Universidad Peruana de Ciencias Aplicadas

UPC



**Tarea 4: Mantenimiento y extensión de tinyUML**

Ingeniería de Software - CC52

**Integrantes:**

Renzo Perez

Daniel Nuñez

Andrés Pautrat

Sergio Villarruel

José Talavera

Lima, 2019

**1. Solicitud de cambio**

Se pide implementar nuevas funcionalidades al software tinyUML, enfocada en funciones de calidad de vida al software en general.

* Modificar el botón File -> New para la creación de diagramas, de tal manera que cada diagrama nuevo tenga enumeración.
* Implementar un nuevo botón File -> Close All con su propio código para el cierre de todos los diagramas. Además, configurar el comportamiento del mismo.
* Implementar un nueva función para enumeración de las clases, paquetes, componentes y diagramas.
* Implementar una nueva opción para Zoom con el tamaño de 200% para los usuarios con pantallas grandes.
* Implementar el conteo de clases, componentes, paquetes y notas. Denotar el número de componente de su respectiva clase si este lo tiene.
* Implementar la inclusión del objeto “Actor” para el diagrama.

**2. Ubicación de concepto**

Se ubicaron los archivos, clases, funciones, etc. que vayan a ser requeridas para la implementación de las funcionalidades mediante los siguientes procesos:

* Búsqueda y entendimiento manual del código.
* Buscada asistida con el comando Ctrl+Shift+f.
* Ubicación de funciones relacionadas (go to Declaration y Find Usages)
* Debug.

La información que fue encontrada fue analizada y entendida para la correcta manipulación del código en la agregación de botones y en la implementación de funcionalidades.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Nombre del Archivo** | **Herramienta Usada** | **¿Ubicado?** | **Comentarios** |
| 1 | Main | Exploración Manual | No Relacionado | Inicia todas las clases |
| 2 | UI | Exploración Manual | Relacionado | Clase encargada de la interfaz de usuario |
| 3 | MenuManager | Exploración Manual | Relacionado | Controla el menú del usuario |
| 4 | AppFrame | Exploración Manual | Relacionado | Controla los eventos de toda la aplicación |
| 5 | MainToolbarManager | Exploración Manual | Relacionado | Maneja la barra de herramientas mostrada al usuario |
| 6 | Scaling | Búsqueda asistida | Relacionado | Maneja las definiciones de escalamiento en general. |
| 7 | EditorCommandDispatcher | Búsqueda asistida | Relacionado | Recibe comandos de aplicación relacionados a la edición y los envía a los lugares correctos. |
| 8 | DiagramElementFactoryImpl | Búsqueda asistida | Relacionado | Maneja la creacion de modulos dentro del diagrama. |
| 9 | StructureDiagram | Find Usages | Relacionado | Se encarga de gestionar la estructura dentro del diagrama. |
| 10 | CreationHandler | Find Usages | Relacionado | Se encarga de la creacion de modulos y los almacena. |

**3. Análisis de impacto**

Ya que las solicitudes que se desean implementar no son de gran complejidad, se intenta que las modificaciones realizadas sean lo más mínimas posibles de modo que el impacto sea el menor a nivel de código.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Nombre del Archivo** | **Herramienta Usada** | **¿Impacto?** | **Comentarios** |
| 1 | Main | Exploración Manual | No | Inicia todas las clases |
| 2 | UI | Exploración Manual | No | Clase encargada de la interfaz de usuario |
| 3 | MenuManager | Exploración Manual | Sí | Controla el menú del usuario |
| 4 | AppFrame | Exploración Manual | Sí | Controla los eventos de toda la aplicación |
| 5 | MainToolbarManager | Exploración Manual | Sí | Maneja la barra de herramientas mostrada al usuario |
| 6 | Scaling | Búsqueda asistida | Sí | Maneja las definiciones de escalamiento en general. |
| 7 | EditorCommandDispatcher | Búsqueda asistida | Sí | Recibe comandos de aplicación relacionados a la edición y los envía a los lugares correctos. |
| 8 | tinyuml-captions.properties | Debug | Sí | Archivo de recursos de comandos para la interfaz gráfica |
| 9 | tinyuml-captions\_de\_DE.properties | Debug | Sí | Archivo de recursos de comandos para la interfaz gráfica |
| 8 | DiagramElementFactoryImpl | Debug | Si | Maneja la creacion de modulos dentro del diagrama |
| 9 | StructureDiagram | Exploracion manual | No | Se encarga de gestionar la estructura dentro del diagrama |
| 10 | CreationHandler | Debug | Si | Se encarga de la creacion de modulos y los almacena |
| 11 | LineHandler | Explotacion manual | No | Manejo de datos |
| 12 | DiagramEditor | Exploracion Manual | No | Maneja la edición de los diagramas |
| 13 | StructureDiagram | Exploracion Manual | No | Maneja la estructura del diagrama |

**4. Prefactoring**

El primer paso para realizar las modificaciones es la identificación de partes relacionadas del código, para nosotros estas partes fueron:

* Menu File, para la creación de los botones.
* Funciones de eliminación de diagramas.

A continuación se implementan los botones en el menú file, que al ser desplegado mostrará tanto el botón New y el botón Close All. Seguido a esto se implementan las funcionalidades de cada botón.

La implementación de lo anterior es la simple agregación de pocas líneas de código independientes dentro del menú file, por lo cual esto no deberia tener ningun inconveniente en otras partes del código, botones o menús. Y no afecta el código de otras funcionalidades que posee el programa.

* Funciones de re-escalamiento de vistas.
* Función de asignación de características a los módulos.
* Funcion de boton de creacion de modulos.
* Función de botón de creación de actores

A continuación se cambio la función de asignación de características a los módulos para que reciba un parámetro numérico que poner al final del nombre del módulo que este siendo creado. Seguido a esto, modificar las funciones que son intermediarias hasta llegar a la clase donde maneja los módulos creados. Luego se modifica la clase que maneja las módulos creados para que lleve la cuenta de cuántos módulos de cada clase han sido creados, y según esto envíe un número a la función de asignación de características.

Para la creación de actores se requirió de obtener una imagen ícono respectiva que represente al objeto, esta es agregada y validada en las propiedades del programa. Y se crearon las clases respectivas para su implementación.

**5. El proceso de aprendizaje**

Las siguientes clases han sido modificadas para lograr el objetivo de nuestras nuevas funcionalidades en el programa:

* MainToolbarManager: Se agregan los botones.
* MenuManager: Se agregan los botones.
* AppFrame: Se asignan funcionalidades a los botones.
* Scaling: Se define el escalamiento de vistas.
* EditorCommandDispatcher: Se asignan los métodos de las funciones.
* DiagramElementFactoryImpl: Se modifica la función que asigna los nombres a los módulos
* CreationHandler:Se agregan 4 contadores, uno para cada modulo. Se modifica una función para que identifique cuántos módulos de cada clase han sido creados y pasa este valor a la función que asigna nombres.
* LineHandler: Debido a modificaciones de otras funciones, debe agregarse un valor cualquiera sin repercusiones dentro del código.
* DiagramEditor:Debido a modificaciones de otras funciones, debe agregarse un valor cualquiera sin repercusiones dentro del código.
* StructureDiagram:Se modifica una función para recibir un valor numérico necesario en una función que contiene y ha sido modificada.
* DrawingContext: Se agrega una función para dibujar un circulo
* ElementType: Se agrega el tipo Actor
* StaticStructureEditorToolbarManager: Se agrega el botón de creación de actor

**6. Actualización**

Para la implementación de las funcionalidades planteadas se crean los botones específicos y se les asignan las acciones a los mismos.

**Implementación de enumeración de diagramas nuevos:**

|  |  |  |
| --- | --- | --- |
| **Nombre de Cambio** | **Lugar** | **Codigo** |
| **Se agrego un selectormap en initSelectorMap** | **AppFrame** | selectorMap.put("CLOSE", new MethodCall(  getClass().getMethod("close"))); |
| **Se agregó una función para cerrar todos los diagramas.** | **AppFrame** | public void close(){  tabbedPane.removeAll();  } |
| **En addFileButtons se agrego un botón llamado close** | **MainToolbarManager** | createButton("close"); |
| **En createFileMenu se agrego un item de close** | **MenuManager** | createMenuItem(fileMenu, "close"); |
| **En new model se agrego un contador y se borro la función de removeAll() para que se pueda agregar más diagramas en la pestalla** | **AppFrame** | public void newModel() {  if (canCreateNewModel()) {  umlModel = new UmlModelImpl();  StructureDiagram diagram = new StructureDiagram(umlModel);  umlModel.addDiagram(diagram);  diagram.setLabelText("Class diagram "+ cont);  //tabbedPane.removeAll();  cont++;  createEditor(diagram);  }  } |

**Implementación de Zoom 200%**

|  |  |  |
| --- | --- | --- |
| **Nombre de Cambio** | **Lugar** | **Codigo** |
| **Definición del escalamiento** | **Scaling.java** | **public static final Scaling SCALING\_200 = new Scaling(2.0);** |
| **Se agrega la función de zoom 200 al selector map** | **EditorCommandDispatcher.java** | **selectorMap.put("ZOOM\_200", new MethodCall(**  **DiagramEditor.class.getMethod("setScaling", Scaling.class),**  **Scaling.SCALING\_200));** |
| **Se agrega al menu ‘View’ la nueva opción de zoom** | **MenuManager.java** | **viewMenu.add(zoom200);**  **group.add(zoom200);** |

**Implementación de módulos numerados**

|  |  |  |
| --- | --- | --- |
| **Nombre de Cambio** | **Lugar** | **Codigo** |
| **Modificacion de asignación de nombres.** | **DiagramElementFacotryImpl.java** | **private void setupElementPrototypeMap(int num) {**  **…**  **pkg.setName("Package "+Integer.toString(num));**  **…**  **comp.setName("Component "+Integer.toString(num));**  **…**  **clss.setName("Class "+Integer.toString(num));** |
| **Creacion de contadores para cada uno de los 4 módulos.** | **CreationHandler.java** | **private int CountClass =1;**  **private int CountPackage =1;**  **private int CountComponent =1;**  **private int CountNote =1;** |
| **Udebtufucacuib dek tuoi de modulo siendo creado y llamado a la función de creación con su nuevo parámetro extra.** | **CreationHandler.java** | **if(type.equals(ElementType.CLASS)){**  **element = editor.getDiagram().getElementFactory(CountClass).createNode(elementType);**  **CountClass++;**  **}**  **else if(type.equals(ElementType.PACKAGE)){**  **element = editor.getDiagram().getElementFactory(CountPackage).createNode(elementType);**  **CountPackage++;**  **}**  **else if(type.equals(ElementType.COMPONENT)){**  **element = editor.getDiagram().getElementFactory(CountComponent).createNode(elementType);**  **CountComponent++;**  **}else if(type.equals(ElementType.NOTE)){**  **element = editor.getDiagram().getElementFactory(CountNote).createNode(elementType);**  **CountNote++;**  **}** |

**Implementación de la creación del objeto “Actor”:**

|  |  |  |
| --- | --- | --- |
| **Nombre de Cambio** | **Lugar** | **Codigo** |
| **Editor del objeto** | **EditorCommandDispatcher** | **selectorMap.put("CREATE\_ACTOR", new MethodCall(**  **DiagramEditor.class.getMethod("setCreationMode", ElementType.class),**  **ElementType.ACTOR));** |
| **Implementación de la creación del objeto** | **DiagramElementFacotryImpl** | **UmlActor act = (UmlActor) UmlActor.getPrototype().clone();**  **act.setName("Actor 1");**  **ActorElement actElem = (ActorElement) ActorElement.getPrototype().clone();**  **actElem.setModelElement(act);**  **actElem.addNodeChangeListener(diagram);**  **elementPrototypes.put(ElementType.ACTOR, actElem);** |
| **Implementación del botón para la creación del objeto** | **StaticStructureEditorToolbarManager** | **toolbar.add(createToggleButton(buttongroup, "actor"));** |
| **Clase Prototype** | **Se creo la clase UmlActor** |  |
| **Clase principal del objeto** | **Se creo la clase ActorElement** |  |

**7. Postfactoring**

Al actualizar y crear estas nuevas funcionalidades, se tuvo un problema ya que estaba programado, pero no funcionaban las funcionalidades. Por ello, se requiere agregar 3 comandos en el archivo de propiedades tinyuml-captions. Además, para cerrar los diagramas se tuvo el mismo problema que el anterior, donde se cambio el archivo propiedades de tinyuml-captions\_de\_DE . Además, se tuvo que modificar una función en AppFrame la cual es getCurrentEditor para que tinyUml se pudiera editar en los distintos diagramas agregado después de cerrarlos.

**Para Implementación de enumeración de diagramas nuevos:**

|  |  |  |
| --- | --- | --- |
| **Se agrego el comando para que se cierre el programa, el nombre del botón y su título.** | **tinyuml-captions.properties** | menuitem.close.command=CLOSE  menuitem.close.name = Close all ...  dialog.close.title= Close all... |
| **Se agrego el comando para que se cierren los diagramas** | **tinyuml-captions\_de\_DE.properties** | menuitem.close.command=CLOSE |
| **Se cambió el método para que se pueda editar en los diagramas.** | **Appframe** | public DiagramEditor getCurrentEditor() {  return currentEditor;  } |

**Para implementación de Zoom:**

|  |  |  |
| --- | --- | --- |
| **Se agregó el nombre y el comando para la nueva función de zoom** | **tinyuml-captions.properties** | **menuitem.zoom\_200.name=Zoom 200 %**  **menuitem.zoom\_200.command=ZOOM\_200** |
| **Se agregó el nombre y el comando para la nueva función de zoom** | **tinyuml-captions\_de\_DE.properties** | **menuitem.zoom\_200.name=Zoom 200 %**  **menuitem.zoom\_200.command=ZOOM\_200** |

**Para Implementación de la creación de actores:**

|  |  |  |
| --- | --- | --- |
| **Se agrego el comando para la creación del actor** | **tinyuml-captions.properties** | statictoolbar.actor.icon=ACTOR  statictoolbar.actor.command=CREATE\_ACTOR  statictoolbar.actor.tooltip=Create actor |
| **Se añadió el tipo ACTOR y se agregó la ruta de la imagen del icono** | **IconLoader** | public enum IconType {  NEW, OPEN, SAVE, CUT, COPY, PASTE, DELETE, UNDO, REDO,  ABOUT,  MOUSE\_POINTER,  CLASS, PACKAGE, COMPONENT, DEPENDENCY, ASSOCIATION, AGGREGATION, ACTOR,  COMPOSITION, INHERITANCE, INTERFACE\_REALIZATION, NOTE, NOTE\_CONNECTOR  }  urlMap.put(IconType.ACTOR, "org/tinyuml/ui/actor.png"); |
| **Se creó una función para dibujar un circulo** | **DrawingContext** | public void drawCircle(double x,double y, double width, double height)  {  g2d.setStroke(shapeFactory.getStandardStroke());  g2d.draw(new Ellipse2D.Double(x, y, width, height));  } |

**8. Conclusión**

La intervención realizada en el sistema se concentró en varias tareas: Enumerar diagramas nuevos, poder cerrar todas los diagramas abiertas con un botón, implementar una nueva función de zoom, implementar la creación de actores. Para completar ello, se debió acceder a 5 clases, ya que cada una estaba utilizada en la anterior. Se hizo el mayor cambio en la clase AppFrame, en donde acá se agrego las distintas funcionalidades requeridas.

Una vez agregadas los funcionamientos dichos, se debió acceder a unos archivos de propiedades para poder habilitar dichas funcionalidades. Además, se puede mejorar estas funcionalidades, implementando el cerrado individual de pestañas. Otra mejora posible para el software sería que el usuario pueda determinar el zoom que él desee para su pantalla al escribirla. Finalmente, se logró implementar satisfactoriamente las funcionalidades propuestas.

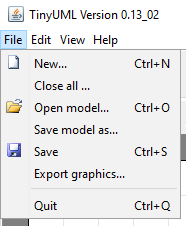
**9. Fuentes**

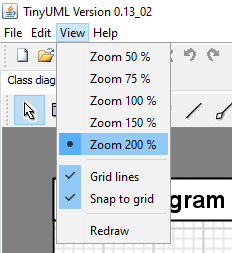
Se utilizó la última versión de la documentación de:

* Java

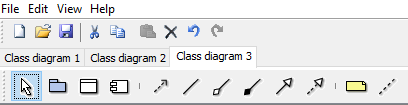
**10. Anexos**

**Botones:**

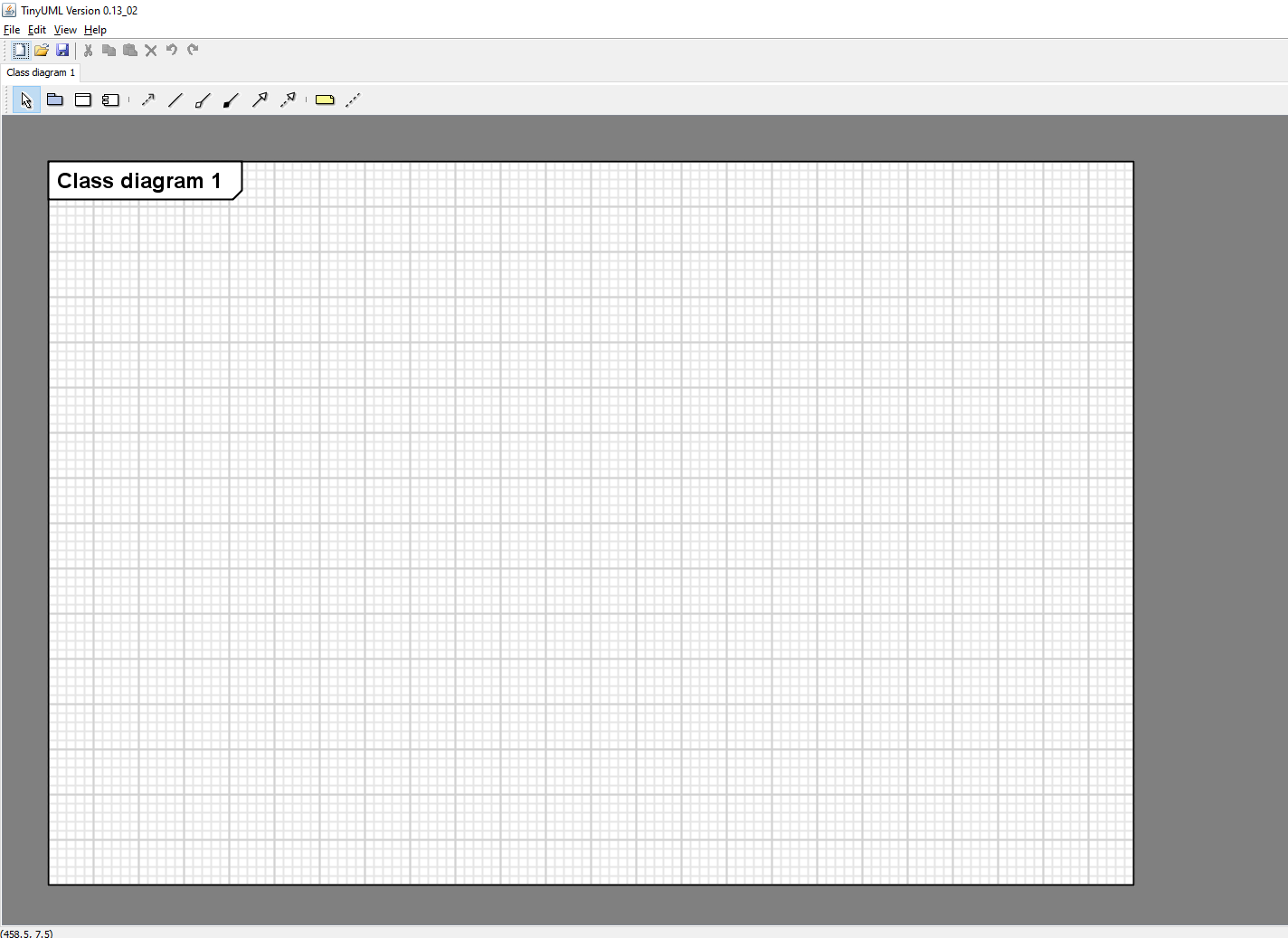
****

****

**Pestañas creadas:**

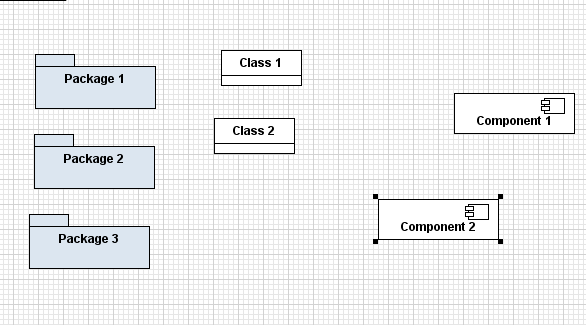
****

**Vistas nuevas:**



En zoom 200% para pantalla 1920x1080

**Enumeración de módulos:**



**Creación de actores:**

