Facultad de Ciencias Exactas, Físicas y Naturales

Ingeniería de Software - 2020

Plan de Gestión de las configuraciones

Grupo CodeRush

Integrantes:

- Molina Franco Elías.
- Nievas Boso Leonel Federico.
- Orionte Andrés.
- Palacios Lucero Matías Iván.

ÍNDICE

1. Introducción

1.1 Propósito y alcance

En el siguiente documento se detalla el Plan de Gestión de Configuraciones para el proyecto de software a realizar, de ahora en adelante “Arkanoid”. El propósito de este texto es detallar la configuración de los requisitos, su documentación y herramientas utilizadas para la elaboración del proyecto.

Se define cómo procesar los cambios propuestos al sistema, se decide qué componentes del sistema modificar, cómo gestionar las distintas versiones y cómo distribuir las distintas versiones a los clientes.

1.2 Propósito del plan

- Definir las competencias de los distintos integrantes del grupo.
- Coordinar los módulos del proyecto.
- Ningún cambio pierda su motivo.
- Asegurar la consistencia entre las distintas partes del proyecto.
- Crear un historial de cambios al proceso.

1.3 Referencias, acrónimos y glosarios

Acrónimo o palabra	Descripción
CM	Del inglés, configuration management, gestión de la configuración
IDE	Del inglés, integrated development environment, entorno de desarrollo integrado. Herramienta para el desarrollo del código fuente del programa
Repositorio	Lugar donde se almacenan los objetos bajo el control de configuración i.e. datos actualizados e históricos. Este puede ser a nivel local o en un servidor en la nube.
CCC	Comité de control de cambios

1.4 Herramientas de gestión de configuración

Herramienta	Descripción	Enlace
Java	Lenguaje de programación	https://www.java.com/
Eclipse Java	IDE utilizado	https://www.eclipse.org/

Travis	Herramienta de integración contínua	https://travis-ci.org/
Gradle	Herramienta de automatización	https://gradle.org/
Visual Paradigm	Herramienta de Diagramas UML.	https://online.visual-paradigm.com/
Arkanoid_ repositorio	Se define el repositorio donde está alojado el proyecto con su código fuente actual e histórico.	https://github.com/CodeRush2020/Arkanoid
Arkanoid_ integración contínua	Cuando se haga cambios al proyecto, esta herramienta realizará pruebas para comprobar que no se hayan introducido errores.	https://github.com/CodeRush2020/Arkanoid/actions
Arkanoid_ issues	Poder notificar de alguna cuestión pertinente al proyecto e.g. pendientes, mejoras, errores,etc.	https://github.com/CodeRush2020/Arkanoid/issues

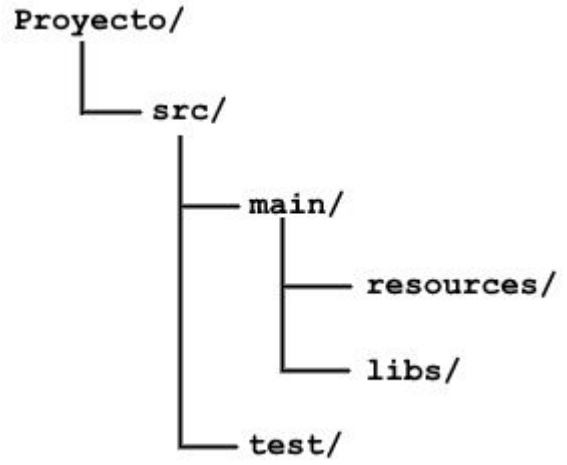
2. Gestión del Proyecto

2.1 Administración de la Configuración del Proyecto

Las actividades en cuanto a la gestión y/o administración de las configuraciones se llevan a cabo y se coordinan por los cuatro integrantes del grupo, todo el grupo está en el mismo nivel jerárquico, de la misma forma, las responsabilidades serán compartidas por todo el grupo.

2.2 Esquemas de los directorios

Se opta por un esquema de directorios clásico en los programas en Java.



Ruta	Propósito
./	La dirección de todo el proyecto
./src/	El directorio del programa
./src/main/	Clases del programa
./src/main/resources/	Audiovisuales tales como imágenes y sonidos
./src/main/libs/	Librerías adicionales implementadas
./src/test/	Clases de pruebas para la integración contínua
./docs/	Directorio de documentos asociados
./docs/images	Imágenes asociadas al proyecto
./docs/diagrams	Diagramas asociados al proyecto
./docs/exe	Archivos ejecutables del proyecto
./test/	Directorio para archivos de testing del proyecto
./changes/	Directorio de los cambios a realizar
./changes/rejected	Cambios descartados por la CCB
./changes/pending	Cambios pendientes a revisión por la CCB
./changes/approved	Cambios aprobados por la CCB

2.3 Normas de Etiquetado y Versionado de Archivos

Para el nombramiento de archivos se utiliza un sistema de tres dígitos más un cuarto dígito como opcional separados por puntos:

primer.segundo.tercero.cuarto

Cada uno de los dígitos tiene el siguiente significado:

- Primer: versión principal del software, abarca un conjunto de funcionalidades principales.
- Segundo: indica una funcionalidad menor y específica del software.
- Tercero: revisión de la versión, ajustes y corrección de fallos.
- Cuarto: número de fallos por no cumplir con los requisitos.

Todos estos números comenzarán con el 0 e incrementarán a medida que se avance en el desarrollo del software.

3. Gestión de la Configuración del Código

3.1 Plan de Esquema de Ramas a Usar

En el proyecto se hará uso de 4 ramas, cada una con un fin específico el cual es detallado a continuación:

- Master: Es la rama principal, en donde se encontrará el código del software sin errores. La mayoría de las fusiones de archivos se realizarán sobre esta rama, y es donde se colocaran las versiones a lanzarse.
- Develop: Esta rama se implementa para los códigos de desarrollo de nuevas funcionalidades y características. Aquí es donde ocurre la mayor parte del trabajo de desarrollo.
- Hotfix: Aquí se corregirán rápidamente los errores indeseados que ocurran en los archivos de la rama *Master*.
- Modification: Esta rama sera utilizada para las modificaciones que se realicen sobre la documentación del proyecto, como por ejemplo diagramas o requerimientos. La identificación de la rama deberá corresponder al documento en cuestión.

3.2 Política de Etiquetado

La rama principal del repositorio va a contener el código a entregar. Cada vez que se solicite agregar funcionalidades creará una nueva rama que deberá cumplir con los requerimientos pedidos, una vez finalizado el trabajo en esta rama y verificación de que todo funcione, se agregará a la rama principal. Al hacer esto, se incrementa el valor del segundo dígito de la versión.

Es posible que, cuando una rama se incorpore a la principal, se hayan agregado varias ramas en el proceso de desarrollo de esta. Si esta rama agregada trata únicamente de corregir errores, se podrá reconstruir su versión anterior que corrige ese error en particular sin agregar las ramas agregadas y se incrementa el tercer dígito de la versión. De optarse por incluir estos cambios en la nueva versión, también se incrementará el valor del tercer dígito.

Cuando el proyecto haya cumplido cierto conjunto de requerimientos y funcionalidades con un nivel de calidad aceptable, se incrementará el dígito primario y el resto volverán a cero. Se espera que la versión 1.0.0.0 sea la versión de lanzamiento del software, en esta versión se combinarán todos los archivos del proyecto porque se considera que ya ha alcanzado su estado estable.

3.3 Política de Fusión de Archivos

Para fusionar las ramas con el código principal nuevamente, el administrador realiza la fusión haciendo un rebase del código junto a los desarrolladores, de esta forma rápidamente nos damos cuenta si hay errores y los desarrolladores pueden trabajar en

resolverlos. Además, el administrador estaría al tanto de la situación para coordinar otro eventual cambio en el código.

En caso de que la fusión se realice de forma correcta, se procede a un etiquetado de acuerdo a la política de etiquetado y versionado de archivos ya descritas en el ítem 3.2.

4. Comité de Control de Cambios

4.1 Propósito

El CCC es una junta que se encargará de tomar la decisión de aprobar o no la implementación de cambio para el software. El proceso de cambio deberá ser solicitado con una notificación y documentación de solicitud de cambio. Luego se tomará la decisión si el cambio se aplica o no. En la documentación de solicitud de cambio se debe incluir:

- Definir cuál parte del software cambiar.
- Explicar el por qué hacer dichos cambios.
- Describir detalladamente los cambios propuestos.
- Proponer fecha límite para la implementación.

4.2 Miembros y funciones

El CCC está integrado por los cuatro miembros del equipo. Se harán reuniones para comunicarse las propuestas, analizarlas, evaluar las implicancias de su implementación o no implementación y finalmente la toma de decisión. La decisión se hará por voto de los miembros del equipo y para que dicho cambio se realice, se requiere al menos la aprobación de tres de los cuatro integrantes del grupo.

Integrantes del CCC:

Rol	Integrante
Consejero	Molina Franco Elías
Analista de propuestas	Nievas Boso Leonel
Vocero de la junta y árbitro de sesiones	Orionte Andrés
Evaluador de riesgos e implicaciones	Palacios Iván

La junta se reunirá cada vez que se solicite un cambio en el proyecto y se hará vía conferencia online. La junta también tendrá una reunión semanal, en un horario acordado entre los miembros, con el fin de mantenerse comunicados y al tanto de las distintas cuestiones.

4.3 Proceso de control de cambios

Una vez presentada la propuesta de cambio, cumpliendo lo establecido en el punto 4.1, dicha propuesta se someterá al siguiente proceso para su correcta evaluación:
1. Definir si la información recibida es suficiente para la evaluación global del cambio.
2. Identificar las áreas afectadas por el cambio requerido.
3. Detallar costos estimados, fechas de apropiación y estimación de la implementación.
4. Convocar al CCC para el análisis económico y evaluación integro del cambio requerido.
5. Si es aceptado por CCC, se lleva a cabo el desarrollo del cambio a través de un documento escrito para los desarrolladores.
6. Si no es aceptado por CCC, se promueve el cambio al archivo de requerimientos pendientes.

5. Herramienta de Seguimiento de Defectos

Para el reporte de defectos y su seguimiento, hemos utilizado la herramienta que nos proporciona GitHub para dicho propósito, denominada *Issues*. A la misma se accede desde la página del proyecto en GitHub, e ingresando a la pestaña Issues, siempre y cuando se cuente con los permisos necesarios. Por seguridad y privacidad, el permiso de acceso está restringido a los colaboradores del repositorio, es decir a los 4 integrantes del grupo.

La dirección es la siguiente: https://github.com/CodeRush2020/Arkanoid/issues

El propósito de dicha herramienta es reportar rápidamente los defectos encontrados en el desarrollo del proyecto, explicando el mismo con el mayor detalle posible, y si es posible una solución al mismo. Una vez que el defecto fue solucionado, se lo cierra aclarando que ya fue resuelto, y se dejan activos aquellos que aún no fueron resueltos.

6. Administración de Releases

Los releases comienzan a construirse una vez conocidos los requerimientos, dependencias y plazos sobre los cuales se va a trabajar. Una vez que se cumple con el requisito para el release, se lo coloca en una carpeta comprimida en formato “.zip”, con el fin de facilitar la instalación y la distribución del mismo. También se debe realizar un tag en el repositorio para marcar la versión utilizada.

Luego la build se somete a una prueba de aceptación, en donde se verifica que cumpla con los requerimientos y esté libre de errores para ser entregada al usuario.

6.1 Documentación del release

En cada release se debe proporcionar la siguiente información:

- Tag de la entrega
- Documentación de fallas corregidas
- Documentación
- Informe de pruebas