Trabajo Práctico 3: Arquitectura de Software

Ejercicio 1

Responda las siguientes preguntas:

a) ¿Cuál es la diferencia entre el diseño arquitectónico y el diseño a nivel de componentes como el que nos proveen los patrones GOF?
b) ¿Cuáles son las diferencias entre un componente y un conector?
c) ¿Deberían los ingenieros de software aceptar la degradación arquitectónica? ¿Cuáles son los peligros de esto?
d) ¿Por qué los arquitectos de software deben tener en cuenta las necesidades de los clientes? Dé un ejemplo concreto.
e) Considere que un ingeniero de software comienza a construir programando directamente, ¿cuál sería una analogía a esto en la construcción de edificios? ¿Históricamente se construyeron edificios de esta forma? ¿Qué sucedía?

Ejercicio 2

a) Mencione al menos dos decisiones de diseño al nivel arquitectónico del sistema SIU Guaraní.
b) Mencione al menos dos decisiones de diseño al nivel arquitectónico de Mercado Libre.

Ejercicio 3

Identificar los componentes y conectores de la arquitectura de software de cada uno de los ejemplos que siguen:

a) Un sistema de gestión de versiones centralizado.
b) Un sistema de home banking que un banco nacional desea proveer a sus clientes para
c) que puedan acceder a sus cuentas bancarias, controlar sus saldos y realizar transferencias.
d) Un programa que ustedes hayan realizado por medio del lenguaje Pascal (explique en qué consistía el programa y luego identifique componentes y conectores).
e) Un programa que haya escrito en el Linux shell (explique en qué consistía el programa y luego identifique componentes y conectores).

Ejercicio 4

Defina los conceptos de *patrón arquitectónico*, *estilo arquitectónico* y *arquitectura específica para un dominio*. Analice las similitudes y diferencias entre los mismos. ¿Cuáles son los beneficios de usar un estilo arquitectónico?

Nota: Para la resolución de los ejercicios que siguen a continuación sólo considere los estilos arquitectónicos: *Basado en Capas, Cliente-Servidor, Pipes and Filters, Basado en Eventos, Peer-to-Peer.*

Ejercicio 5

Para cada una de las siguientes situaciones, discuta qué estilo o estilos arquitectónicos resultan aplicables:

a) La aplicación necesita procesar la entrada en forma de flujo continuo (*stream*) y producir una salida también en forma de flujo continuo.

b) Hay una gran cantidad de usuarios que se conectarán al sistema a través de una red.

c) La aplicación necesita ser ejecutada en diferentes plataformas.

d) El sistema debe tener una alta performance tanto en tiempo como en espacio (memoria).

e) El sistema debe ser capaz de ser ajustado para alcanzar el mejor uso de los recursos de software y hardware, los cuales pueden ser cambiados en el futuro.

f) Una componente clave del sistema es la solución de un problema recurrente en el dominio de aplicación. Se desea reusar la componente en sistemas similares.

Ejercicio 6

Para cada uno de los problemas descriptos a continuación, modele el estilo arquitectónico solicitado. Para cada caso analice las ventajas y desventajas del modelo, e indique si podría utilizar otro estilo arquitectónico (o combinación de estilos). En caso afirmativo, justifique adecuadamente su decisión y realice además el modelo alternativo.

6.1: Tubos y filtros

Se desea contar con un sistema capaz de mejorar la calidad del sonido de escuchas telefónicas, grabación a distancia de conversaciones, etc. El sonido será digitalizado y a partir de allí se lo someterá a diversas transformaciones para mejorar su calidad. Las transformaciones posibles son:

- Dividir los sonidos según su frecuencia, lo que produce varias bandas de sonido diferentes que pueden ser analizadas separadamente o vueltas a reunir.
- Disminuir la velocidad con que se reproduce el sonido de una cantidad fijada dinámicamente por el usuario.
- Eliminar todos los sonidos de una frecuencia dada por el usuario.
- Aumentar y disminuir el volumen en una cantidad fijada por el usuario.

El usuario puede determinar en cualquier paso si desea escuchar el resultado. Cada vez que se realiza una mejora a la calidad del sonido, es el usuario quien decide qué operación de las disponibles desea realizar.

6.2: Sistemas en capas o estratificados

Los pacientes de una guardia de terapia intensiva son monitoreados por dispositivos electrónicos análogos adosados a sus cuerpos. Los dispositivos miden los signos vitales de los pacientes. Hay un sensor para el ritmo cardíaco (activo, una señal para cada latido del corazón), presión sanguínea (pasivo) y la temperatura (pasivo). Se necesita un programa que lea los signos vitales con una frecuencia especificada para cada paciente. Los valores leídos serán comparados con rangos que serán reportadas con mensajes de alarma en el monitor del box de enfermería. También debe mostrarse un mensaje apropiado si falla un dispositivo. Además, se requiere almacenar los datos y proveer reportes estadísticos.

6.3: Sistemas basados en eventos

Debido al gran éxito de la película “El código Da Vinci”, mayor cantidad de personas se ha acercado al Museo del Louvre para conocer las obras de Leonardo Da Vinci, en especial, “La Mona Lisa”. Esta obra ha sido cambiada a una sala más importante para que pueda ser mejor apreciada por el público. Entre las tantas medidas de seguridad tomadas, se instaló además un importante sistema de iluminación en forma automática de manera tal de mantener la luz dentro de cierto rango que permite a los visitantes apreciar la obra siempre correctamente iluminada y al mismo tiempo no provocarle daños serios. La sala cuenta con ventanas e iluminación artificial. Cerca de la obra de arte se instaló un sensor que mide la cantidad de luz que le llega a la obra. El sistema de software de los sensores deberá mantener la cantidad de luz sobre la obra dentro de ciertos límites pre-establecidos. Éstos son modificados automáticamente de manera tal de regular su intensidad, aumentándola o disminuyéndola de manera discreta según haga falta. El sistema deberá optar primero por utilizar la luz proveniente de las ventanas, y si esta no es suficiente deberá hacer uso de la iluminación artificial. El sensor no puede “manejar” la luz natural que le llega a la obra aunque sea demasiada, pero sí puede detectar cuándo le hace falta utilizar la luz artificial en función de los límites ingresados por un especialista en el sistema de software del museo.

Ejercicio 7

El vector de frecuencia de palabras clave (VFPC) de un archivo de texto es una secuencia de pares de palabras clave y la cantidad de veces que aparece en el texto. Es una buena representación de los contenidos de un texto; los VFPC son muy utilizados en búsqueda y recuperación de información (*information search and retrieval*). Por ejemplo, a continuación exponemos el vector de frecuencia de palabras clave de este párrafo (para palabras que aparecen al menos dos veces).

Palabra	Frecuencia
palabras	4
texto	3
clave	3
vector	2
VFPC	2
frecuencia	2

Este vector indica claramente que el texto se refiere a palabras clave, textos, y vectores. Cuando un usuario busca información acerca de "vector de frecuencia de palabras clave" (en un buscador como Google, Bing, o Yahoo!), este párrafo se asocia claramente con la consulta (*query*) del usuario porque las palabras de la consulta tienen una frecuencia alta en el texto. Otra ventaja de los VFPC es que resume al texto y por lo tanto ocupa menos espacio. Esta técnica funciona mejor cuando se ignoran las palabras cortas (como "el", "de", "los", etc.), como se hizo en el ejemplo. Además, las palabras con la misma raíz deben ser tratadas como una; por ejemplo, "vectores" y "vector" deben ser tratadas como una única palabra.

a) Diseñe la arquitectura de un sistema para calcular VFPCs, proponiendo dos alternativas que utilicen estilos arquitectónicos diferentes.
b) Para cada uno de los diseños propuestos, evalúe su desempeño con respecto a los siguientes atributos:
    i. *Modificabilidad*: qué tan difícil es modificar al sistema luego de su implementación para cumplir con nuevos requerimientos. Considere cambios en el uso del sistema (por ejemplo, que el VFPC se pueda computar en forma incremental a medida que se lee el texto), cambios en la representación de los datos de entrada/salida, y cambios en las funciones del sistema (por ejemplo, modificar al sistema para que trate a los sinónimos como una sola palabra, que el sistema sea interactivo, o que el usuario pueda agregar o borrar palabras en el texto original).
    ii. *Performance*: el desempeño del sistema en cuanto al uso de tiempo y memoria que requiere su ejecución.
    iii. *Reusabilidad*: qué tan reusables son sus componentes.

Ejercicio 8

Las conferencias de profesionales son organizadas a fin de anunciar y discutir resultados en un campo científico dado. La actividad central de organizar una conferencia se centra en seleccionar los trabajos (de ahora en más llamados *papers*) a ser presentados. Generalmente, se realiza primero una invitación abierta convocando el envío de trabajos, luego se circulan los trabajos recibidos entre un grupo de revisores predeterminado, y en base a sus evaluaciones se seleccionan los mejores trabajos para incluir en el programa de la conferencia. El sistema de soporte para la organización, que estará disponible por la web, debe considerar los siguientes aspectos:

- Se anuncia el "*call for papers*".
- Los autores contribuyen con papers basados en su trabajo de investigación. Los papers son recibidos por un administrador.
- Se registran los papers recibidos.
- En una fecha determinada, se distribuyen los papers entre el panel de revisores. Cada paper se envía a tres revisores.
- Se recolectan los informes de los revisores sobre los papers.
- En una fecha dada, se selecciona un conjunto de papers para ser incluidos en el programa de la conferencia. De acuerdo a lo indicado por los revisores, algunos de estos papers pueden requerir ciertos cambios por parte de los autores.
- Se avisa a los autores de los resultados del proceso de evaluación. Una vez recibidas las versiones finales de los papers, se construye el programa final.

Ejercicio 9

La construcción de un compilador involucra la división del proceso en una serie de fases que variará con su complejidad. Generalmente estas fases se agrupan en dos tareas: el análisis del programa fuente y la síntesis del programa objeto.

Análisis: Se trata de la comprobación de la corrección del programa fuente, e incluye las fases correspondientes al *análisis léxico* (que consiste en la descomposición del programa fuente en componentes léxicos), *análisis sintáctico* (agrupación de los componentes léxicos en frases gramaticales) y *análisis semántico* (comprobación de la validez semántica de las sentencias aceptadas en la fase de análisis sintáctico).

Síntesis: Su objetivo es la generación de la salida expresada en el lenguaje objeto y suele estar formado por una o varias combinaciones de fases de *generación de código* (normalmente se trata de código intermedio o de código objeto) y de *optimización de código* (en las que se busca obtener un código lo más eficiente posible).

Desarrolle un modelo arquitectónico para un nuevo compilador del lenguaje Pascal que se pretende realizar de manera que sea factible compilar código para diferentes plataformas. Especifique qué estilo y/o mezcla de estilos usaría. Considere que el compilador debe

proveer también una GUI para facilitar el uso del mismo, que permita editar y compilar los programas.

Ejercicio 10

Considere el sistema del ejercicio anterior; alternativamente, las fases descriptas para las tareas de análisis y síntesis se pueden agrupar en *front end* y *back end*:

Front end: es la parte que analiza el código fuente, comprueba su validez, genera el árbol de derivación y rellena los valores de la tabla de símbolos. Esta parte suele ser independiente de la plataforma o sistema para el cual se vaya a compilar, y está compuesta por las fases comprendidas entre el análisis léxico y la generación de código intermedio.

Back end: es la parte que genera el código máquina, específico de una plataforma, a partir de los resultados de la fase de análisis, realizada por el front end.

Esta división permite que el mismo back end se utilice para generar el código máquina de varios lenguajes de programación distintos y que el mismo front end que sirve para analizar el código fuente de un lenguaje de programación concreto sirva para generar código máquina en varias plataformas distintas. Suele incluir la generación y optimización del código dependiente de la máquina.

El código que genera el back end normalmente no se puede ejecutar directamente, sino que necesita ser enlazado por un programa enlazador (*linker*)

Muestre cómo variaría la arquitectura usando este nuevo esquema.

Ejercicio 11

Una empresa multinacional de desarrollo de software desea desarrollar un servicio de calendario que pueda ser accedido por usuarios desde diferentes plataformas y dispositivos. El calendario estará disponible en una página web a la cual los usuarios podrán ingresar identificándose con una clave de usuario y una contraseña. Esta no será la única forma de interactuar con los calendarios, dado que también se podrán instalar distintas aplicaciones en diferentes dispositivos (celulares, tablets, PCs, etc.) capaces de generar diferentes visualizaciones de los calendarios

Debido a que se almacenan calendarios de diferentes usuarios que pueden contener información sensible, éstos se guardan de forma encriptada. Es por esta razón que cuando un usuario desea acceder a un calendario debe primero identificarse ante el sistema y luego, si el usuario tiene los permisos necesarios para operar sobre el mismo, se desencripta el calendario y se realizan las operaciones necesarias para preparar la vista del usuario.Además, un usuario podrá configurar alertas en su calendario para que le provea notificaciones de determinados eventos que se reflejarán en las diferentes aplicaciones.

a) Modele la arquitectura del sistema, proponiendo un estilo o mezcla de estilos adecuado. Construya al menos un diagrama con el punto de vista de componentes y conectores, y un diagrama con el punto de vista de deployment.

b) Explicite qué restricciones posee la solución propuesta.

Ejercicio 12

Se desea desarrollar un sistema de correo electrónico (es decir la provisión a usuarios particulares registrados como tales por el dueño del sistema de casillas de correo electrónico privadas, con las que puedan comunicarse con cualquier otra casilla de correo electrónico en internet, como por ejemplo*Gmail*, *Yahoo mail*, o *Outlook*). El sistema debe brindar las siguientes características:

- Operación de la casilla a través de una página web.
- Detector de spam, enviando los mensajes así clasificados a una carpeta especial que almacena ese tipo de mensajes.
- Uso de una "blacklist", es decir, si el remitente de un mensaje a casillas del sistema está en esta lista, el mensaje es siempre rechazado silenciosamente.
- Configuración de filtros por parte de los usuarios.
- Operación desde aplicaciones especiales para leer correo desde una PC (como por ejemplo Thunderbird o Outlook) o desde aplicaciones para tablets o smartphones.
- Se debe resguardar la privacidad de los clientes.

a) Modele la arquitectura con un estilo o mezcla de estilos que sean adecuados para la funcionalidad y tipo de sistema proyectado. Explique por qué eligió ese estilo o mezcla de estilos. Elija qué vistas presentará del modelo elegido; debe realizar al menos dos vistas.

b) Indique un estilo arquitectónico distinto del (o los) usado(s) en el inciso anterior, que no sea adecuado como modelo de la arquitectura del sistema proyectado. Explique las razones por las que este otro estilo resulta verdaderamente inapropiadopara la situación.

Ejercicio 13

Se quiere diseñar un sistema de reproducción de multimedia online. En este sistema el cliente corre una aplicación mediante la cual solicita y recibe datos (video y/o música) de un servidor de aplicación (lo llamaremos AppS de ahora en más) con el cual se conecta a través de Internet. El AppS obtiene los archivos correspondientes al pedido del usuario de dos servidores dedicados: un servidor de video (VS) y un servidor de audio (AS), los cuales pueden estar localizados en diferentes hosts. Según lo indicado por el cliente en el pedido el AppS puede a los videos codificarlos a diferentes formatos, modificar su calidad y/o agregar subtítulos, mientras que a los audios puede comprimirlos y modificar su calidad. Una vez recibidos, el cliente ve/escucha los datos multimediales en la aplicación.

La aplicación de *setup* puede descargarse a través de un sitio web dedicado que se conecta al AppS para descarga el archivo instalador correspondiente y se instalada en una computadora o dispositivo personal. Luego de la instalación y pasada la etapa de registración, la aplicación puede comunicarse con el "Media Store" residente en el AppS donde puede comprar y descargar música y video y almacenarlo en una librería.

Los dispositivos móviles como tabletas o smartphones pueden actualizar su propia librería desde una computadora personal a través de una conexión USB, o pueden descargar media directamente del "Media Store" usando algún protocolo de red inalámbrico como Wi-Fi, 4G, etc.

Proponga una arquitectura para el sistema de la siguiente manera:

- Dibuje y explique brevemente la arquitectura del sistema mediante una vista de *componentes y conectores*, a nivel de una perspectiva global del sistema. Describa brevemente la operación de cada subsistema. Detalle en qué estilo arquitectónico o combinación de estilos está basada esta vista (incluya una descripción de la notación que usa).
- Dibuje y explique brevemente la arquitectura del sistema mediante la vista de *deployment*.

Ejercicio 14

*Una empresa de video juegos está comenzando el desarrollo de una API grafica (**gAPI**) que será la base de una serie de nuevos juegos de rol de fantasía en tercera persona.*

*El software **gAPI** busca tener una interfaz con los principales lenguajes de programación: C++, C# y Java. Esta interfaz abstrae al programador de los detalles del motor gráfico, traduciendo los pedidos de un programa a directivas del motor. Posteriormente, para efectivizar la redenerización de los gráficos requeridos, el motor gráfico puede comunicarse tanto con DirectX como con OpenGL, los cuales interactúan directamente con el sistema operativo para mostrar los gráficos por pantalla.*

*La actividad principal del motor gráfico de **gAPI** es dibujar escenas. Una escena se compone de un terreno, estructuras, personajes y postefectos. Primero se dibuja el terreno, luego se agregan las estructuras, después los personajes y posteriormente se aplican los postefectos sobre todos los elementos de la escena (sombras, iluminación, etc.).*

*Una particularidad del motor gráfico de **gAPI** es la renderización completa de los objetos que un personaje tiene equipados. Para esto, el motor debe ir montando los objetos portados por el personaje en el orden adecuado de manera tal que no haya "clipping" (que los objetos que deberían estar más abajo tapen en alguna parte a los que deberían estar más arriba, por ejemplo que la ropa tape a la mochila). En la primera versión del motor los personajes podrán portar: Ropas, Armadura, Guantes, Calzado, Gorro/Casco, Túnica, Capa, Equipo en el Cinto, Equipo en la Espalda, Capa. El motor debe ser eficiente dibujando a la vez todos los objetos que no se superpongan.*

(a) Dibuje y explique la arquitectura del sistema mediante una vista de *componentes y conectores* considerando todas las características mencionadas en la descripción de arriba. Detalle en qué estilo arquitectónico o combinación de estilos está basada la vista presentada. Para mayor claridad, puede presentar la vista en varios niveles; es decir, un

componente complejo en un nivel puede ser expandido a un diagrama propio de componentes y conectores de otro nivel.

(b) Considere dos estilos arquitectónicos de los vistos en la materia y que *no haya utilizado* en el inciso anterior. Explique claramente por qué no los utilizó. Si los hubiese utilizado, ¿qué problemas hubiese tenido la arquitectura? Justifique su respuesta.

(c) En base a la arquitectura presentada en el inciso (a), considere las siguientes extensiones. ¿Cuál le parece más simple de incorporar? ¿Por qué?

- Los personajes pueden tener accesorios como anillos, aros, colgantes, y/o bandoleras.
- El motor gráfico se comunica directamente con el sistema operativo para mostrar la escena.

(d) Suponga que se está diseñando **Fulanito Quest**, un juego de rol multijugador, utilizando el software **gAPI**del ejercicio anterior. El juego lo pueden jugar varios jugadores en una red local o a través de servidores dedicados, y éste podrá adquirirse a través del sitio web de la empresa, de la plataforma online de ventas *JuegosAlVapor* o mediante copias físicas en DVD. Dibuje y explique la arquitectura del sistema mediante la vista de *deployment*.