2016

UNC – FCEFyN
Ingeniería en Computación
“Ingeniería de Software”

Mgr. Martín Miceli

[PRÁCTICO DE ING. DE SW]

Este documento describe los requisitos mínimos que debe tener el trabajo práctico final para la materia “Ingeniería de SW”. Dicho trabajo consiste en la aplicación de las prácticas más importantes de ingeniería de software en base un trabajo de programación específico.

Tabla de Contenidos

Descripción del trabajo práctico

Acerca del informe final

El alumno, en grupos de aproximadamente 3 personas (máx. 4), deberá usar el ejemplo de MVC para extenderlo según se especifica en la sección "Acerca del Tema del trabajo". También, se podrá usar el trabajo actual de concurrentes, u otro trabajo particular previa aprobación del profesor de Ing. de SW. El grupo deberá aplicar las metodologías y conceptos vistos en clase. Se deberá presentar un reporte hecho en Word 2010/2013 (o exportado a *.pdf) que incluya los requisitos mínimos listados en las sección ("Lista de requisitos del informe"). Se recomienda incluir cada grupo de requisitos como secciones separadas de un mismo documento que tenga su historial propio de versiones de modificación (i.e. 1 sólo informe con secciones como "Requerimientos", "Manejo de las Configuraciones" etc.). Agregar fotos de los participantes indicando el nombre de cada uno. Se deben incluir capturas de pantalla de las partes del software que considere necesarias.

Acerca de la presentación

Además, los gráficos y tablas claves deberán ser incluidos en una presentación que se usará para defender el trabajo el día acordado (estas deberán ser enviadas en formato *.ppt. En caso de usar otro formato de presentación, como ejemplo prezit, se deberá enviar la presentación exportada a*.pdf). El trabajo debe ser presentado a más tardar el día anterior a la presentación de dicho trabajo. La fecha límite de entrega del trabajo y de la presentación será el Domingo 28-Jun-15, mientras que la presentación se llevará a cabo el Lunes 29-Jun-2015 a las 18:00hs. Cada equipo deberá defender el trabajo durante 10 minutos (7' presentación y 3´ de preguntas y respuestas). El día de la presentación, el profesor asignará dos temas dentro de la lista de requisitos del informe en los cuales cada grupo deberá focalizarse durante la presentación. Agregar fotos de los participantes indicando el nombre de cada uno. Se deben incluir capturas de pantalla de las partes del software que considere necesarias.

Acerca del formato de la entrega

Los archivos deberán ser comprimidos en formato *.zip (no rar u algún otro formato) y enviados por email a martinmiceli@gmail.com, noninojulian@gmail.com antes de la fecha final de cierre de recepción.
Los archivos a entregar son:

- Informe.
- Presentación.
- Ejecutables (JARs) que no necesiten dependencias externas. (Enviarlo antes así lo podemos evaluar antes de la fecha de presentación).
- Proveer acceso al repositorio.

Acerca del tema de Trabajo

En base el ejemplo del Libro Head First design Patterns en la página 526 a 548 (DJView), extenderlo de la siguiente forma:

1. Modifique el HeartModel para que sólo se pueda crear una instancia (usando el patrón Singleton) y extienda la ventana de control del BeatController para "tratar" de generar nuevas instancias cada vez que se clickea en el botton ">>". La ventana de la BeatBar debería mostrar en texto el número de intentos de creación de un nuevo HeartBeat model en el texto donde se mostraba la frecuencia cardíaca.
2. Crear un nuevo modelo con su controlador específico que pueda usarse para verse desde la vista DJView en la ventana BeatBar. Generar un java main class para poder ejecutar tal modificación llamándolo "My<modelName>TestDrive.java" (similar to "HeartTestDrive"). Se proveerán puntos adicionales por la originalidad del modelo creado.
3. Generar una vista propia que permita usar su modelo sin modificar el código existente del ejemplo pero que permita mostrar los cambios gráficamente y por medio de texto.
4. Generar un TestDrive que permita mostrar a los tres modelos trabajando al mismo tiempo (se esperarán ver al menos 3 ventanas BeatBar con los 3 modelos andando simultáneamente.
5. Modificar la vista BeatBar para que permita cambiar gráficamente en tiempo de ejecución (ej. Mediante un dropdown box) el modelo usado (el Beat model, el Nuevo modelo y el Heartbeat model). Por favor use para tal implementación el patrón strategy. Generar un TestDrive que permita ejecutar tal acción.
6. Utilizar algún sistema como el sistema de Issues de GitHub para gestionar las tareas y defectos encontrados.
7. Incluir todos los documentos generados en el repositorio dentro de una carpeta llamada docs. Los documentos generados deben estar en formato Markdown. Pueden utilizar algún programa para editar este tipo de archivos como MarkDown Pad. De ésta manera se podrá observar el historial de cambios sobre los documentos y quien realizó cada cambio.

Acerca de la propiedad intelectual

Es importante notar que el trabajo práctico debe ser realizado enteramente por el grupo, esperándose contribuciones similares de cada integrante (debe mencionarse el autor de cada sección y los revisores y contribuidores). Cualquier intento de copia o modificación de trabajos presentados por otros en el presente o con anterioridad, derivará a la desaprobación directa y consecuentes acciones por fraude del grupo y sus participantes.

Lista de requisitos del informe

El informe a presentar deberá tener al menos una sección por cada apartado identificado abajo (ej. 1-Nota de Entrega*, 2- Manejo de las Configuraciones, 3-Requerimientos, 4-Arquitectura, 5-Diseño e Implementación, 6-Pruebas Unitarias y del Sistema, 7-Datos Históricos, 8-Información Adicional).

Nota de Entrega

Esta sección es lo último a escribir en el informe ya que requiere de haber completado las secciones siguientes para tener su contenido. Por favor, continúe leyendo las siguientes secciones y retorne aquí luego de haber finalizado.

1. Proveer la "**Nota de Entrega** (i.e. el "Release Note") con el estado del entregable al momento de la presentación del trabajo práctico. Incluir los siguientes puntos:
    a. Breve **listado de la funcionalidad** incluida (con el **estado de implementación** de c/u)

b. **Pass/Fail Ratio** de sistema
c. **Bugs conocidos** (i.e. no resueltos) en la entrega
d. **Lugar/link** del **entregable y de las instrucciones de instalación**

Manejo de las Configuraciones

2. Incluir una sección para el **plan de manejo de las configuraciones** que incluya la forma de acceder al código fuente y a los documentos incluidos en la herramienta de control de versiones. Esta sección es el índice a toda la información del proyecto incluyendo:
   a. Dirección y forma de accesos a la herramienta de control de versiones
   b. Esquema de directorios y propósito de cada uno.
   c. Normas de etiquetado y de nombramiento de los archivos.
   d. Plan del esquema de ramas a usar (y en uso).
   e. Políticas de fusión de archivos (o sea mergeo) y de etiquetado de acuerdo al progreso de calidad en los entregables.
   f. Forma de entrega de los "releases", instrucciones mínimas de instalación y formato de entrega.
   g. Listado y forma de contacto de los integrantes del equipo, así como sus roles en la CCB. También incluir periodicidad de las reuniones y miembros obligatorios.
   h. Herramienta de seguimiento de bugs usado para reportar los defectos descubiertos y su estado.
   i. Cualquier otra información relevante (ej. Direcciones del servidor de integración continua, etc.)
   Nota: usar el plan de CM de ejemplo dado en clase como guía de contenidos "PROJECT_CM_Plan_Example.pdf"

Requerimientos

3. Incluir una sección para los **Requerimientos** que incluya los "**Diagrama de Casos de Usos**", "**Diagrama de Actividades**" y "**Diagramas de secuencias**" para introducir al lector en los requerimientos de usuario para el sistema a construir.
4. También deberá incluir el detalle de los **requerimientos funcionales** y **no funcionales de sistema**.
5. Se deberán incluir un **diagrama de arquitectura preliminar** que permita asociar requerimientos y casos de usos con los sistemas, subsistemas y módulos identificados.
6. También, una **Matriz de trazabilidad** entre los **casos de usos y los requerimientos**.

Arquitectura

7. Un **gráfico de arquitectura** general para mostrar los **componentes y sus relaciones** con las **interfaces externas**. Explicitar que **patrón de arquitectura** fue usado y porqué, haciendo énfasis **como resuelve los requerimientos no funcionales**. Adicionalmente se deben incluir al menos los gráficos **UML de despliegue** y de **componentes**. Opcionalmente se puede incluir in diagrama de **contexto** (**composite: actividades y sub-sistemas**) que incluya la relación de los sistemas/subsistemas con las actividades del dominio del conocimiento de la aplicación.

Diseño e implementación

8. Incluir los **Diagramas de clases** y de **objetos**. Opcionalmente generar los **diagrama de secuencia** y/o **estados** según sea aplicable con alguna parte del diseño.
9. Agregar el **Diagramas de paquetes** de agrupamiento de clases.
10. Implementar al menos un **Patrón de diseño** adicional a los ya incluidos en el ejemplo. Para aquellos que se basen en el práctico de concurrente u otro trabajo deberán incluir al menos un

Observer y/o un Strategy. Resaltar los patrones de diseño usado en el diagrama de clases e indicar porqué se optó por este (explicar qué problema soluciona).

Pruebas unitarias y del sistema

11. Generar **pruebas unitarias** automáticas (o sea Unit test basándose en JUnit o similar) para el código, explicar cómo correrlos y verificar su estado. Opcionalmente: proveer los porcentajes de cobertura de código desde la herramienta.
12. Generar los **Casos de Prueba de Sistema** contra los **requerimientos funcionales** y **no funcionales**. Incluir **casos de prueba alternativos** que prueban valores límites o inusuales y que tratan de generar errores no esperados.
13. Seleccionar un subconjunto de los **casos de prueba** para actúen como **Smoke o Sanity Tests** y permitan la aceptación por parte del cliente.
14. Actualizar la **Matriz de trazabilidad** para incluir el mapeo entre casos de usos, requerimientos, **módulos o clases y casos de pruebas** (unitarias y de sistema).
15. Calcular el **"Pass/Fail Ratio** (i.e. % Pruebas Pasadas/Falladas)**"** para **cada tipo de pruebas (**i.e. para los Test Unitarios, de Integración y de Sistema).
16. Proveer el **número de bugs identificados y corregidos** (agrupados por severidad). Incluir el **link a la herramienta de administración de bugs (**ej. Bugzilla, Jira, etc.)
17. Opcional: incluir cualquier otra herramienta de gestión de la calidad de SW que haya sido usada (ej. FindBugs for análisis estático de código, etc.)

Datos Históricos

18. **Detalle de dedicación de esfuerzo** para realizar el trabajo (en Personas Horas), distinguiendo la contribución personal de cada miembro del grupo y el esfuerzo invertido por cada tarea realizada. Esto incluye todas las tareas para poder construir la aplicación, documentarla y elaborar el informe y la presentación de esta materia.

Información Adicional

19. **Lecciones aprendidas** durante la elaboración del práctico y errores cometidos.