Universidad de Burgos

Escuela Politécnica Superior

Grado en Ingeniería Informática

Práctica 5

Simulación de un proceso de refactorización.

Alumnos Álvaro Ruiz Molledo
Víctor Pérez Esteban

Tutor Carlos López Nozal
Departamento de Ingeniería Civil
Área de Lenguajes y Sistemas Informáticos

Burgos, 5 de junio de 2015

Índice de contenido

Índice de ilustraciones

Índice de tablas

1 OBJETIVOS ESPECÍFICOS

- Aplicar un proceso de refactorización sobre un pequeño proyecto existente.
- Conocer como corregir defectos de código usando refactorizaciones.
- Relacionar una lista de tareas de desarrollo y de mantenimiento software con el proceso de refactorización.

2 REQUISITOS

2.1 Requisitos teóricos

- Conocimiento de un repositorio de refactorizaciones http://www.refactoring.com
- Evaluar la calidad de los códigos de prueba a través de métricas de cobertura.
- Conocimiento de catálogos de defectos de código y técnicas de detección.
- Relación entre defectos de código y refactorizaciones que ayuden a eliminarlos.
- Relación entre métricas y defectos de código de distintos autores.
- Gestión de configuración de software mediante operaciones de control de versiones.

2.2 Requisitos técnicos

- Ejecución de refactorizaciones desde el entorno de desarrollo Eclipse. http://www.ibm.com/developerworks/opensource/library/os-ecref/
- Obtención y ejecución de históricos de refactorización generados en Eclipse(menú Refactor→History).
- Cálculo de cobertura de pruebas del sistema con el plugin de eclipse EclEmma.
- Búsqueda defectos con el plugin de Eclipse InCode.
- Selección y cálculo de métricas orientadas a objetos con el plugin de Eclipse RefactorIt.
- Acceder y actualizar las versiones de la práctica mediante un repositorio de control de versiones. Para ello, hay que crear un fork del repositorio público en GitHub https://github.com/clopezno/reafactoring_lab_session.git añadiendo a todos los miembros del grupo con permisos de escritura y lectura. El nombre del repositorio será 1415dassxx, donde xx se corresponde con el código de grupo que asigne el profeso

2.3 Requisitos software

- Eclipse IDE for Java Developers http://www.eclipse.org/downloads/
- Plugin Eclipse EclEmma 2.2.0 http://www.eclemma.org/. Manual de instalación. Instalar directamente de la red http://update.eclemma.org/
- Plugin Eclipse RefactorIt 2.7. http://sourceforge.net/projects/refactorit/
- InCode http://www.intooitus.com/products/incode
- Disponer de una cuenta en un repositorio público de de control de versiones GitHub https://github.com/
- Aplicación cliente para comunicarse con un repositorio de control de versiones Git (http://code.google.com/p/tortoisegit/ , Plugin Eclipse...)

3 ENUNCIADO

Esta sesión aborda la refactorización de un sistema que simula una red de área local (LAN).

La documentación del sistema está contenida en el repositorio público https://github.com/clopezno/reafactoring_lab_session.git

La descripción del contenido del repositorio es:

readme_es.md

El fichero actual

./doc/SRe2LICRefacLabo.pdf

Especificación detallada del enunciado. Lista de tareas de la práctica. Cada tarea que implique alguna refactorización deberá ser guarda con un commit/push en el repositorio de control de versiones y asociada a una issue previamente definida en GitHub.

./doc/LANSimulationDocu.pdf

Diagramas UML (casos de uso y clases) documentando la simulación LAN

toDoList

La lista de los casos de uso que han sido implementados y los que están todavía sin hacer.

src

El directorio que contiene la versión de código Java (organizado en subdirectorios para los diferentes paquetes). Los ficheros de comandos compileLAN, generateJavaDoc, runLAN compila, ejecuta o genera la documentación de la simulación. Puedes definir estas tareas en tu entorno de desarrollo.

4 PROCESO DE REFACTORIZACIÓN

4.1 Dirección del repositorio

El repositorio del proyecto se puede encontrar en https://github.com/vpe0001/1415dass06

4.2 Histórico de versiones

Versión	Refactorizaciones
Eliminadas las copias de accounting de printDocument	Extract method accounting
Eliminadas las copias de logging de requestWorkstationPrintsDocument y requestBroadcast	Extract method logging
Movido el método logging a Node	Move method logging
Movido el método printDocument a Packet	Move method printDocument Change method public void accounting
Creado del método atDestination para los bucles de navegación	Extract method atDestination
Eliminados los bucles de navegación de Network	Extract method atRegistering Extract circleNetwork
Movidos los métodos de impresión a la clase Node	Extract method printOnSwitch Extract method printHTMLOnSwitch Extract method printXMLOnSwitch Move method printOnSwitch Move method printHTMLOnSwitch Move method printXMLOnSwitch
Creados métodos para las condiciones de tipo para la impresión y pasados a las subclases correspondientes	Extract method printXMLOnAppend Extract method printXMLOnAppendWorkStation Extract method printXMLOnAppendPrinter
Eliminados los bucles de navegación de Node	Extract method atLastNode Extract method sendPrintOf Extract method sendPrintHTMLOn Extract method sendPrintXMLOn

4.3 Preguntas

¿Cuál es tu primera impresión sobre el sistema? ¿Dónde centrarías tus esfuerzos de refactorización? Discutir con los miembros del equipo.

Es un sistema con una clase principal que se encarga de toda la funcionalidad y otras clases que funcionan como entidades

Centraríamos nuestros esfuerzos en la clase principal Network para añadir las nuevas funcionalidades y en Node para los nuevos tipos

¿Cuál es la segunda impresión sobre el sistema? ¿Estas de acuerdo con la impresión inicial? Con este nuevo conocimiento sobre el código? ¿dónde centrarías tus esfuerzos de refactorización? Discutir con los miembros del equipo.

La segunda impresión viendo el código confirma la primera de que la clase Network es la que realiza toda la funcionalidad y que las otras clases funcionan como entidades.

Nuestros esfuerzos para refactorizar el código se deberían centrar en la clase Network para poder incluir la nuevas formas de impresión de la lista de tareas y en la clase Node para poder añadir los nuevos tipos de nodos que se añadirán en futuras versiones.

¿Crees que el código base esta ya refactorizado? ¿Qué puedes decir de la calidad de los tests: puedes empezar a refactorizar de manera segura? ? Discutir con los miembros del equipo.

No. A primera vista se ve código repetido y funcionalidad que debería estar en otras clases.

En cuanto a los test su cobertura se queda pequeña al no llegar al 90% para las clases que vamos a modificar, lo que puede suponer un riesgo a la hora de modificar el código y asegurarnos de que funciona correctamente.

Desarrolla un plan de proyecto listando a)los riesgos, b) las oportunidades de refactorización (detección de defectos), c) las actividades (plan de refactorizaciones).

Riegos: El mayor riesgo es modificar el código que funciona correctamente y crear un código incorrecto que no funcione

Oportunidades de refactorización: eliminar el código duplicado, eliminar los condicionales switch y sustituirlos por clases tipo, añadir clases tipo a Node

Actividades: Utilizar Extract method para eliminar el código duplicado; mover el código de la clase Network que no pertenece a esa clase a las clases correspondientes usando State/Strategy; modificar la clase Node para permitir subclases que permitan

añadir código correspondiente a la subclases de cada tipo de nodo

¿Estas seguro que estas refactorizaciones no rompen el código?¿Crees que estas refactorizaciones merecen la pena? ¿La herramienta de refactorización hace un buen trabajo? Discutir con los miembros del equipo.

- **Extract Method:** Si estamos seguros porque no modificamos el código en si. Merece la pena porque elimina el código duplicado y permite mejor mantenimiento. La herramienta de refactorización hace un buen trabajo
- **Mover el comportamiento cerca de los datos:** El problema que puede surgir es debido a la visibilidad de los métodos, que pueden no ser accesibles y romper el código. Si que merece la pena porque eliminamos referencias a otras clases innecesarias. La herramienta hace en general un buen trabajo pero a veces no permite elegir el destino deseado.
- **Eliminar código de navegación:** Para asegurarnos de que no rompen el código debemos hacer los tests puesto que modificamos el flujo de los bucles. La herramienta hace un buen trabajo extrayendo los métodos pero los métodos recursivos nuevos tenemos que hacerlos nosotros.
- **Transformar códigos de tipo:** Estas modificaciones rompen el código y hay revisarlo y modificarlo para que funcione puesto que modificamos como funciona el programa al eliminar el atributo tipo. La herramienta ayuda a mover los métodos a las nuevas clases pero el resto debemos hacerlo nosotros.

¿Hay asuntos que no has considerado? ¿Hay refactorizaciones que parecen innecesarias? Discutir con los miembros del equipo.

Refactorizaciones como la de eliminar los bucles de navegación is parecen innecesarias porque no tienen relación con las futuras versiones planeadas, pero ayudan al mantenimiento del código y a su lectura.

5 PREGUNTAS DE REFLEXIÓN

- **¿Se puede automatizar completamente el proceso de refactorización a través de herramientas?**

No totalmente pero si una gran parte y ayudan a reducir la cantidad de errores en el proceso y a llevar un control de los cambios realizados.

- **¿Qué relación encuentras entre el proceso de refactorización y la utilización de sistemas de control de tareas y versiones?**

La relación es directa porque permite llevar un control de los cambios realizados partiendo de un código inicial que funciona pero que puede tener defectos hacia un código donde se van eliminando esos defectos y termina siendo un código de calidad.

6 CONCLUSIONES SOBRE LA REALIZACIÓN DE LA PRÁCTICA

Las conclusiones que sacamos es que el proceso de refactorización esta en gran parte automatizado y que permite eliminar muchos defectos del código de forma automática. Esto permite transformar nuestro código en uno mas mantenible y que permita futuros cambios y modificaciones más fácilmente. Ademas, las herramientas de refactorización permiten un seguimiento de los cambios realizados a nuestro proyecto.