Universidad Rey Juan Carlos – Diseño y Arquitectura Software (DAS)

Práctica1: Captura y Representación de Decisiones de Diseño

Curso 2022-2023

1. **Objetivo**: Dada una especificación de un problema se precisa identificar, capturar y documentar las decisiones de diseño y construir la arquitectura software de un sistema.
2. **Roles**: Se formarán grupos de **6 personas** con los siguientes roles:
    a. **Arquitectos Software Senior (ASS)**: Dos integrantes que tengan más experiencia que el resto del grupo actuaran como las personas que toman las decisiones de diseño y las capturan.
    b. **Arquitectos Software Junior (ASJ)**: Dos integrantes con experiencia en UML u otra notación similar se encargan de modelar y documentar la arquitectura software en base a las decisiones de diseño tomadas. En aquellos grupos de 5 miembros habrá un solo ASJ.
    c. **Arquitectos Software Cognitivos (ASC)**: Dos integrantes se encargan de cuestionar las decisiones tomadas y plantear nuevos problemas de diseño.
3. **Herramientas sugeridas***:* Cualquier herramienta de modelado UML (se recomienda MagicDraw).
   Otros aspectos: Se sugiere a todos los grupos instalar la herramienta ADMentor para estudiar cómo se capturan las decisiones y se modelan las soluciones de diseño. ADMentor se encuentra en: https://www.ifs.hsr.ch/index.php?id=13201&L=4, https://github.com/IFS-HSR/ADMentor pero necesita una instalarse sobre una licencia de Sparx Enterprise Architect. Puede utilizar una versión de prueba en: http://www.sparxsystems.com/products/ea/trial/request.html

   Captura de decisiones de diseño: Las decisiones se capturarán utilizan el formato MADR para capturar las decisiones de diseño. **El formato MADR se encuentra en** https://github.com/adr/madr. No requiere instalación. Un ejemplo de decisiones capturadas en MADR se encuentra en https://microservice-api-patterns.org/. Las decisiones capturas y las arquitecturas así como ficheros de voz se almacenarán cada semana en GitHub. No se podrán subir decisiones o arquitecturas de semanas anteriores.
4. **Reparto de grupos:** La mitad de los grupos serán denominados **TEST** y la otra mitad **CONTROL**. La asignación de las dos categorías la realizará el profesor.
    a. Los grupos de **TEST** deberán grabar ficheros de voz de entre 2 y 10 minutos con algunas de las discusiones que tengan lugar entre los ASS-ASC y entre ASS-ASJ. Si algún grupo necesita en algún momento grabar una discusión mayor de 10 mins pueden hacerlo.
    b. Los grupos de **CONTROL** no grabarán las decisiones con voz, solo las capturan con MADR.
5. **Tareas**: Las tareas a realizar se desarrollarán durante **5 semanas**. **Fecha de comienzo: 19/10 Fecha Finalización de las iteraciones: 16/11. El TEST online será 17/11.**
    a. **Tarea 1: Semana 1: Análisis de requisitos**: Todos los miembros del grupo deberán comprender el problema y realizar una fase de análisis y captura de requisitos

funcionales y de dominio. Cada requisito deberá llevar un identificador (RF1, RF1.1., RF2, etc o bien RNF para los no funcionales), un nombre suficientemente descriptivo y una descripción detallada.

b. **Tarea 2: Semana 2: Iteración 1 de la arquitectura:**
    i. Los arquitectos senior (ASS) comienzan a plantear las primeras decisiones de diseño mientras que el resto del grupo ayuda en la búsqueda de patrones de diseño y estilos arquitectónicos. Estas primeras decisiones se capturan según la plantilla escogida y se deberá medir el tiempo (**Time in ADD**) en tomar dichas decisiones desde que se comienza a pensar en el problema de diseño hasta que se captura una decisión. Los ASS de los grupos de CONTROL deberán familiarizarse con el formato MADR. Se deberá crear una cuenta en GitHub para almacenar las decisiones y las arquitecturas de las iteraciones. Si una iteración s compleja puede contener sub-iteraciones y en cada sub-iteración se producirá un refino de la arquitectura. Los ASS de los grupos de TEST deberán familiarizarse con la grabación de ficheros en TEAMS u otro medio similar.
    ii. Los arquitectos cognitivos (ASC) se reúnen con los seniors para cuestionar dichas decisiones y se genera un documento de riesgos o problemas en las decisiones. Asimismo, pueden plantear nuevos problemas de diseño. Se mide el tiempo (**Reflection time**) empleado en el proceso de reflectividad.
    iii. Los ASS reflexionan sobre las cuestiones planteadas y elaboran de nuevo las decisiones modificadas o nuevas y se mide de nuevo el tiempo (**Time in refine ADD**).
    iv. Los arquitectos junior (ASJ) toman las decisiones tomadas y documentan la arquitectura en UML con las vistas que ellos consideren necesarias. Se mide el tiempo (**Design ADD**) en modelar la primera versión de la arquitectura. Cada iteración de la arquitectura se almacenará en un fichero diferente. Los ASJ junior deberán realizar la descarga e instalación del software necesario para realizar la práctica y practicar con la herramienta ADMentor.
    v. Rellenar la plantilla de tiempos por cada semana de trabajo.
    vi. Subir las decisiones y las arquitecturas producidas al GitHub para cada iteración.

c. **Tarea 2. Semanas 3, 4, 5:** Iteraciones de la arquitectura: Se repiten los pasos de la Tarea 2 y se deberá entrega la plantilla de tiempos estimados.

d. **Tarea 3. Semana 5 (17/11):** No hay más iteraciones. El grupo se reúne para discutir los resultados del diseño y extraer las conclusiones. Se terminará la documentación explicando conclusiones y aspectos finales. **Cada alumno deberá completar unas preguntas finales de manera individual.**

6. **Entrega de la práctica**: La entrega de la práctica se realizará via email a rafael.capilla@urjc.es o mediante el campus virtual. El nombre de fichero deberá ser "**DAS-**

P1-NombreApellidos del responsable del grupo". **Cualquier fichero que no siga esta nomenclatura será rechazado y se supone práctica no entregada**. El fichero debe **enviarse comprimido** y debe contener la memoria en **formato WORD o PDF. LA fecha límite de entrega de la memoria es el 22/11/2022. El fichero comprimido contendrá las siguientes partes:**

a. **Figuras de las arquitecturas producidas en cada iteración**
b. **Un fichero en texto plano con el enlace al repositorio GitHub con el usuario y clave de acceso**
c. **Ficheros de voz para los grupos de TEST**
d. **Una memoria en Word o PDF no superior a 25 páginas con la siguiente información:**
    i. **Portada con nombre de la práctica, asignatura, lista de miembros del grupo, email del responsable. Índice generado de forma automática y numerado. Las figuras y tablas deben numerarse y llevar un pie de figura y tabla apropiado.**
    ii. **Nombres de los integrantes para cada rol**
    iii. **Breve informe con capturas de pantalla del uso de ADMentor (opcional) y lista de requisitos**
    iv. **Descripción de los resultados para cada tarea incluyendo los resultados intermedios. Las arquitecturas resultantes deben incluir al menos un diagrama de clases y paquetes y el de despliegue.**
    v. **Decisiones tomadas en cada iteración y las arquitecturas resultantes**
    vi. **Conclusiones en base a lecciones aprendidas**
    vii. **Bibliografía adecuadamente formateada**
    viii. **Anexo con todos los tiempos estimados**

Evaluación. La evaluación sobre 10 consistirá en: 80% el contenido, 20% calidad y presentación de la memoria. El contenido se dividirá un 30% las arquitecturas producidas y un 40% las decisiones de diseño con los tiempos estimados y un 10% las discusiones en los ficheros de voz. Los grupos de CONTROL al no grabar ficheros de voz reparten los pesos en 30% y 50%. **Una práctica en la que los alumnos no hayan asumido los diferentes roles se considera suspensa**. El profesor podrá realizar preguntas a los alumnos de un grupo en cualquier momento lo cual puede influir en la nota individual de un alumno respecto de la nota de los restantes miembros del grupo. **Si algún grupo no sube cada semana con su fecha correspondiente las decisiones y las arquitecturas al GitHub la práctica se considera suspensa.**

Anexo I: Descripción del sistema que se pretende diseñar

Un sistema basado en una arquitectura Web de tres capas (tiers) se desea migrar a una arquitectura de microservicios más flexible y escalable. El sistema actual gestiona las solicitudes mediante una lógica de negocio dos bases de datos (una SQL y otra NoSQL) que almacenan datos de compras por Internet y de preferencias de los clientes respectivamente. Actualmente, el uso de móviles y tabletas para las compras demanda una mayor escalabilidad del sistema actual. La aplicación existente consta de los siguientes módulos:

- Componente de presentación: Son los responsables del control de la interfaz de usuario y el consumo de servicios remotos.
- Componentes de lógica de negocio: Módulo de pedidos y compras de clientes, módulos de detección de preferencias, módulo de conexión a sistemas de pago, módulo de mensajería a dispositivos móviles, módulo de seguridad en compras y módulo de devoluciones.
- Lógica de acceso a bases de datos. Son los componentes de acceso a datos responsables de obtener acceso a las bases de datos (SQL o NoSQL).
- El sistema de mensajería se soporta por un middleware independiente.

La nueva aplicación deberá ser capaz de integrar microservicios de forma asíncrona y altamente escalable para soportar un mayor número de compras vía móvil y detección de preferencias de usuarios de una forma más eficiente. Los microservicios que soporten la reingeniería y migración de está funcionalidad deberán usar el protocolo REST mientras que la mensajería usará AMQP. Los arquitectos software decidirán cuantos microservicios y en que lenguaje de programación se van a desarrollar así como el middleware necesario para albergar dichos microservicios.

Entre los problemas adicionales de diseño que se plantean es decidir cuantos microservicios son necesarios dado que se deberá contar con una nueva base de datos en MongoDB para almacenar la localización de microservicios de la empresa y otros similares. Además, es necesario que exista una coherencia entre las bases de datos de los diferentes microservicios que se gestione mediante a través de un bus de eventos lógicos (e.g. Command and Query Responsibility Segregation) que utilicen tecnología de mensajería como RabbitMQ o un “bus” dedicado como pueden ser Azure Service Bus, NServiceBus, MassTransit o Brighter. La selección de la tecnología se basará en aspectos de escalabilidad y prestaciones así como de interoperabilidad.

Asimismo, se deberá limitar el número de intentos de compra a 5 de manera que no se sobrecargue el número de peticiones a los microservicios correspondientes. Los arquitectos software deberán determinar cuantos contenedores de microservicios son necesarios.

Todas las comunicaciones son vía HTTP. Los clientes se comunican con los microservicios mediante un componente Gateway que contiene un módulo para monitorizar el estado de cada microservicio

El alumno deberá suponer que datos requiere cada microservicio o el cliente para gestionar una tienda virtual (e.g. numero de pedido, identificación de cliente, producto, cantidad, precio, total, forma de pago, etc.)

Anexo II. Plantilla de tiempos (en minutos)

Week	Iteration	Time in ADD (ASS)	Reflection Time (ASS-ASC)	Time in refined ADD (ASS)	Design ADD Time (ASJ)
1	1	28	33	20	28
1	2	25			
2	3				
3	4				
3	5				

Universidad Rey Juan Carlos – Diseño y Arquitectura Software (DAS)

Práctica2: Evaluación de Atributos de Calidad

Curso 2022-2023

7. **Objetivo**: Poner en práctica el método ATAM para evaluar atributos de calidad en la arquitectura final producida en la práctica 1.
8. **Roles**: Se formarán grupos de **6 personas** con los siguientes roles:
    a. **Cliente (CLI)**: 1 integrante del grupo actúa como cliente que exige calidad en la arquitectura del sistema de la práctica 1.
    b. Equipo ATAM: **1 Lider de la evaluación, moderador y controlador del tiempo (LID), 2 Redactores de escenarios (SCE), 2 Cuestionadores (CUE)**
    c. Intercambio de roles:

Práctica 1	**Práctica 2**
Arquitectos Junior	**Cuestionadores (CUE)**
Arquitectos Cognitivos	**Redactar escenarios (SCE)**
Arquitectos Senior	**1 moderador (LID), 1 Cliente**

9. Tareas: Las tareas a realizar se desarrollarán durante **4 semanas**. **Fecha de comienzo: 23/11/2020. Fecha Finalización: 22/12/2022. Entrega el 30/12/2022 hasta las 14.00. La semana del 6 de Diciembre no computa a efectos de trabajo.**

a. **ATAM Fase 1 (pasos 1-3) (23-24/11). Preparación equipo ATAM y presentar la arquitectura y objetivos de calidad**:

Paso 1: El líder organiza el equipo ATAM y presenta el método al resto del equipo y al cliente.

Paso 2: El cliente presenta los objetivos de negocio y calidad a conseguir para el diseño del sistema de la práctica 1. El Cliente discute con el resto del equipo ATAM qué aspectos de calidad son más importantes y cuales se deberían introducir.

Paso 3: Los dos arquitectos Junior de la práctica 1 presentan y explican la arquitectura de dicha práctica al resto del equipo y que atributos de calidad son los deseados para conseguir los objetivos de calidad. Explicar que aproximaciones arquitectónicas podrían ser válidas paras soportar los atributos de calidad. Se deberá incluir una imagen completa de la arquitectura de partida.

b. **ATAM Fase 2 (pasos 4-6) (30/11, 01/12, 14/12 y 15/12):**

Paso 4. Identificar las partes del a arquitectura y del sistema que son susceptibles de soportar los elementos de calidad que se desean y que sean medibles con atributos de calidad.

Paso 5. Construir el utility tree con los escenarios para cada atributo de calidad (QA).

Paso 6. Cuando existan, analizar las diferentes arquitecturas candidatas para diferentes escenarios. Asignar prioridades a los escenarios. Identificar puntos de riesgo y sensivity points. Realizar el trade-offs de los QA.

c. **ATAM Fase 2 (pasos 7-9) (21/12 y 22/12). Finalización de ATAM**

Paso 7. Discutir y presentar los escenarios seleccionados

Paso 8. Volver al paso 6 o introducir más escenarios si fuera necesario volviendo al paso 5.

Paso 9. Presentar los resultados al cliente y modificar la arquitectura software de acuerdo a los atributos de calidad introducidos. Se deberá incluir una imagen de la arquitectura final modificada.

Entrega de la práctica: La entrega de la práctica se realizará mediante el campus virtual. El nombre de fichero deberá ser "DAS-P2-NombreApellidos del responsable del grupo". **Cualquier fichero que no siga esta nomenclatura será rechazado y se supone práctica no entregada**. El fichero debe **enviarse comprimido** y debe contener la memoria en **formato WORD o PDF**. La memoria no debe superar las 20 páginas y contener los siguientes apartados:

Portada con nombre de la práctica, asignatura, lista de miembros del grupo, email del responsable. Índice generado de forma automática y numerado. Las figuras y tablas deben numerarse y llevar un pie de figura y tabla apropiado.

a. **Nombres de los integrantes para cada rol**
b. **Descripción de los resultados para cada fase y paso de ATAM incluyendo los resultados intermedios, los atributos de calidad elegidos motivando su elección.**
c. **Utility tree con los escenarios y discusión sobre cómo se han seleccionado.**
d. **Arquitectura resultante después de añadir los aspectos de calidad seleccionados.**
e. **Conclusiones en base a lecciones aprendidas**
f. **Bibliografía adecuadamente referenciada formateada**

Evaluación. La evaluación sobre 10 consistirá en: 80% el contenido, 20% calidad y presentación de la memoria. **Una práctica en la que los alumnos no hayan asumido los diferentes roles se considera suspensa**. **Atributos de calidad con un solo escenario se consideran inválidos. Parte de la nota ira en función de la calidad y amplitud del utility tree.**

Recomendaciones

- No escoger demasiados atributos de calidad (no más de 4)
- Las necesidades de calidad del cliente se deben asociar a atributos de calidad

- Evitar atributos difíciles de evaluar como maintainability, usability, etc o bien limitaciones de Coste, si estas no vienen indicadas
- Atributos como Security, Performance, Interoperability, etc son más asequibles de evaluar
- Seleccionar los QAs con su nombre en Inglés y revisar que la definición encaja con el objetivo de calidad del cliente
- Los escenarios deben ser suficientemente concretos y que se puedan evaluar. Deben afectar a partes concretas de la arquitectura
- Debe haber al menos 2-3 escenarios por cada atributo de calidad y parte del sistema
- EL utility tree se puede hacer en forma de tabla en lugar de forma de árbol

Problema de diseño

Problema: Un sistema recoge datos de sensores de humedad y temperatura y los debe enviar para su procesamiento a un centro de control que visualiza la información y emite alarmas en caso de superación de valores umbrales. Los datos de ambos tipos de sensores se transmiten en ráfagas cada 30 minutos. En caso de fallo de un sensor se invoca un servicio Web remoto por tipo de sensor que recoge datos de otra estación situada en un radio de 50 km. Asimismo, el sistema de control permite a usuarios desconocidos suscribirse a un canal de información para obtener datos procesados de los sensores. Finalmente, existirá una capa de seguridad donde la información de los sensores se encripta entre el centro de control y el módulo de procesamiento de datos.

Se pide: Elegir el estilo arquitectónico o estilos más adecuados para un diseñar una arquitectura básica a nivel de bloques funcionales. Capturar las decisiones de diseño mas importantes. Para cada decisión, proporcionar un nombre corto, descripcción breve, pros y cons, y decisiones alternativas.

Modelo Mind1-Mind2

Problema: Respecto al problema de diseño anterior, formar grupos de 2/3 personas y criticar las decisiones tomadas. Su pueden sugerir alternativas, riesgos y nuevas decisiones. Justificar las nuevas alternativas propuestas.

Problema 1: Un software de procesamiento gráfico necesita renderizar imágenes con alta calidad. Para ello es necesario calcular el número de polígonos y vértices al que posteriormente se le aplica una reducción de caras ocultas. A continuación se imprimen texturas en las caras pudiendo elegir entre tres tipos diferentes. Por último, se imprime una capa de color sobre las texturas pudiendo elegir tres combinaciones diferentes de colores.
¿Qué estilo arquitectónico resultaría más adecuado y cuantas clases se ven involucradas?

Problema 2: Un software gráfico necesita renderizar se compone de 4 filtros diferentes para reducir comprimir imágenes. Cada filtro dispone a su vez de dos métodos distintos para calcular el tamaño del gráfico según tenga elementos ocultos o no. Finalmente, cada método de cálculo del tamaño permite seleccionar imágenes en base a un algoritmo diferente basado en una imagen de referencia. *¿Qué patrón o patrones de diseño podría resolver el problema?*

Problema 3: *¿Qué patrón o patrones de diseño permite seleccionar entre diferentes algoritmos de routing basados en el estado de sobrecarga de la red y en unos pesos que discriminan el tráfico de red en una organización determinada?*

Problema 4: *¿Si tuviéramos que mantener un software con un número excesivo de llamadas entre sus diferentes métodos y por otra parte es necesario vigilar el comportamiento de la clase A que se comporta de manera agresiva realizando comprobaciones de su estado cada poco tiempo, que patrones de diseño podríamos utilizar?*

Es lícita la inclusión en una obra propia de fragmentos de otras ajenas de naturaleza escrita, sonora o audiovisual, así como la de obras aisladas de carácter plástico o fotográfico figurativo, siempre que se trate de obras ya divulgadas y su inclusión se realice a título de cita o para su análisis, comentario o juicio crítico. Tal utilización solo podrá realizarse con fines docentes o de investigación, en la medida justificada por el fin de esa incorporación e indicando la fuente y el nombre del autor de la obra utilizada.