- *attachments* sólo pueden ser hechos solamente entre puertos y roles compatibles
- La delegación de interfaces puede ser definida solamente entre dos puertos compatibles (o dos roles compatibles)
- Los conectores no pueden aparecer en forma aislada. Un conector debe estar asociado a un componente.

NOTACION – 1

Notación Informal

- Diagramas de cajas y líneas
- Aunque pueden conducir a semánticas limitadas, los siguientes lineamientos pueden ayudar a obtener descripciones con rigor y profundidad:
  - asignar a cada tipo de componente y cada tipo de conector una forma visual distinta
  - especificar el significado de cada tipo de elemento
  - tener cuidado con el significado de los conectores, especialmente qué significa la dirección en los que usan flechas

NOTACION – 2

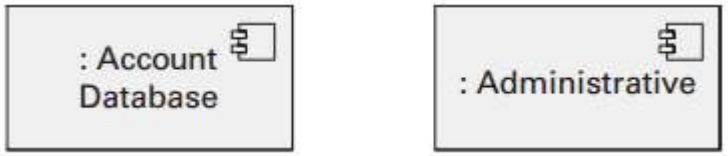
Notación Semiformal - UML

Componentes:

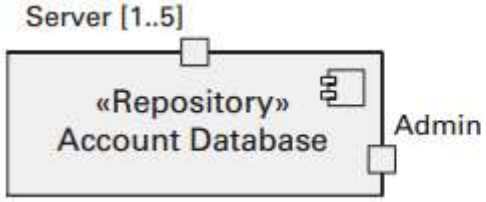
- Los nombres de las instancias contienen dos puntos
- Los nombres de los tipos NO contienen dos puntos

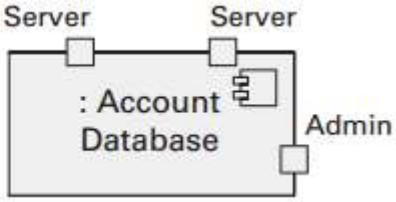
Instancias del mismo tipo

Instancias anónimas



- Se pueden usar estereotipos para relacionar un tipo específico de una vista con un tipo más genérico
- Los puertos pueden ser decorados con su multiplicidad, ya sea a nivel tipo o instancia



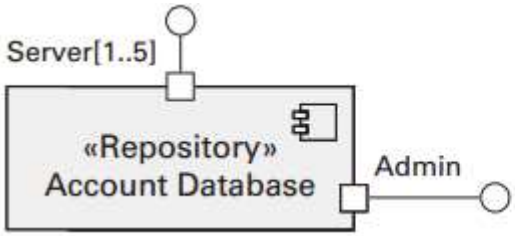


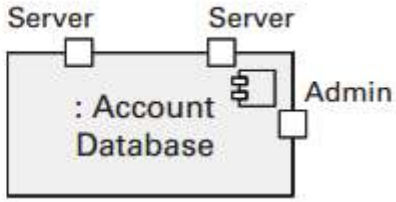
NOTACION – 3

Notación Semiformal - UML

Componentes:

- Interfaces provistas/requeridas – son aconsejable para los tipos, pero se deben evitar en instancias



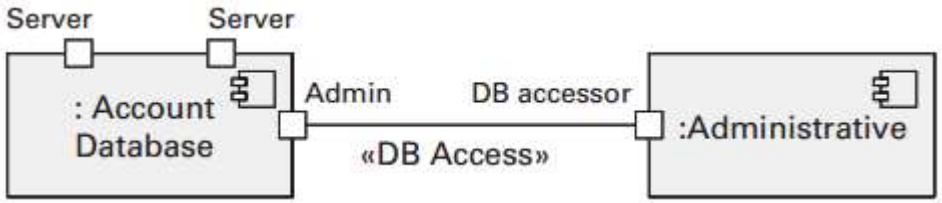


Notación Semiformal - UML

Conectores:

- a diferencia de los componentes, no son lo suficientemente ricos para especificar sub-estructuras, atributos o una descripción del comportamiento

- usar estereotipos para denotar el tipo del conector

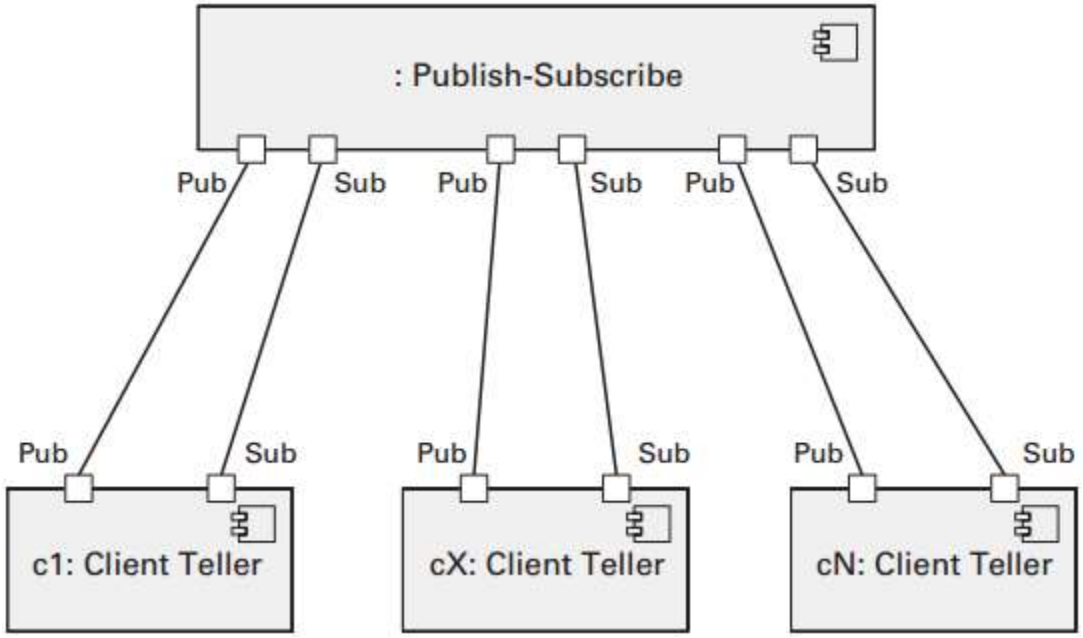


- si se necesita representar un conector “rico”, se puede usar una nota relacionada al conector, o incluir información adicional.

Notación Semiformal - UML

Conectores:

- en contadas ocasiones se puede usar un componente UML como conector



Notación Semiformal - UML

Desventajas:

- todos los tipos de componentes son mostrados con la misma forma y distinguidos a través de estereotipos
- no se puede tener una forma distinta para cada tipo de componente o conector

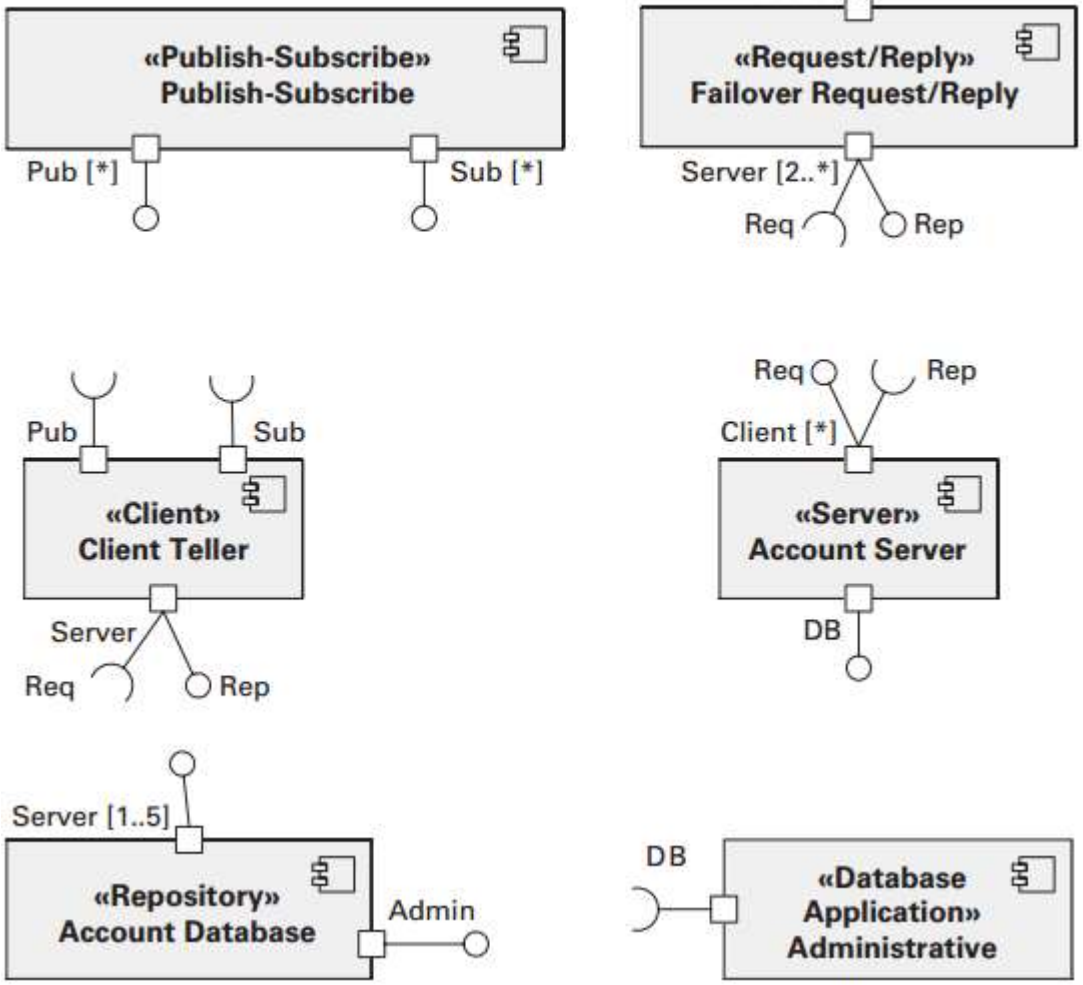
Notación Formal

- La mayoría de los ADLs puede ser usado para describir tipos de componentes y conectores, restricciones sobre la topología de sus grafos, y las propiedades asociadas a los elementos.

- Por ejemplo, algunas herramientas asociadas a ADLs puede determinar si un conjunto de procesos puede ser planificado tal que, dados los recursos de la CPU, cumplan con los deadlines de procesamiento.

NOTACION – EJEMPLO UML 1

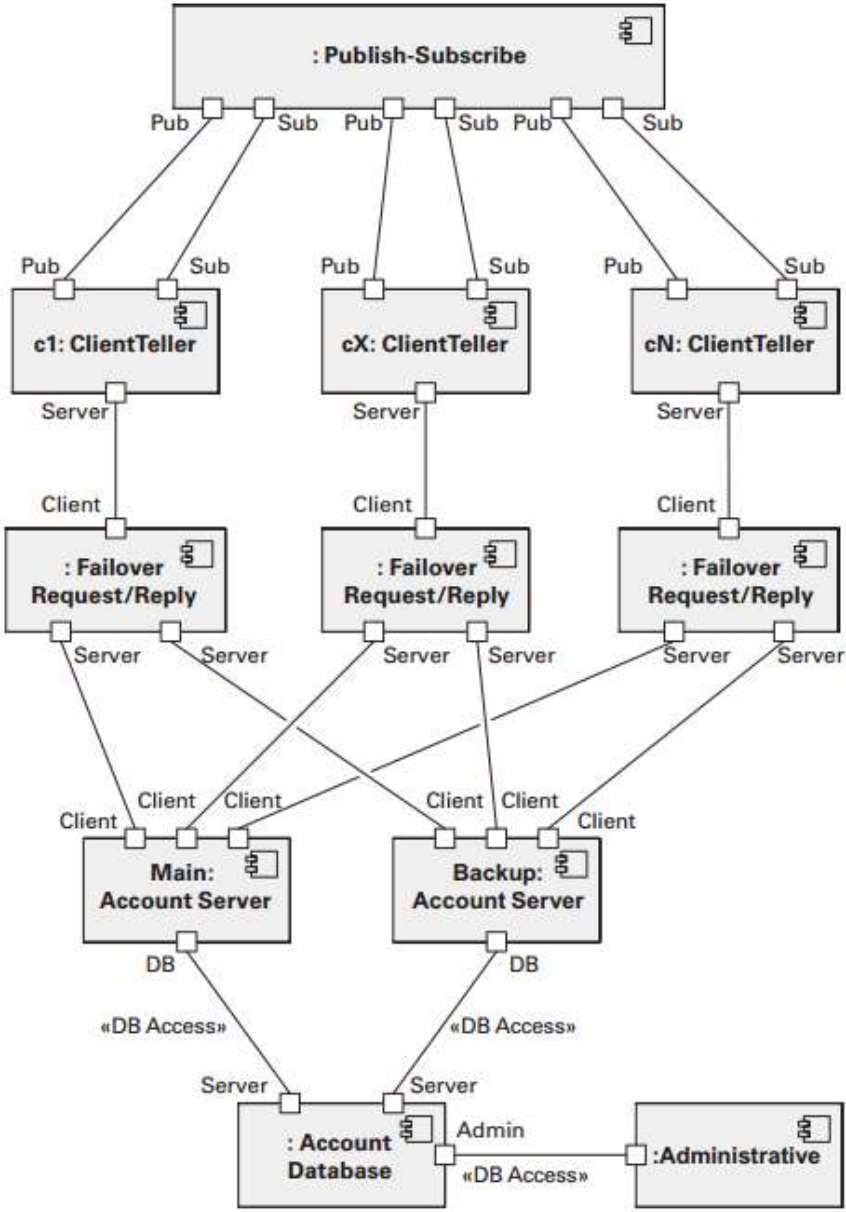
Catálogo de tipos de componentes y conectores



NOTACION – EJEMPLO UML 2

Vista Primaria

- Conector Publish-Subscriber como componente UML
- Conector Failover Request/Replay como componente UML
- Cantidad variable de clientes, con notación c1, c2… cN



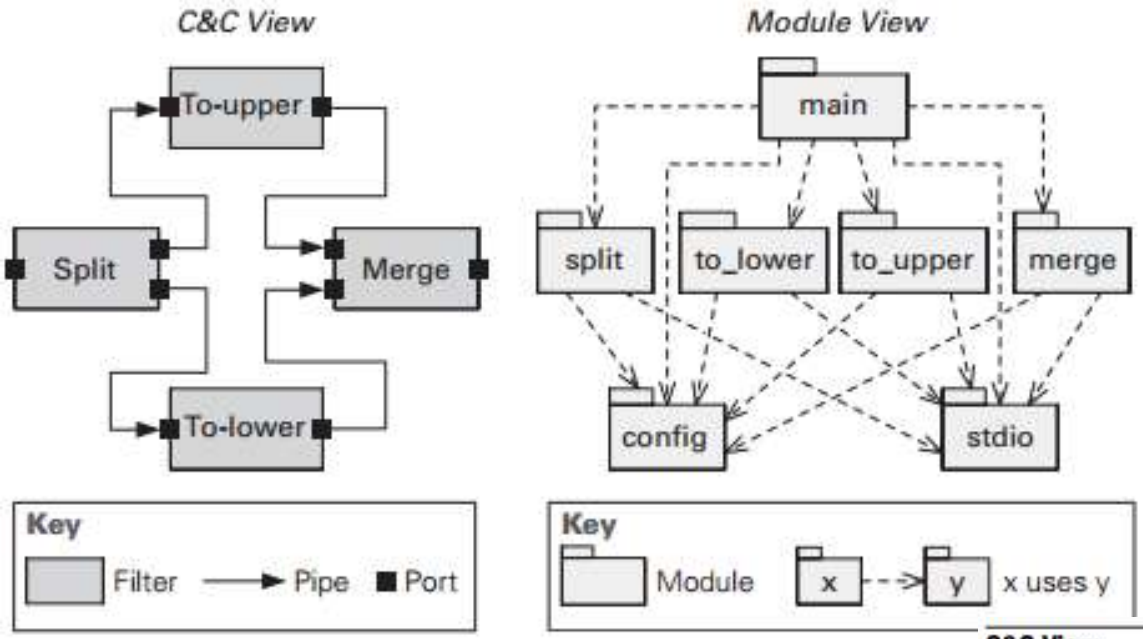
RELACION CON OTRAS VISTAS

Consideraciones al relacionar vistas de componentes y conectores con vistas de módulos

- El mismo módulo podría ser ejecutado por muchos de los elementos de una vista de componentes y conectores
- Un único componente de una vista de componentes y conectores podría ejecutar código definido por varios módulos
- Un componente de una vista de componentes y conectores podría tener muchos puntos de interacción con su ambiente, cada uno definido por la misma interfaz del módulo
- Como no es necesario mostrar todos los módulos en cada vista de módulos, un componente de una vista de componentes y conectores podría no mapear con ningún módulo de la vista de módulos.

RELACION CON OTRAS VISTAS – EJEMPLO

Pipe & Filter – Relación entre vista de componentes y conectores y vista de módulos

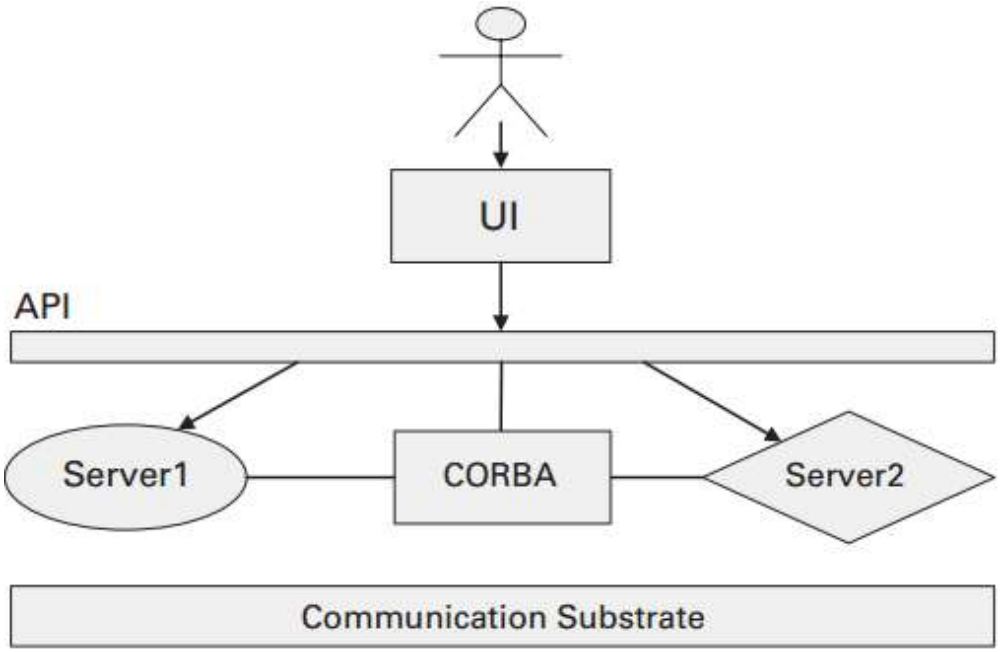


Mapeo entre vistas

C&C View	Module View
System as a whole	main
Split	split, config, stdio
To-lower	to_lower, config, stdio
To-upper	to_upper, config, stdio
Merge	merge, config, stdio
Each pipe	stdio

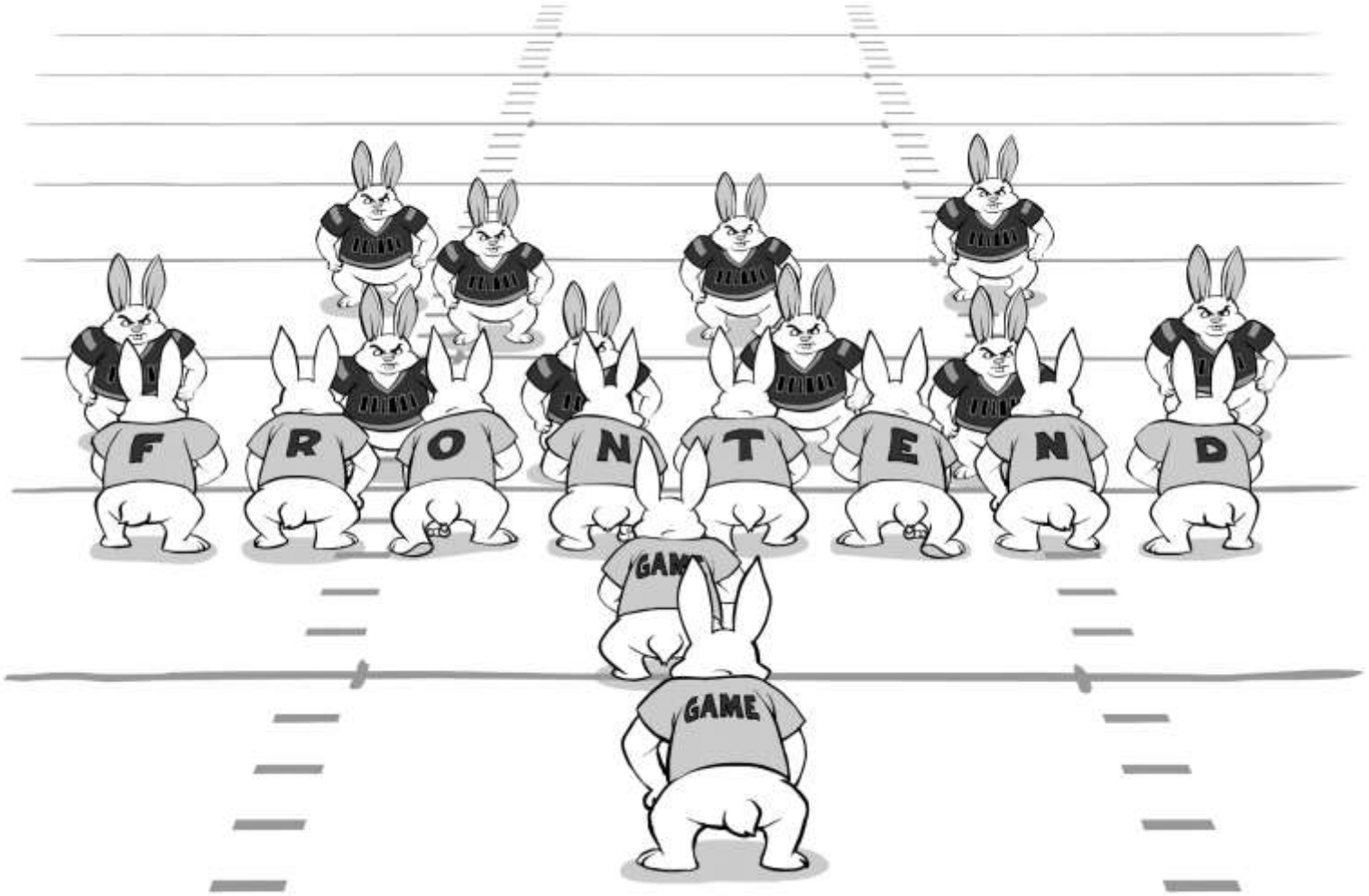
VISTA DE COMPONENTES Y CONECTORES - EJEMPLO

Ejemplo de mala documentación



- No especifica las referencias
- Representa una interfaz como un componente (asumiendo que API tienen el significado común de una interface)
- Usa diferentes formas para el mismo tipo de componente
- Usa la misma forma para diferentes tipos de componentes y conectores
- Confunde el contexto con el sistema a ser construido
- Las flechas no están explicadas
- Los componentes no tienen puertos

(EJEMPLO DE MALA ARQUITECTURA)



Ref: highscalability.com

CONTENIDO

VISTA DE ASIGNACION – INTRODUCCION 1

- Las vistas de asignación presentan un mapeo entre los elementos de software (ya sea de una vista de módulo o de componentes y conectores) y los elementos no-software:
  - hardware de computación y comunicación
  - sistemas de manejo de archivos
  - equipos de desarrollo

VISTA DE ASIGNACION – INTRODUCCION 2

- Qué permite el mapeo de la arquitectura de software con…

  - el hardware?

    Que la performance del sistema pueda ser analizada

  - la estructura de archivos?

    Que el sistema en producción se pueda gestionar

  - la estructura del equipo?

    Que puedan proceder las actividades de administración de proyecto

ELEMENTOS

1) Elementos de software - requieren propiedades del ambiente

2) Elementos del ambiente - proveen propiedades al software

RELACIONES Y RESTRICCIONES

RELACIONES

asignado-a (*alocado-a*) – un elemento de software es mapeado a un elemento del ambiente.

- un elemento de software puede mapearse a muchos elementos del ambiente
- muchos elementos de software pueden mapearse a un único elemento del ambiente

RESTRICCIONES

Varían dependiendo del estilo

ESTILOS DE LA VISTA DE ASIGNACION

- Estilo de Despliegue

  describe el mapeo entre los componentes y conectores de software y el hardware de la plataforma sobre el cual ejecuta el software

- Estilo de Instalación

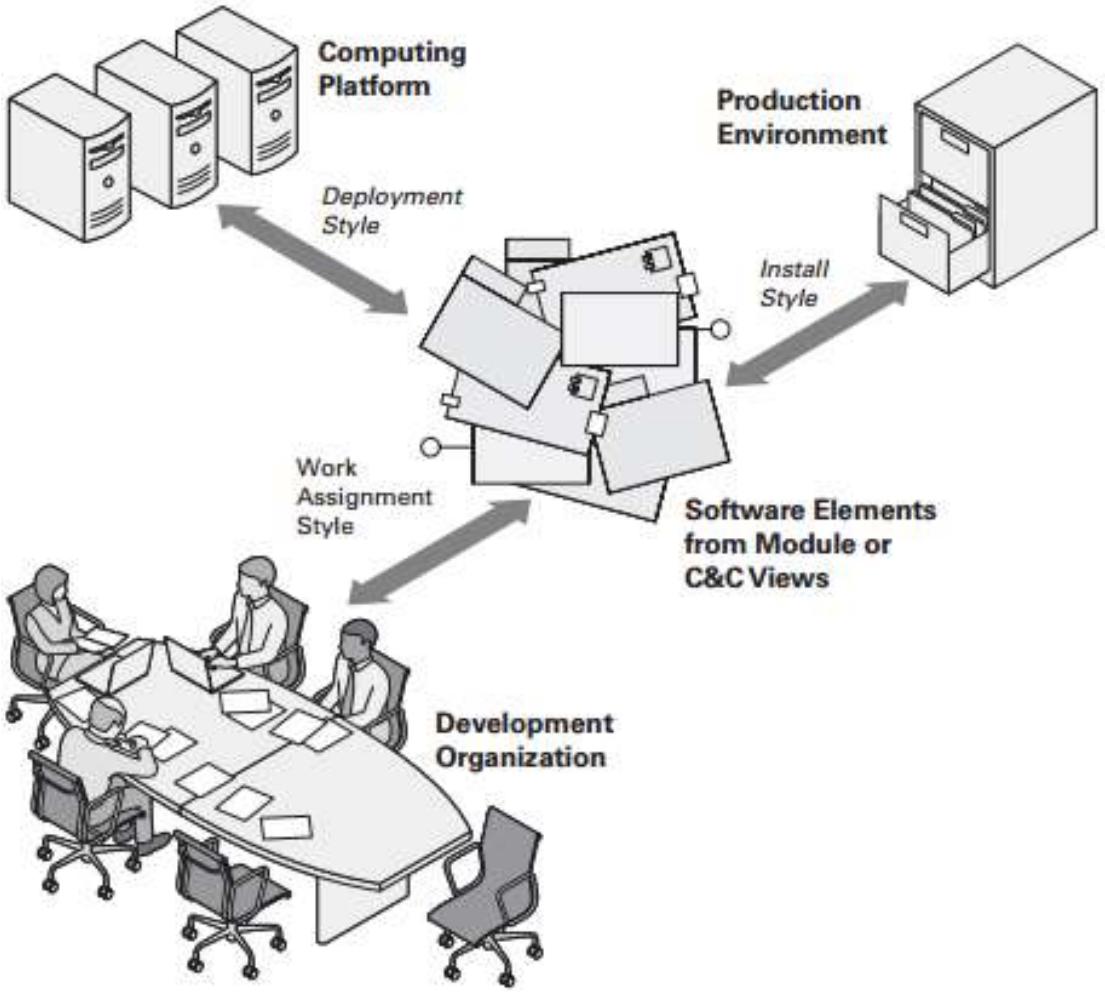
  describe el mapeo entre los componentes de software y las estructuras en el sistema de archivos del ambiente de producción

- Estilo de Asignación de Trabajo

  describe el mapeo entre los módulos de software y las personas, equipos o unidades de trabajo que tienen que desarrollar dichos módulos

ESTILOS DE LA VISTA DE ASIGNACION – EN UNA IMAGEN

Estilo de Despliegue



Estilo de Instalación

Estilo de Asignación de Trabajo

ESTILO DE DESPLIEGUE – DESCRIPCION Y ELEMENTOS

DESCRIPCION	Describe el mapeo de componentes y conectores en una arquitectura de software con el hardware de la plataforma de cómputo.
ELEMENTOS	○ Elemento de software - Son elementos de una vista de componentes y conectores. Las propiedades útiles para documentar son las características requeridas del hardware – e.g. capacidad de procesamiento, memoria, capacidad de recepción de requerimientos, tolerancia a fallos. ○ Elementos del ambiente - hardware de la plataforma de cómputo – e.g. procesador, memoria, disco, red (router, ancho de banda, firewalls), etc. Las propiedades útiles a considerar son aquellos aspectos del hardware que pueden influenciar la decisión de asignación

ESTILO DE DESPLIEGUE – RELACIONES

RELACIONES	○ *alocado-a* representa las unidades físicas donde el software residirá en ejecución. Propiedades: si la alocación puede cambiar en tiempo de ejecución o no. ○ *migra-a* indica que el elemento de software puede moverse de un procesador a otro, pero no puede existir simultáneamente en ambos procesadores. ○ *copia-migra-a*, similar a *migra-a*, excepto que el elemento de software manda una copia de sí mismo al nuevo procesador. ○ *ejecución-migra-a*, similar a la anteriores, a diferencia que una copia de un proceso existe en más de un procesador, pero solamente una está activa.

ESTILO DE DESPLIEGUE – RESTRICCIONES

RESTRICCIONES	La topología de asignación no tiene restricciones. Las propiedades requeridas del software tienen que ser satisfechas por las propiedades ofrecidas por el hardware.

ESTILO DE DESPLIEGUE – USOS

- analizar performance, disponibilidad, confiabilidad y seguridad
- entender las dependencias en ejecución (uso por los testeadores)
- facilitar la estimación de costos de hardware

ESTILO DE DESPLIEGUE – NOTACION

Notacion Informal

Se usan cajas, círculos, líneas y flechas a las cuales se les agregan imágenes generalmente representando elementos de hardware

Notación Semiformal – UML

Es un grafo de nodos conectados por asociaciones de comunicación.

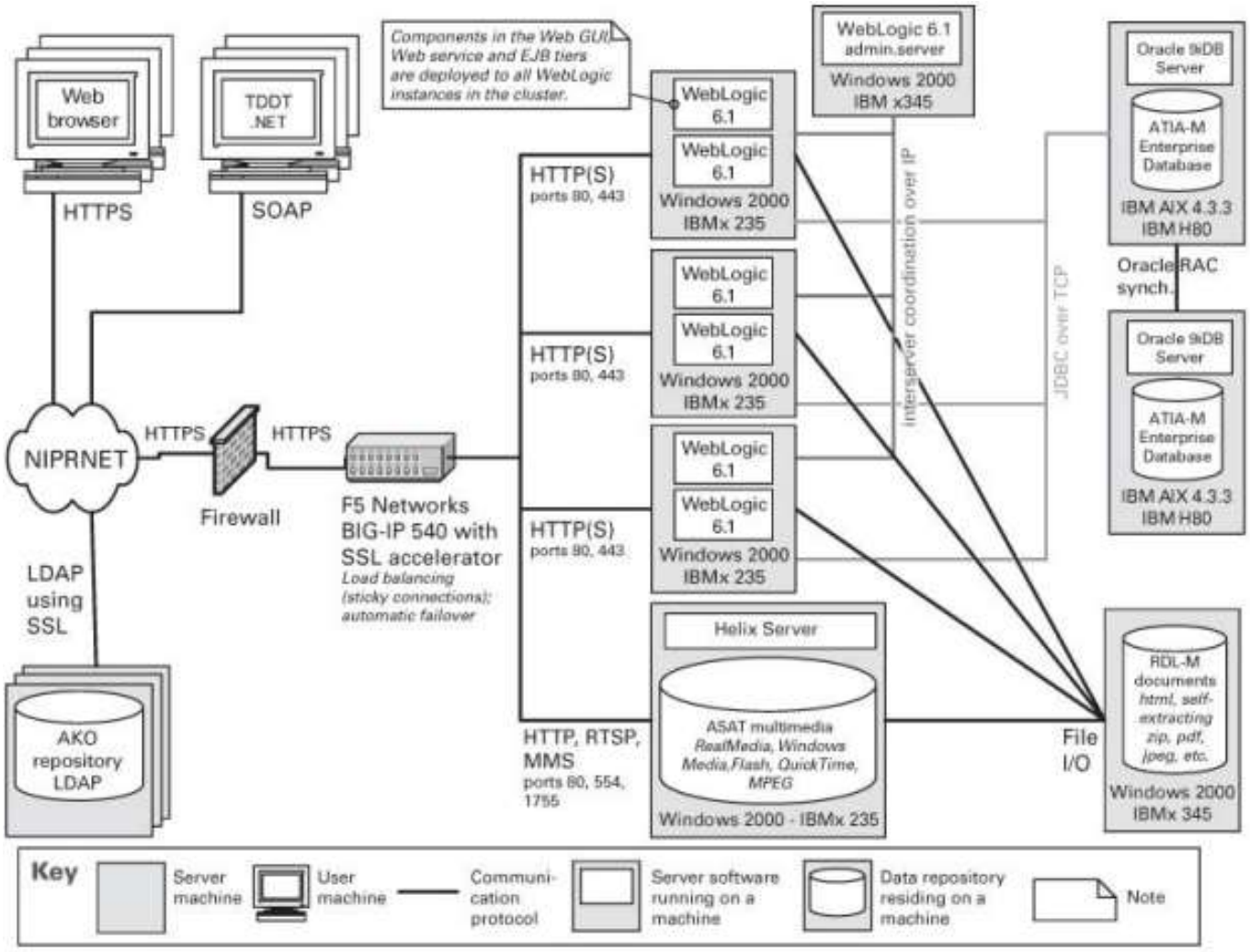
Notación Formal – UML

AADL y SysML son ejemplos de lenguajes de descripción de arquitecturas (ADL) que proveen notaciones formales para describir vistas de despliegue.

ESTILO DE DESPLIEGUE – NOTACION EJEMPLO 1

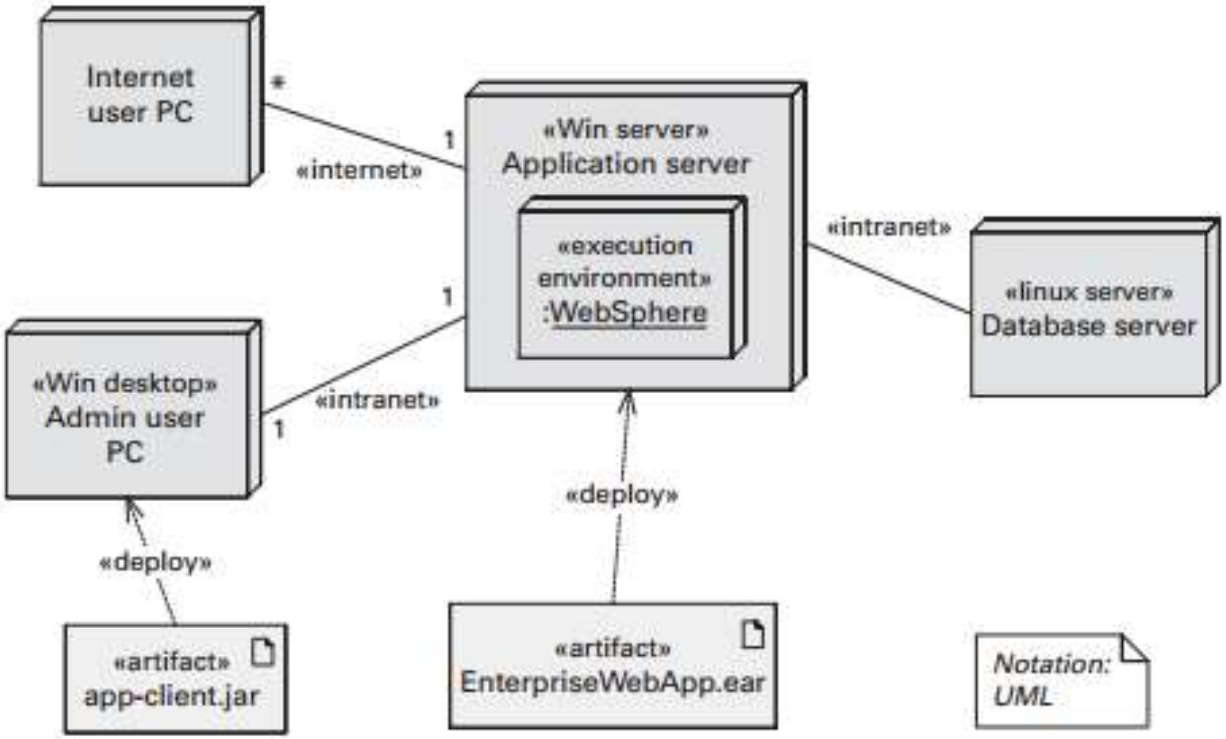
Notacion Informal

Sistema ATIA-M



ESTILO DE DESPLIEGUE – NOTACION EJEMPLO 2

Notacion Semiformal – UML



ESTILO DE INSTALACION – DESCRIPCION Y ELEMENTOS

DESCRIPCION	Describe el mapeo de componentes en una arquitectura de software con el sistema de archivos en el ambiente de producción.
ELEMENTOS	○ Elemento de software - Un componente de la vista de componentes y conectores. Las propiedades requeridas generalmente incluyen requerimientos sobre el ambiente de producción – e.g. soporte de Java o de un tipo de base de datos, permisos específicos sobre el sistema de archivos. ○ Elementos del ambiente - Un ítem de configuración, como un archivo o carpeta. ○ Las propiedades incluyen indicaciones de las características provistas por el ambiente de producción.

ESTILO DE INSTALACION – RELACIONES Y RESTRICCIONES

RELACIONES	○ *alocado-a* indica que un componente es ubicado en el ítem de configuración ○ *contenido-en* indica que un ítem de configuración está contenido en otro
RESTRICCIONES	Los archivos y carpetas están organizados en una estructura de árbol, siguiendo la relación está *contenido-en*

ESTILO DE INSTALACION – USOS

- crear procedimientos de build-and-deploy
- navegar a través de una gran cantidad de archivos y carpetas que constituyen el sistema instalado, y localizar archivos específicos (logs o configuración)
- seleccionar y configurar archivos para una versión específica
- identificar el propósito de un archivo faltante o dañado que trae problemas.
- diseñar e implementar características de actualización automática

ESTILO DE INSTALACION – TIPOS DE ARCHIVOS

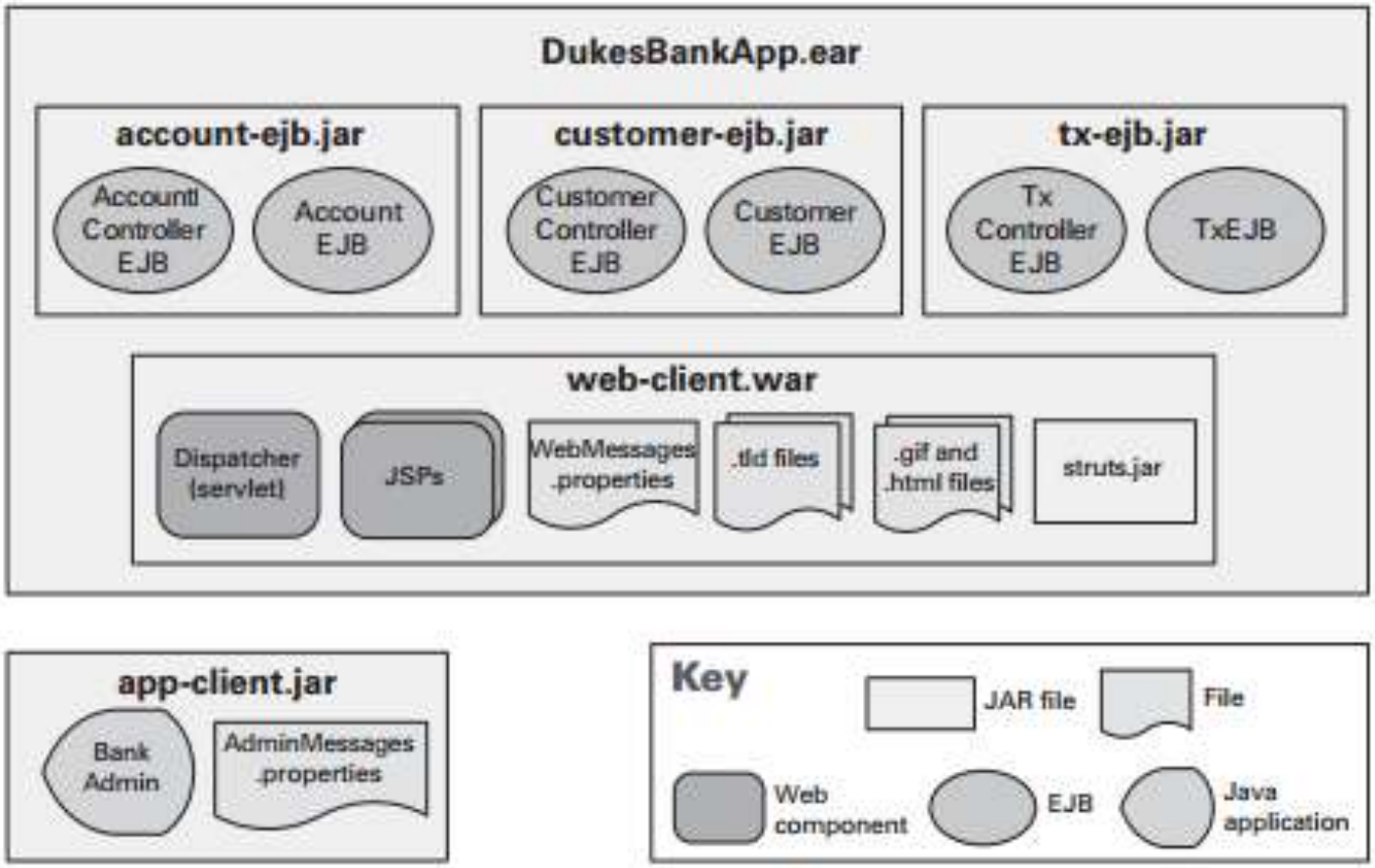
Tipos de archivos involucrados en una instalación

- ejecutables
- librerías
- datos
- logs
- archivos de control de configuración y versión
- licencias
- ayudas
- scripts
- contenido estático
- descriptores de despliegue

ESTILO DE INSTALACION – NOTACION 1

Notacion Informal

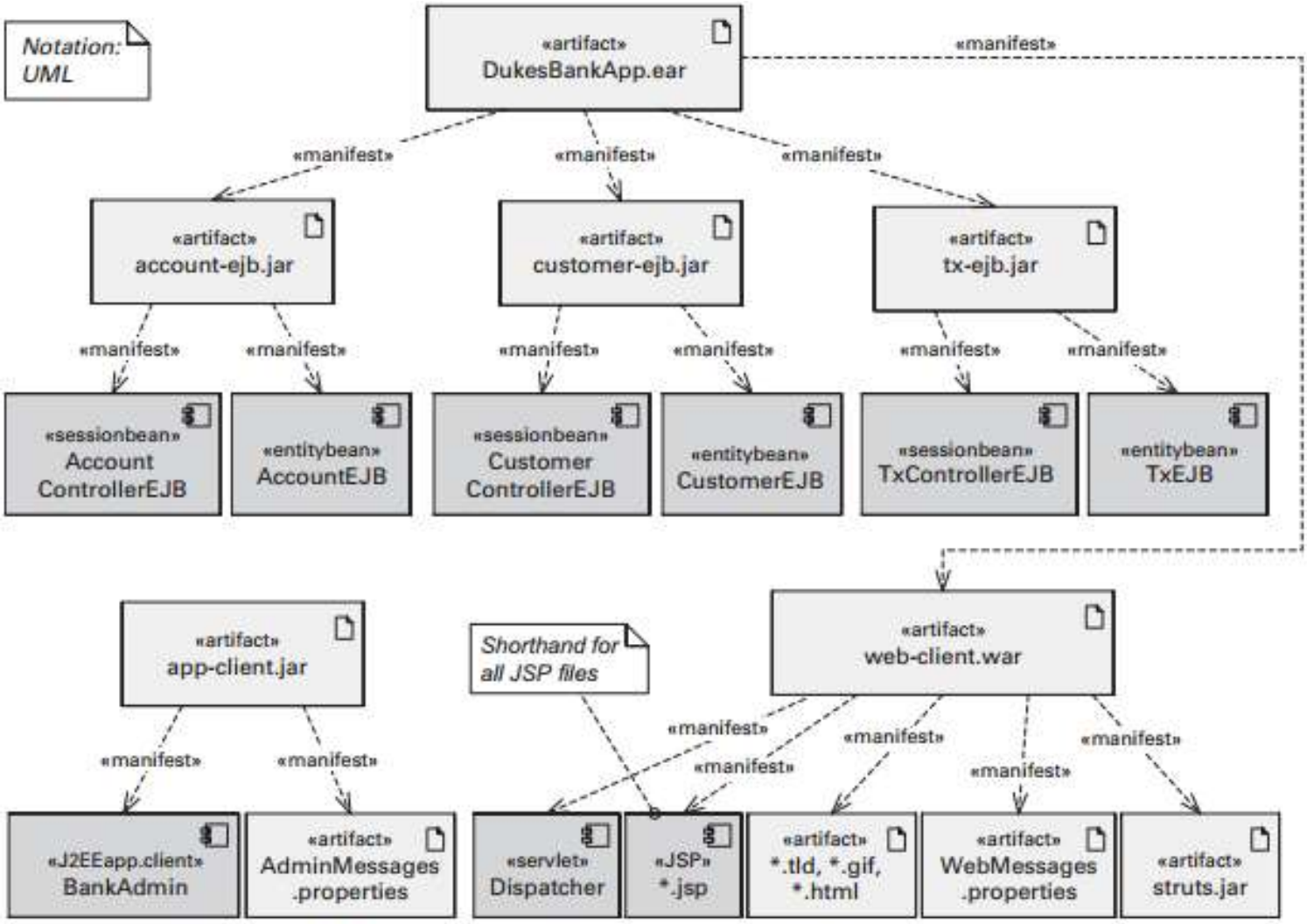
Sistema
Dukes Bank



ESTILO DE INSTALACION – NOTACION 2

Notacion Semiformal – UML

Sistema Dukes Bank



ESTILO DE ASIGNACION DE TRABAJO – DESCRIPCION Y ELEMENTOS

DESCRIPCION	Describe el mapeo de la arquitectura de software con los equipos en la organización de desarrollo.
ELEMENTOS	○ Elemento de software - Un módulo Las propiedades incluyen el perfil requerido y la capacidad disponible (esfuerzo y tiempo). ○ Elementos del ambiente - Una unidad organizacional, tal como una persona, un equipo, etc. Las propiedades incluyen el perfil provisto y la capacidad.

ESTILO DE ASIGNACION DE TRABAJO – RELACIONES Y RESTRICCIONES

RELACIONES	○ *alocado-a* representa que un elemento de software está asignado a una unidad organizacional ○ *contenido-en* indica que un ítem de configuración está contenido en otro
RESTRICCIONES	En general, no tiene restricciones

ESTILO DE ASIGNACION DE TRABAJO – USOS

- mostrar quién producirá cada unidad de software
- ayudar a planificar y gestionar la asignación de equipos

ESTILO DE ASIGNACION DE TRABAJO – NOTACION

No existen notaciones especiales

Sistema ECS (NASA)

ECS Element (Module)		
Segment	**Subsystem**	**Organizational Unit**
Science Data Processing Segment (SDPS)	Client	Science team
	Interoperability	Prime contractor team 1
	Ingest	Prime contractor team 2
	Data Management	Data team
	Data Processing	Data team
	Data Server	Data team
	Planning	Orbital vehicle team
Flight Operations Segment (FOS)	Planning and Scheduling	Orbital vehicle team
	Data Management	Database team
	User Interface	User interface team
. . .	. . .	. . .

CONTENIDO

SELECCIONAR LAS VISTAS

Determinar qué vistas producir, en qué momento y con cuánto detalle sólo es posible en el contexto de un proyecto.

Se debe conocer:

- Que personas están disponibles y con que perfil
- Que estándares hay que cumplir
- Con que presupuesto se dispone
- Cual es la planificación
- Cuales son las necesidades de información de los interesados importantes
- Cuales son los requerimientos de atributos de calidad más influyentes
- Cual es el tamaño del sistema

METODO PARA ELEGIR LAS VISTAS

Tres pasos:

1) Construir una tabla de interesados/vistas
2) Combinar vistas
3) Priorizar y construir

METODO PARA ELEGIR LAS VISTAS – PASO 1

1) Enumerar los interesados del proyecto en las filas
2) Enumerar las vistas que aplican al sistema en las columnas
3) En cada celda, indicar cuánta información requiere el interesado para la vista:
   - ninguna
   - alguna informacion
   - algunos detalles
   - mucho detalle
4) La lista candidata de vistas consiste de aquellas para las cuáles algunos interesados tienen un interés particular

	Module Views					C&C Views	Allocation Views				Other Documentation					
	Decomposition	Uses	Generalization	Layered	Data Model	Various	Deployment	Implementation	Install	Work Assignment	Interface Documentation	Context Diagrams	Mapping Between Views	Variability Guides	Analysis Results	Rationale and Constraints
Project managers	s	s		s			d			d		o				s
Members of development team	d	d	d	d	d	d	s	s	d		d	d	d	d		s
Testers and integrators	d	d	d	d	d	s	s	s	s		d	d	s	d		s
Designers of other systems					s						d	o				
Maintainers	d	d	d	d	d	d	s	s			d	d	d	d		d
Product-line application builders	d	d	s	o	s	s	s	s	s		s	d	s	d		s
Customers							o			o		o			s	
End users						s	s		o						s	
Analysts	d	d	s	d	d	s	d		s		d	d		s	d	s
Infrastructure support personnel	s	s			s	s	d	d	o					s		
New stakeholders	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Current and future architects	d	d	d	d	d	d	d	s	d	s	d	d	d	d	d	d
Key: d = detailed information, s = some details, o = overview information, x = anything																

METODO PARA ELEGIR LAS VISTAS – PASO 2

El Paso 1 probablemente generará una enorme lista de vistas. Este paso se encargará de reducir ese tamaño para que sea manejable.

- Buscar en la tabla aquellas vistas que sólo requieren un “alguna información” o que sirven a pocos interesados (vistas marginales)
- Determinar si los interesados podrían ser satisfechos por otra vistas que tenga interesados más fuertes.
- Determinar si hay vistas que puedan combinarse
- Considerar el costo de producir y mantener una vista

METODO PARA ELEGIR LAS VISTAS – PASO 3

El Paso 2 logra el conjunto mínimo de vistas necesario.

En el Paso 3, hay que decidir qué vista hacer primero.

La priorización depende el proyecto. Algunas recomendaciones:

- No todas las necesidades de información de todos los interesados deben ser satisfechas en su totalidad - proveer el 80% de la información requerida puede ser bastante trabajo y podría ser suficiente para que los interesados puedan hacer su trabajo. Verificar con ellos que el subconjunto sea suficiente.
- No es necesario completar una vista antes de comenzar otra. Algunos interesados pueden comenzar a trabajar con información de alto nivel.
- Resistir la tentación de relegar la documentación de justificación para “cuando haya tiempo”, ya que las justificaciones se pueden capturar mejor cuando están frescas.
- Una vista de descomposición es muy útil en etapas tempranas. La descomposición de alto nivel generalmente es simple de diseñar y provee información para construir los equipos de desarrollo, planificar capacitaciones necesarias, buscar componentes o módulos para comprar o reutilizar, comenzar a producir presupuestos y planificaciones

CONTENIDO

DOCUMENTAR UNA VISTA

5 secciones:

1) Presentación principal
2) Catalogo de elementos
3) Diagrama de contexto
4) Guía de variabilidad
5) Justificación

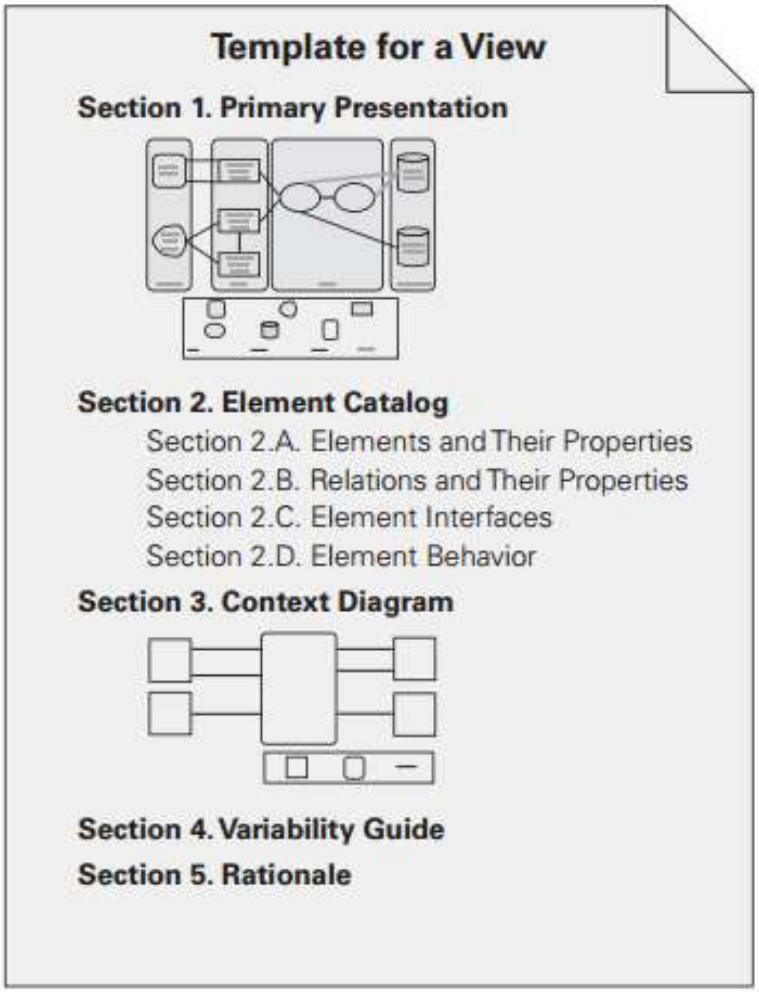
DOCUMENTAR UNA VISTA – SECCIONES 1 Y 2

SECCION 1 – PRESENTACION PRINCIPAL

- muestra los elementos y relaciones de la vista
- generalmente es un diagrama
- las referencias son muy valiosas!

SECCION 2 – CATALOGO DE ELEMENTOS

- detalla los elementos y sus relaciones mostrados en la presentación principal
- generalmente incluyen:
  - elementos y sus propiedades
  - relaciones y sus propiedades
  - interfaces de elementos
  - comportamiento de elementos



DOCUMENTAR UNA VISTA – SECCIONES 3 - 5

SECCION 3 – DIAGRAMA DE CONTEXTO

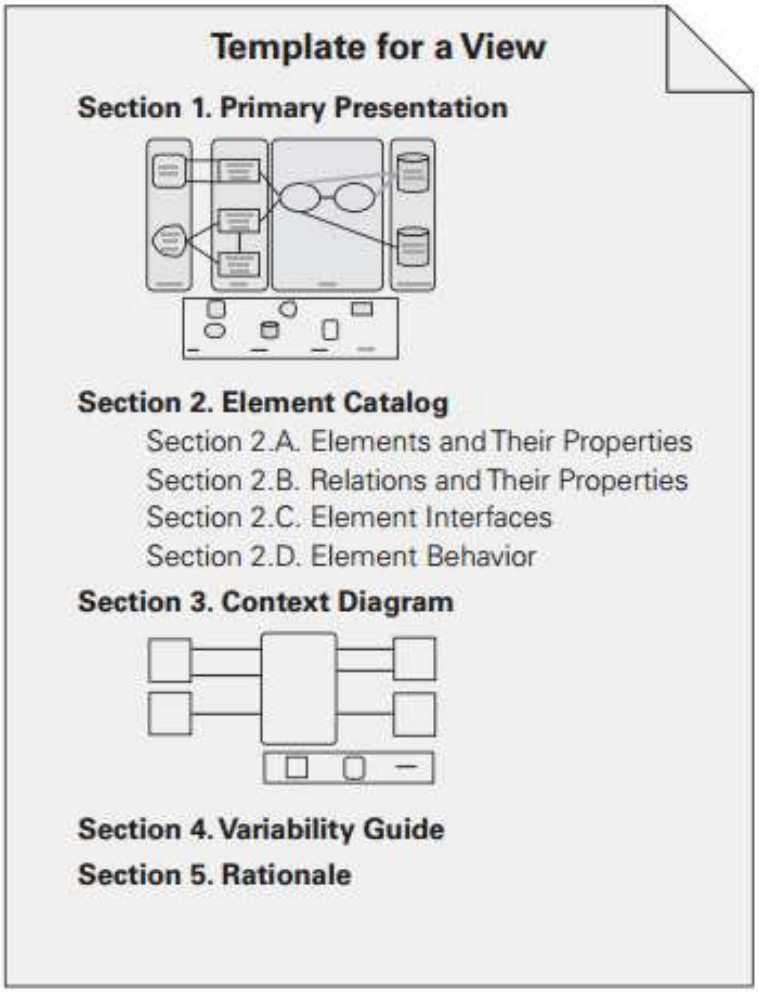
- Muestra como el sistema (o parte de él) se relaciona con su ambiente

SECCION 4 – GUIA DE VARIABILIDAD

- muestra como ejercitar los puntos de variación que son parte de la arquitectura

SECCION 5 – JUSTIFICACION

- explica el por que del diseño de la vista



DOCUMENTACION ADICIONAL

La documentación adicional a las vistas puede clasificarse en:

1) Información sobre la documentación de arquitectura – como la documentación esta diseñada y organizada para que los interesados puedan fácil y rápidamente encontrar lo que necesitan

2) Información sobre la arquitectura – toda otra información de utilidad adicional a las vistas – e.g. propósito del sistema, la forma en que las vistas están relacionadas, justificaciones de diseño, glosario, lista de acrónimos, etc.

DOCUMENTACION ADICIONAL – EN UNA IMAGEN

Formato para documentar informacion adicional

Información sobre la documentación de la arquitectura

Información sobre la arquitectura

Formato para Documentación Adicional

Sección 1. Hoja de ruta de la documentación
Sección 2. Como se documenta una vista

Sección 3. Descripción del sistema
Sección 4. Mapeo entre vistas
Sección 5. Justificación
Sección 6. Directorio – índice, glosario, lista de acrónimos

DOCUMENTACION ADICIONAL – SECCIONES 1 – 3

SECCION 1 – HOJA DE RUTA DE LA DOCUMENTACION

- indica qué información está en la documentación y dónde se la puede encontrar
- consiste de 4 secciones:
  - alcance y resumen
  - cómo está organizada la documentación
  - descripción de las vistas
  - cómo los interesados pueden usar la documentación

SECCION 2 – COMO SE DOCUMENTA UNA VISTA

- indica cómo está organizada la información de una vista

SECCION 3 – DESCRIPCION DEL SISTEMA

- consiste de una descripción (en prosa) de distintas características del sistema - funciones, usuarios, cualquier otra característica o restricción importante

- provee al lector con un modelo mental consistente del sistema y su propósito

SECCION 4 – MAPEO ENTRE VISTAS

- Ayuda al lector a entender las asociaciones entre vistas, proveyendo un entendimiento sobre cómo la arquitectura funciona como un todo.
- Formas de documentar el mapeo
  - establecer reglas de mapeo – e.g., por convenciones de nombres
  - a través de tablas de mapeo
  - gráficamente
- Para qué vistas se debe proveer un mapeo
  - Vista de descomposición y cada vista de componentes y conectores
  - Al menos entre una vista de módulos y una vista de componentes y conectores
  - Si se usa más de una vista de módulos, mapearlas entre ellas.

DOCUMENTACION ADICIONAL – SECCION 4 EJEMPLO

Element in C&C View X	Element in Module View Y
BankAdmin	com.sun.ebank.appclient com.sun.ebank.util stubs from com.sun.ebank.ejb
Web browser	—
WebUI	web com.sun.ebank.web com.sun.ebank.web stubs from com.sun.ebank.ejb
AccountControllerEJB	com.sun.ebank.ojb com.sun.ebank.util
AccountEJB	com.sun.ebank.ojb com.sun.ebank.util
...	...

DOCUMENTACION ADICIONAL – SECCIONES 5 – 6

SECCION 5 – JUSTIFICACION

- documenta las decisiones arquitectónicas que aplican a más de una vista
- incluye decisiones asociadas al sistema como un todo y sus causas – e.g. de background, restricciones organizacionales, requerimientos funcionales, etc.

SECCION 6 – DIRECTORIO

- incluye un conjunto de material de referencia que ayuda a los lectores a encontrar fácilmente más información.
- contiene:
  - índice de términos
  - glosario
  - lista de acrónimos

EMPAQUETAR LA DOCUMENTACION

Esquemas de empaquetado:

- Producir un unico documento de arquitectura

- Producir documentos separados
  - poner todas las vistas en un solo documento
  - poner cada vista en su propio documento
  - dividir por susbistema

REVISION DE UN DOCUMENTO DE ARQUITECTURA

- solamente los usuarios previstos de un documento serán capaces de decir si contiene la información correcta presentada de la manera correcta
- antes de liberar una versión de la documentación de arquitectura, hacerla revisar por los representantes de las comunidades a las cuales va dirigida

REVISION – PROCESO PASOS 1 - 3

- Paso 1: Establecer el propósito de la revisión
  - Por qué, cuándo y quién
- Paso 2: Establecer el sujeto de revisión
  - Identificar los tipos de artefactos, sus versiones, sus orígenes y su grado de completitud que son necesarios para realizar la revisión
- Paso 3: Construir o adaptar el conjunto de preguntas apropiado
  - Identificar las preguntas que se realizarán durante la revisión

REVISION – PROCESO PASOS 4 – 6

- Paso 4: Planificar los detalles de la revisión
  - Planificar fecha y hora de la reunión
  - Identificar los participantes de la revisión
  - Manejar la logística de la revisión

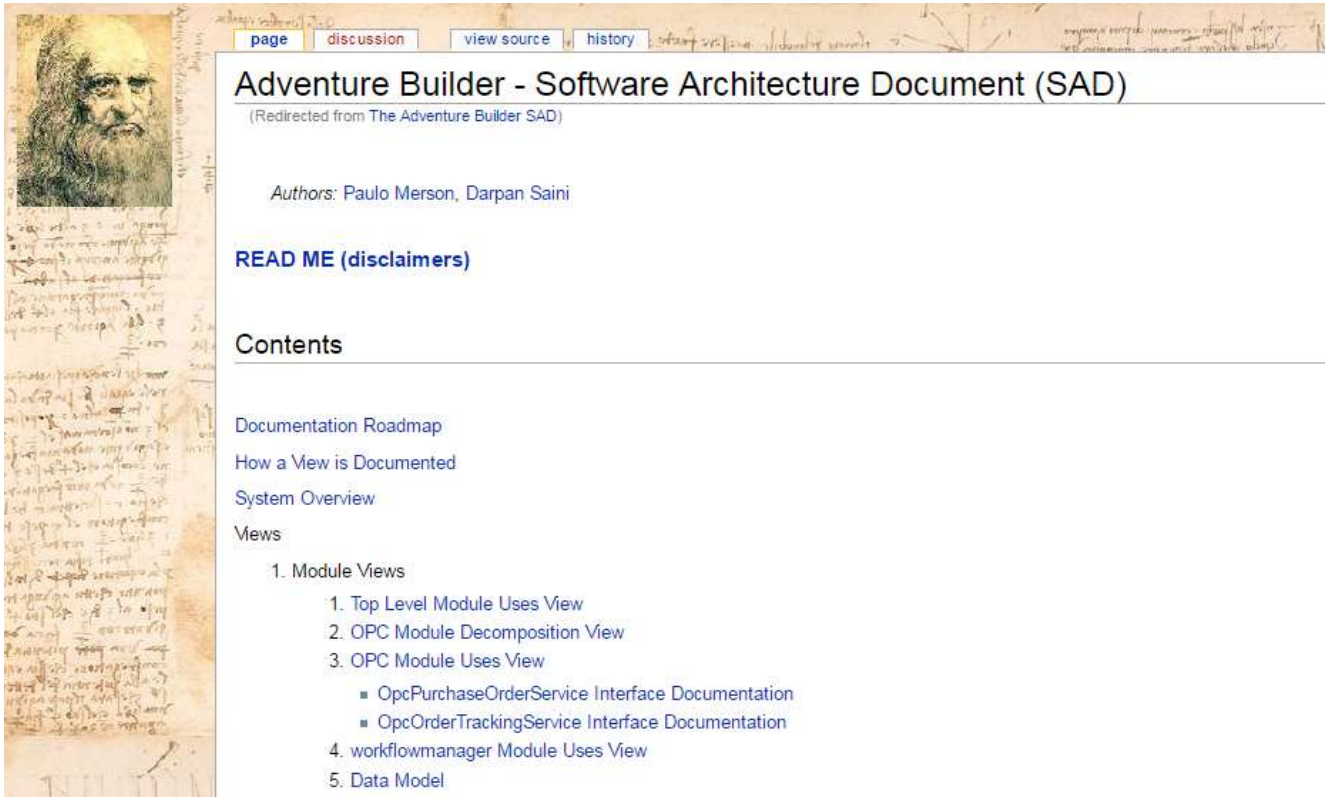
- Paso 5: Realizar la revisión
  - Realizar las preguntas a los interesados involucrados en la revisión y capturar sus respuestas

- Paso 6: Analizar y resumir los resultados
  - Procesar las respuestas a las preguntas para hacer una determinación cualitativa del impacto de la visión de los interesados sobre la arquitectura del sistema

EJEMPLO DE DOCUMENTACION DE ARQUITECTURA

Adventure Builder - Software Architecture Document (SAD)

https://wiki.sei.cmu.edu/sad/index.php/The Adventure Builder SAD



page | discussion | view source | history

Adventure Builder - Software Architecture Document (SAD)

(Redirected from The Adventure Builder SAD)

Authors: Paulo Merson, Darpan Saini

READ ME (disclaimers)

Contents

BIBLIOGRAFIA

Documenting Software Architectures
Clements, Bachmann, Bass, Garlan, Ivers, Little, Merson, Nord, Stafford
Addison-Wesley, 2011

Elsa Estevez
ece@cs.uns.edu.ar