MagicUWE4R

Una herramienta de refactoring en el modelado de aplicaciones Web

Miguel Djebaile

Tesina de Licenciatura
Directora: Alejandra Garrido
Co-Director: Gustavo Rossi

Contenido

- Introducción
- Trabajos Relacionados
- Arquitectura base
- Arquitectura de MagicUWE4R
- Refactorings en el modelo de navegación
- Refactorings en el modelo de presentación
- Conclusiones

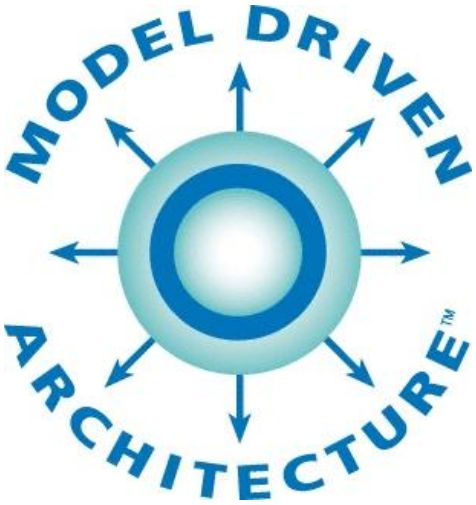
Introducción

- Presentación
  - Escenario actual
  - Es necesario hacer refactoring (constante cambio – metodologías ágiles)
  - Model Driven Development (nueva tendencia, UWE, OOHDM)
- Motivación
  - La importancia de la Usabilidad de las aplicaciones Web en el core de un negocio
  - Adaptaciones en la usabilidad = Refactoring
  - Refactoring + MDD = Refactoring de Modelos
- Objetivo
  - Refactoring + MDD = MagicUWE4R
  - Composición de refactorings
  - Motor de refactorings extensible

Trabajos Relacionados

- Model Driven Architecture (MDA)
- Refactoring
- Refactoring en Aplicaciones Web para mejorar Usabilidad
- UWE

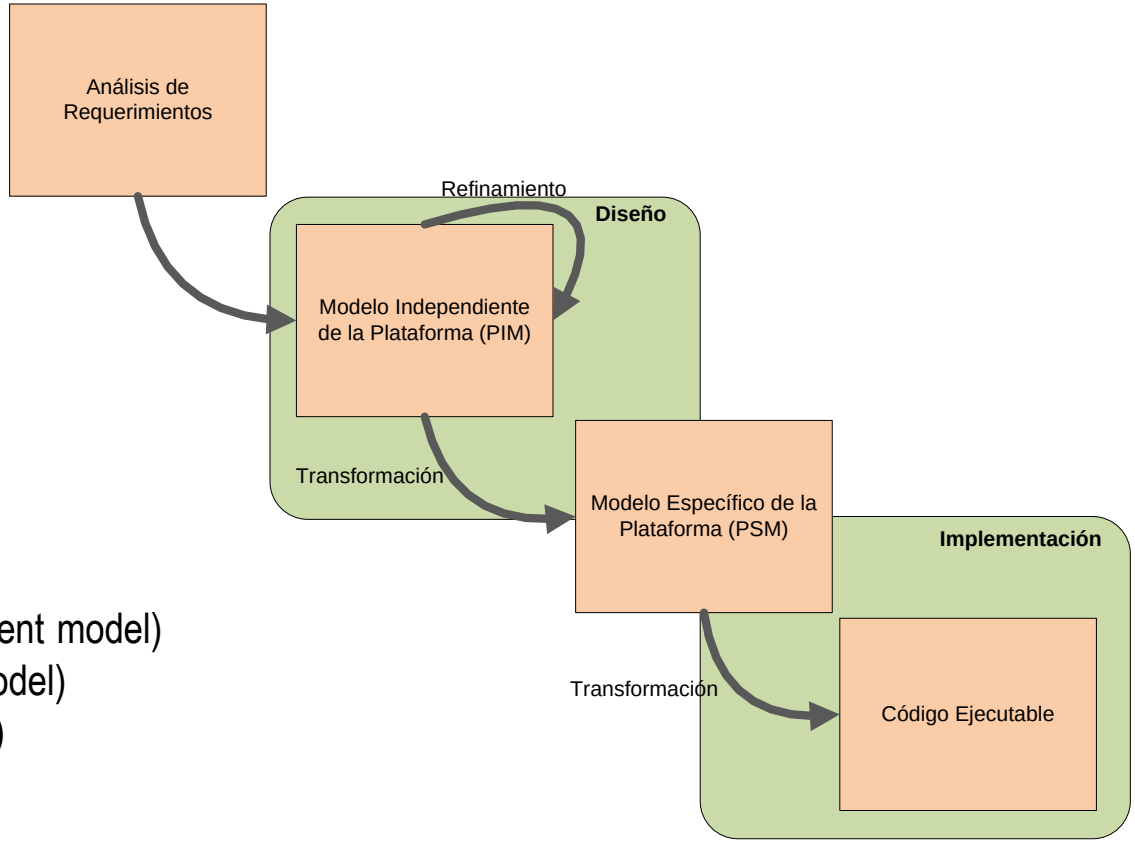
Model Driven Architecture



- OMG (Object Management Group)
  - Propuso MDA
  - MDA como implementación de MDD
- Descripción
  - Basado en UML (estándar)
  - Portabilidad
  - Interoperabilidad (basado en modelos de alto nivel)
- Proceso MDA
  - Transformaciones (generan un nuevo modelo)
  - Refinamientos (dentro de un mismo modelo)
  - Separación de concerns (requerimientos tecnológicos y del negocio)
  - Evolución en paralelo

Model Driven Architecture

- Proceso MDA

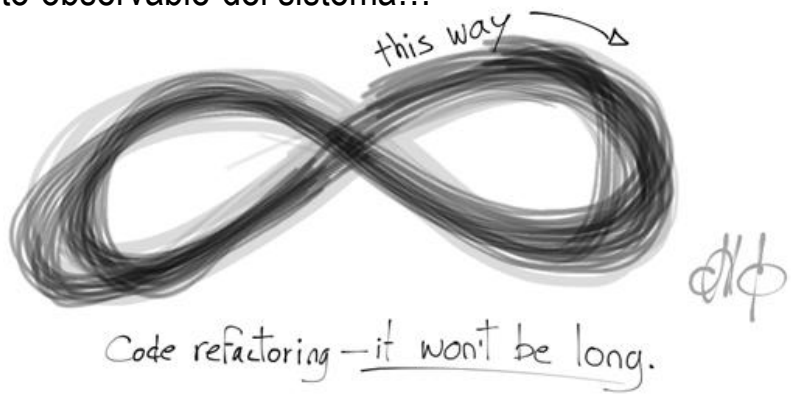


- Tipos de Modelos en MDA
  - CIM (Computational-independent model)
  - PIM (Platform-independent model)
  - PSM (Platform-specific model)
  - Código Fuente.

Refactoring

- Martin Fowler lo define como:
  - "… un cambio en la estructura interna del software para hacerlo más fácil de entender y menos costoso de modificar, sin cambiar el comportamiento observable del sistema…"
  - "… reestructuración de software mediante la aplicación de una serie de refactorings sin cambiar el comportamiento observable del sistema…"

- Características
  - Sobre la estructura interna
  - Preserva el comportamiento
  - Implica una mejora de un req. no funcional
  - Conjunto de pasos atómicos que pueden combinarse
  - Por desarrolladores



- Ventajas
  - Diseño
  - Escalabilidad y detección de errores
  - Mantenimiento y Legibilidad (por eso empleado en SCRUM, TDD, etc)

Refactoring en Aplicaciones Web para mejorar Usabilidad

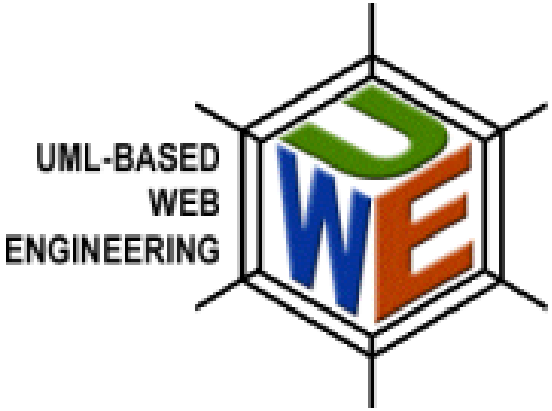
- Clave en el core de un negocio
- Factores
  - Accesibilidad (para todas las capacidades)
  - Navegabilidad (facilidad de acceso a todos los recursos)
  - Eficiencia (velocidad)
  - Credibilidad (confianza)
  - Intuición (predicción)
  - Personalización

→ Aplicables mediante Refactoring

- Investigación en Patrones para mejorar Usabilidad
  - En ellos se basa MagicUWE4R
  - Add Link, Split Node Class
  - Autocomplete, BreadCrumbs

UWE

- UML-Based Web Engineering
- Características
  - UML (perfiles y estereotipos)
  - Procesos bien definidos
  - Acepta OCL



- Modelos de UWE
  - Requerimientos (casos de uso)
  - Modelo Conceptual (clases, asociaciones, variables y métodos)
  - Modelo Navegacional (nodos, links, atributos y operaciones)
  - Modelo de Presentación (páginas, widgets)

Arquitectura Base

- Caso de estudio: MagicDraw OpenAPI
- Extensibilidad: Plugins
  - Actions
  - Configurators
- MagicUWE

Caso de estudio: MagicDraw OpenAPI

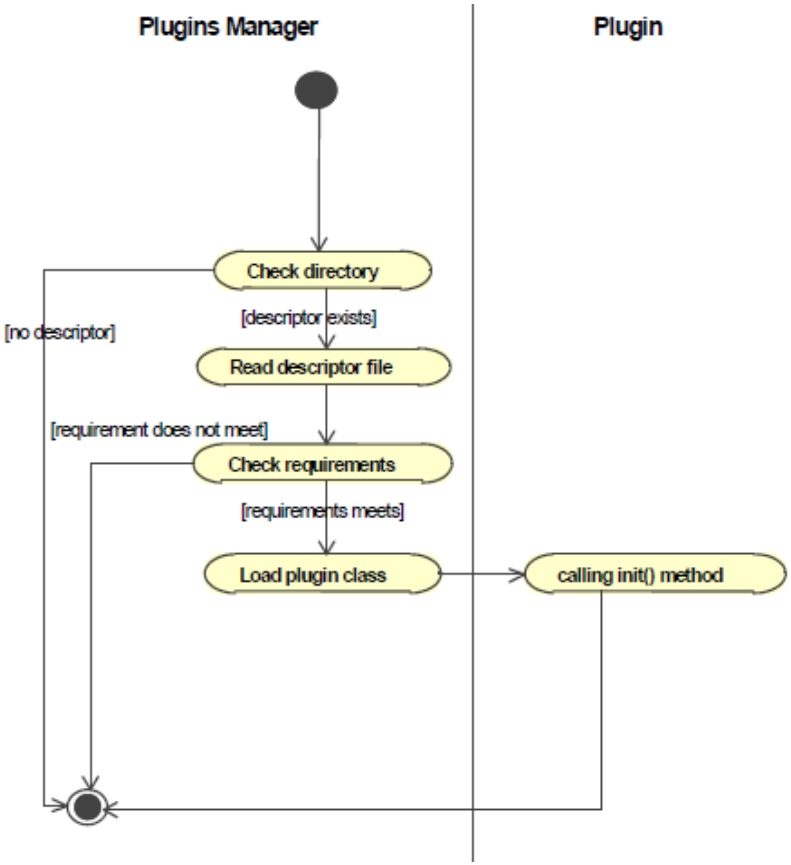
- ¿Qué es MagicDraw?
  - Aplicación de modelado UML
- ¿Por qué MagicDraw?
  - Soporte a UWE
  - MagicUWE es plugin de MagicDraw
  - Paquetes (separación de concerns)
  - Portabilidad (JVM)
  - Generación de código automáticamente
  - Deriva modelos a partir de código
  - Fácil de usar
- Extensibilidad ➔ OpenAPI

Extensibilidad: Plugins

- Nueva funcionalidad
- Recursos
  - Directorio
  - JAR (archivos Java compilados)
  - XML Descriptor

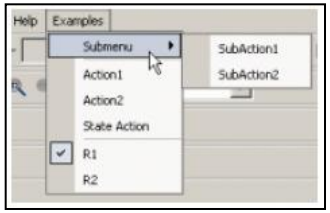
```
<plugin id="MagicUWE4R" name="Miguel Djebaile Thesis" version="1.0"
  provider-name="Facultad de Informática - Universidad Nacional de La Plata"
  class="magicUWE.core.PluginManager">
   <requires>      <api version="1.0"/>                  </requires>
   <runtime>       <library name="MagicUWE4R.jar"/>      </runtime>
</plugin>
```

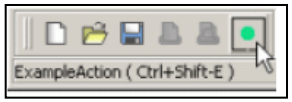
- Funcionamiento
  - 1º Plugin scan
  - 2º Busca descriptor
  - 3º Chequea el descriptor
  - 4º Carga main Class
    (extends com.nomagic.magicdraw.plugins.Plugin)
  - 5º Invoca método *init()*
  - *6º close()* al cerrar

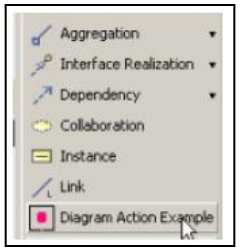


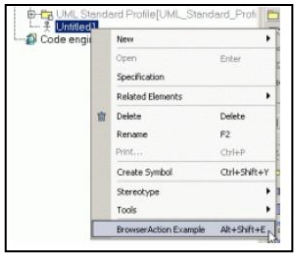
Extensibilidad: Plugins

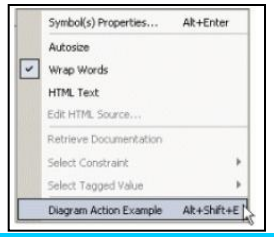
- Actions
  - Menú principal
  - Barra de menú
  - Barra de diagrama
  - Menú del browser
  - Menú contextual

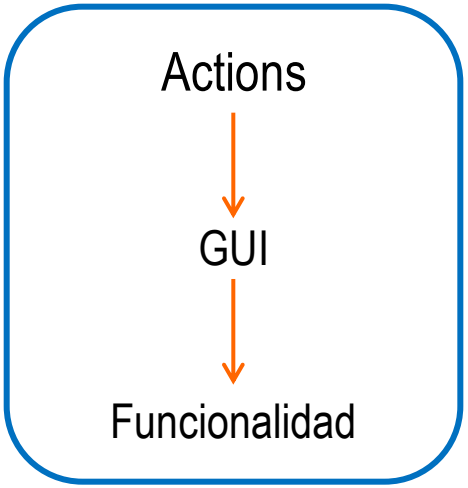








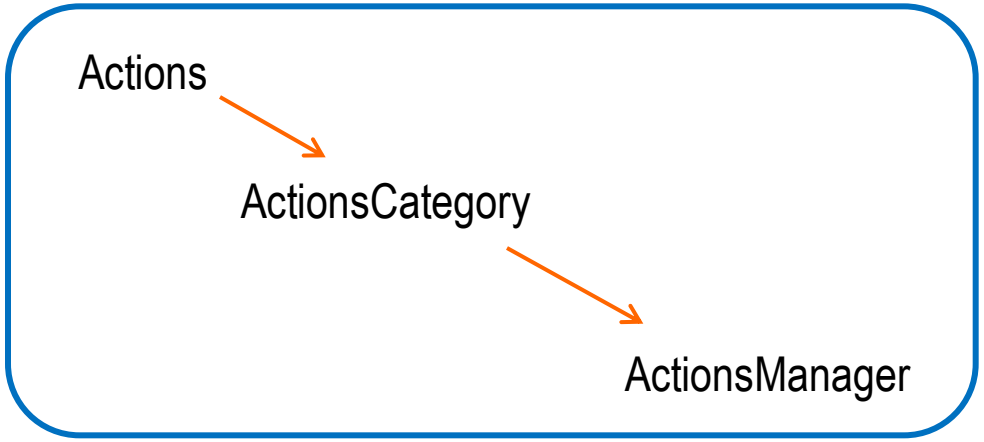




MagicUWE4R

Extensibilidad: Plugins

- Jerarquía de clases de Actions
  - DefaultBrowserAction
  - DefaultDiagramAction
  - PropertyAction
  - **NMStateAction**
  - **DrawPathDiagramAction**
  - DrawShapeDiagramAction
- ActionsCategory
- ActionsManager



	ActionsManager	ActionCategory	Action
Menú	Barra de menú	Menú	Ítem del menú
Barra de herramientas	Todas las barras	Una barra	Botón
Menú contextual	Menú contextual	Submenú	Ítem del menú

- Configurators
  - AMConfigurator (menúes o barras de herramientas)
  - BrowserContextAMConfigurator (browser)
  - DiagramContextAMConfigurator (diagrama)

MagicUWE

- UWE

- Características
  - Simple
  - Intuitivo
  - Separación de concerns
- Código fuente modular, bien diseñado y documentado

MagicUWE + Refactoring =
MagicUWE4R

Arquitectura de MagicUWE4R

- Arquitectura basada en:

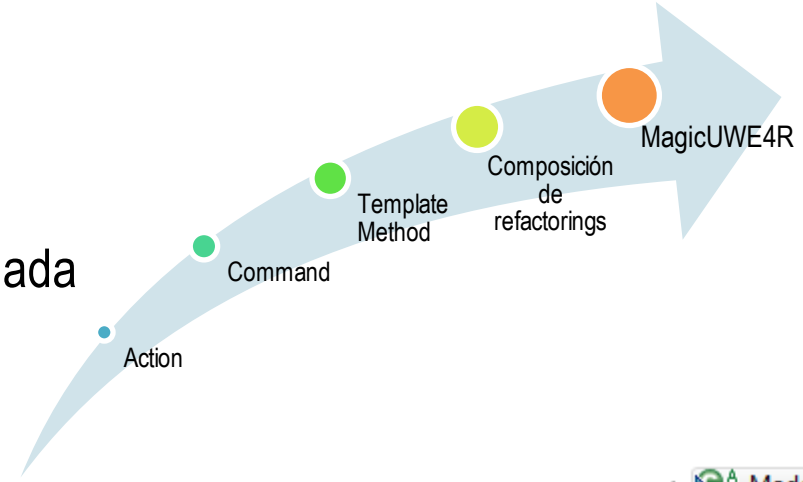
  - Composición de refactorings en el código

  - Diseño extensible basado en patrones
    - Template method como generador de un framework
    - Command + Observer para la ejecución de actions
    - Visitor para el acceso de los elementos

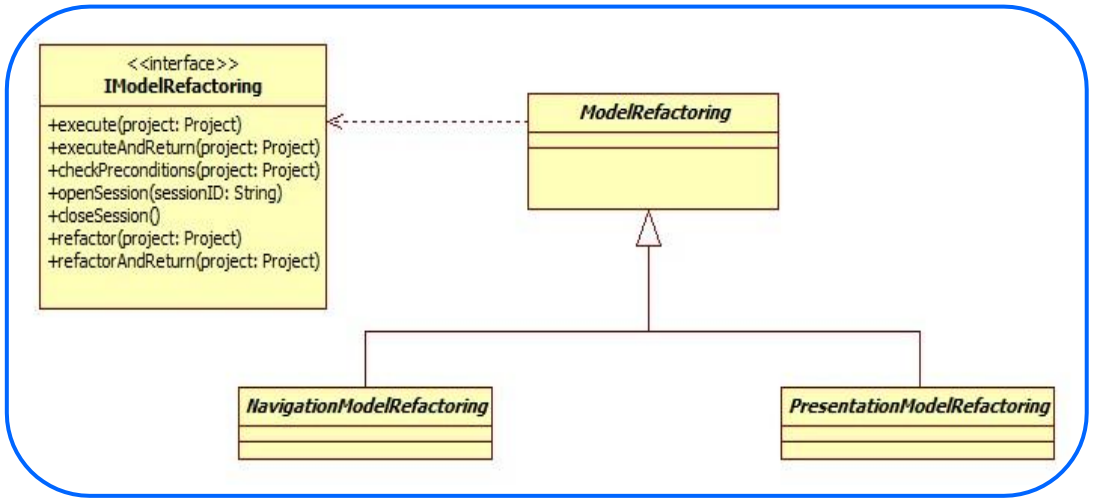
- Extensión de MagicUWE4R

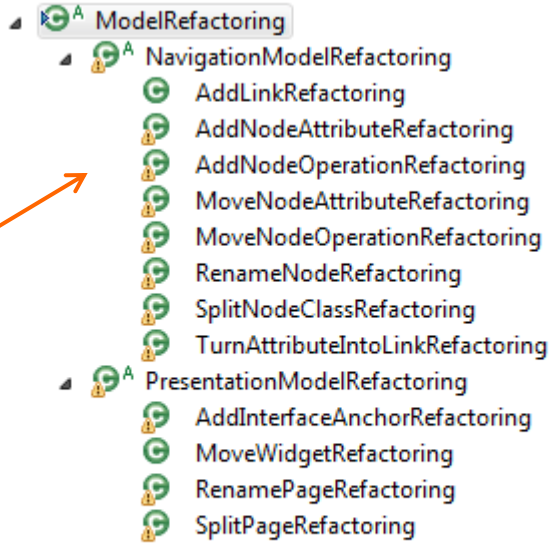
Arquitectura

- Desarrollo iterativo
- Arquitectura rediseñada N veces



- Diseño



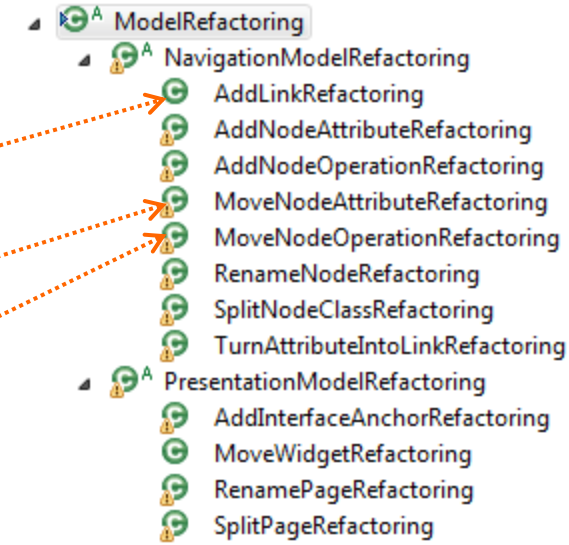


Composición de refactorings en el código

- Refactorings como unidades atómicas
  - Robusto
  - Seguro
  - Pueden componerse
- Motor de refactorings
  - Extensible
  - Flexible
  - Escalable

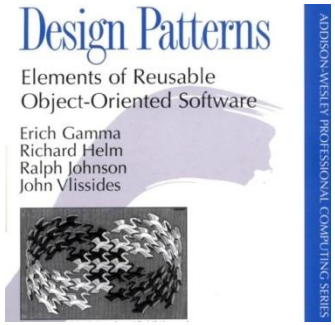
Nuevo refactoring complejo =
(Split Node Class)

R1
+
R2
+
R3



Diseño extensible basado en patrones

- ¿Qué es un patrón de diseño?
  - Tomado de la Arquitectura (profesor de Berkeley Christopher Alexander, 1979)
  - "… describe un problema que ocurre una y otra vez en nuestro entorno, explicando el núcleo de la solución a ese problema, de tal manera que esa solución pueda ser usada más de un millón de veces sin hacerlo siquiera dos veces de la misma forma."
  - Gang of Four (1995) presenta un catálogo de 23 patrones de diseño.

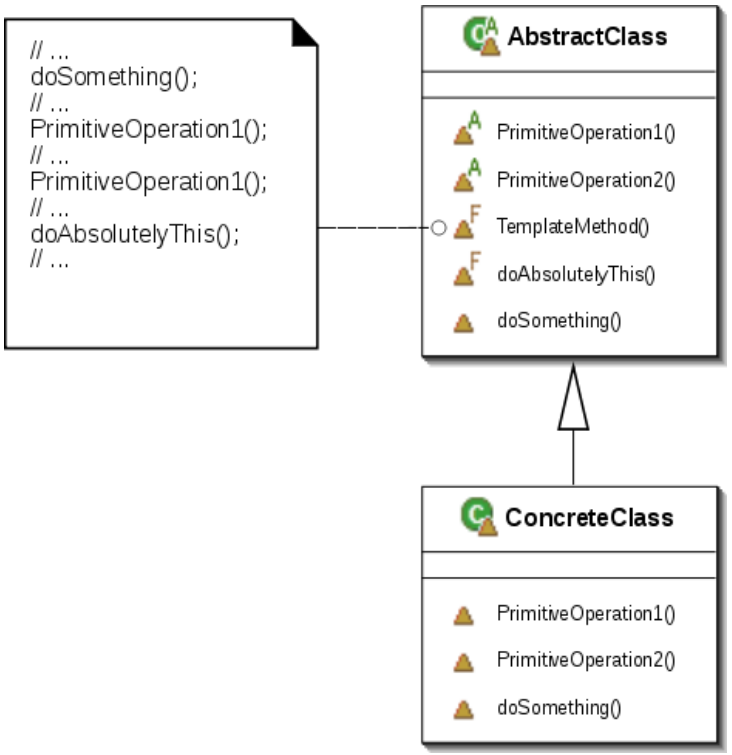


- Ventajas

  - Proporcionar catálogos de modelos reusables
  - Evitar la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente (teniendo la mejor disponible)
  - Formalizar un vocabulario común entre diseñadores.
  - Estandarizar el modo en que se realiza el diseño.
  - Facilitar el aprendizaje de las nuevas generaciones de diseñadores (buenas prácticas)

Template Method

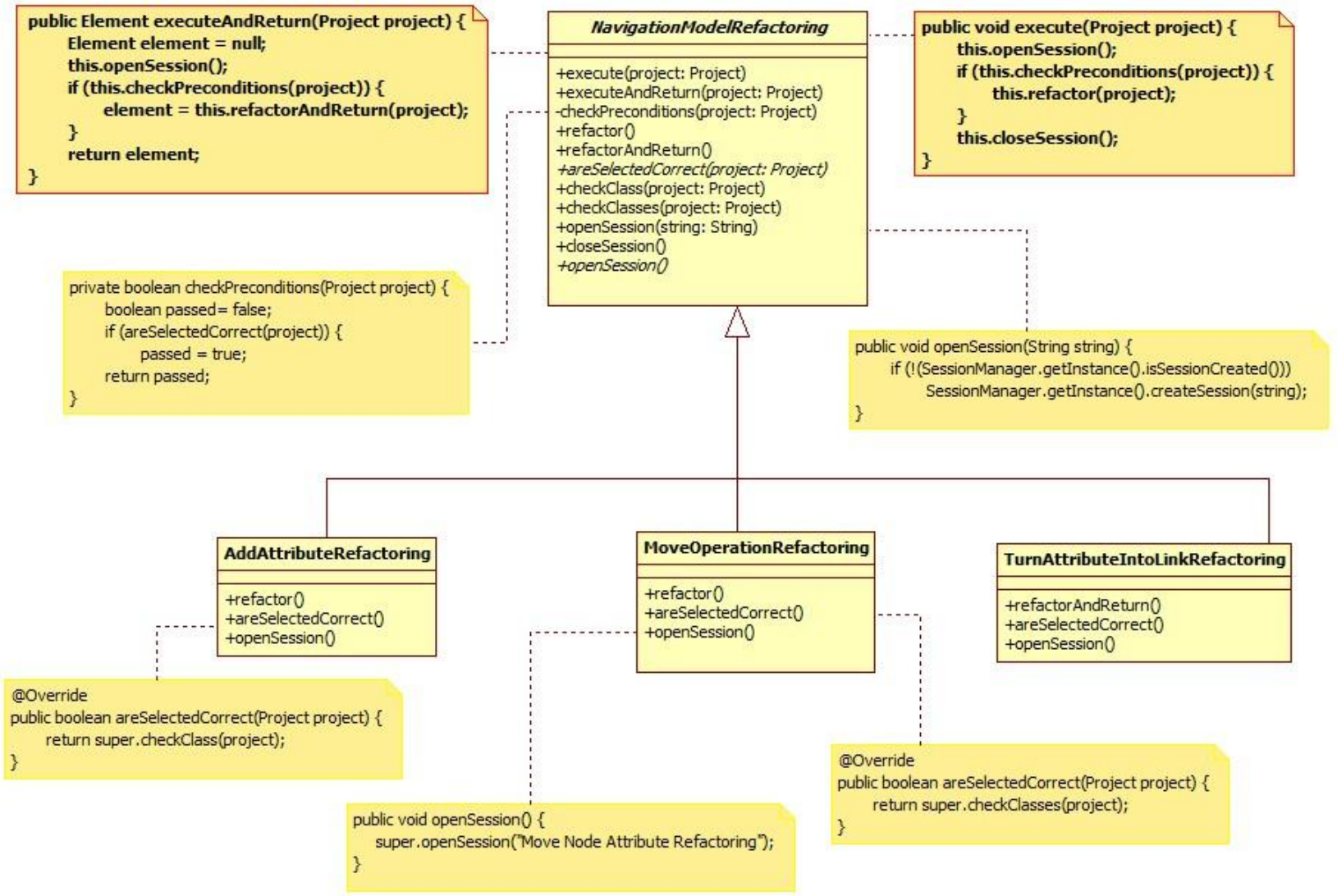
- Patrón de comportamiento
- Aplicabilidad
  - Define un esqueleto de un algoritmo con partes invariantes + comportamiento variable
  - Comportamiento común evitando duplicación de código



- Hook methods

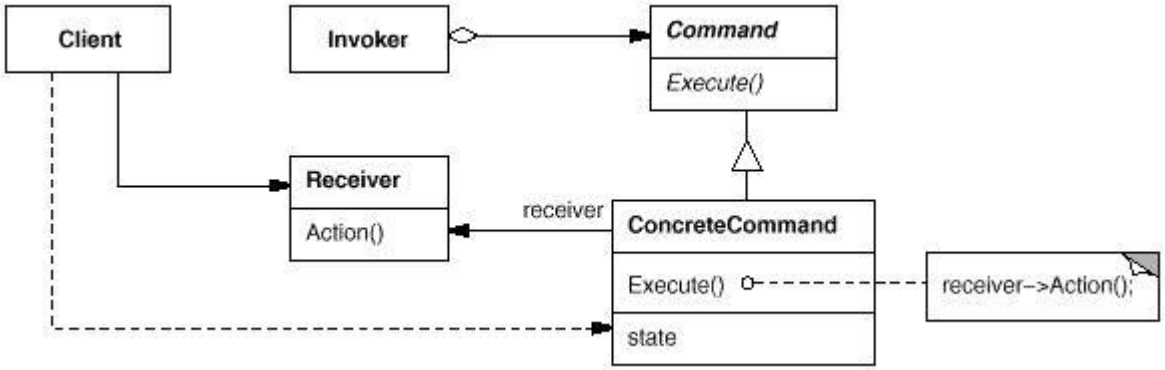
```
class Superclass {
        protected void preSomethingHook(){}
        protected void postSomethingHook(){}
        void something() {
                preSomethingHook();
                // alguna implementación
                postSomethingHook();
        }
}
class Subclass extends Superclass {
        protected void preSomethingHook() {
                // código customizado
        }
        protected void postSomethingHook() {
        // código customizado
        }
}
```

Template Method en MagicUWE4R



Ejecutando actions: patrón Command

- Patrón de comportamiento

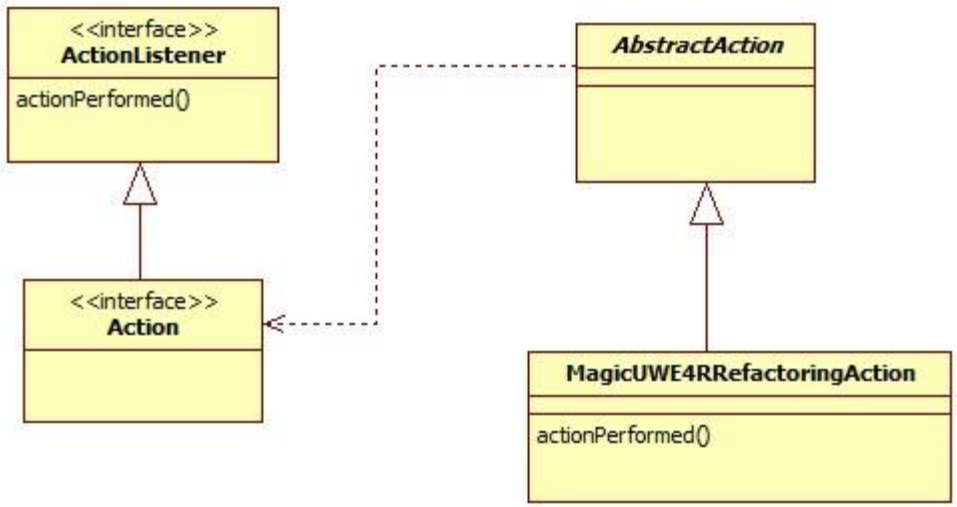


- Encapsular el mensaje como un objeto
  - Interfaz común para invocar acciones de forma uniforme
  - Envío de solicitudes a objetos que desconocemos
  - Execute
  - Log
  - Undo
  - Ej: Copy del procesador de textos

Command + Observer en MagicUWE4R

- Actions en todas sus representaciones ⇢ objeto Command
- Listeners en Java Swing ⇢ Patrón Observer
  - MouseListener
  - WindowListener
  - ***ActionListener*** *(que nuestros actions implementan)*

✓ Ioc (Inversion of Control)
vs.
✗ Polling



Extensión de MagicUWE4R

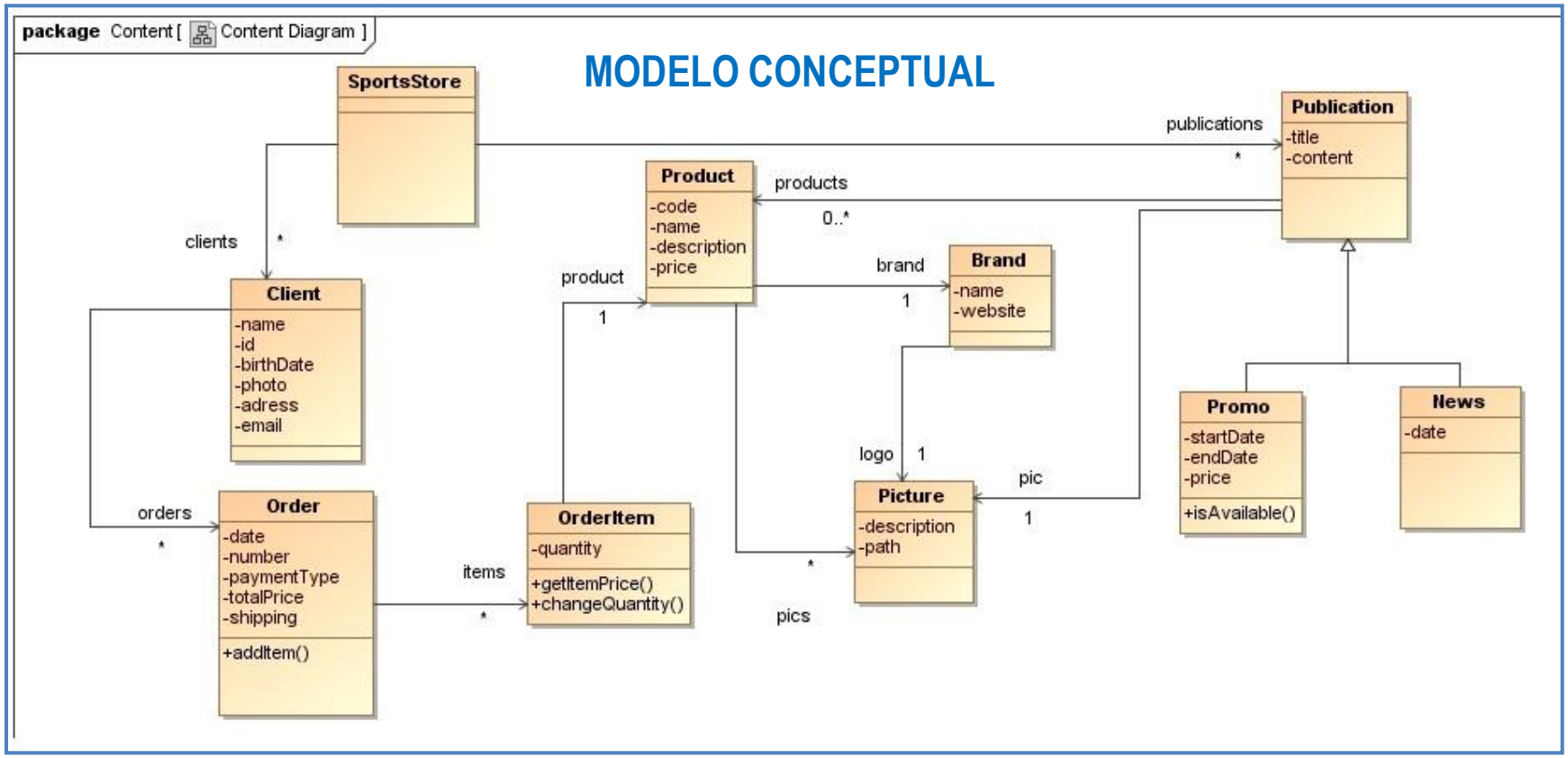
- Hot – spots en caso que queramos extender
  - PluginManager.java (setea Configurators)
  - PluginManagerActions.java (asigna los Actions a cada Configurator)
  - Paquete magicUWE.actions.refactoring (donde se encuentran los Actions)
  - Paquete magicUWE.configurators.context.refactoring (donde se encuentran los Configurators)
  - Paquete magicUWE.core.model (posee las implementaciones de los refactorings)
- Nuevo refactoring
  - 1º Creación del Action (recupero los elementos del diagrama y delego al modelo)
  - 2º Desarrollo de la implementación del refactoring en el modelo
  - 3º Actualización del Configurator a cargarse en la inicialización del plugin, para que incluya el nuevo action.

Refactorings en el Modelo de Navegación

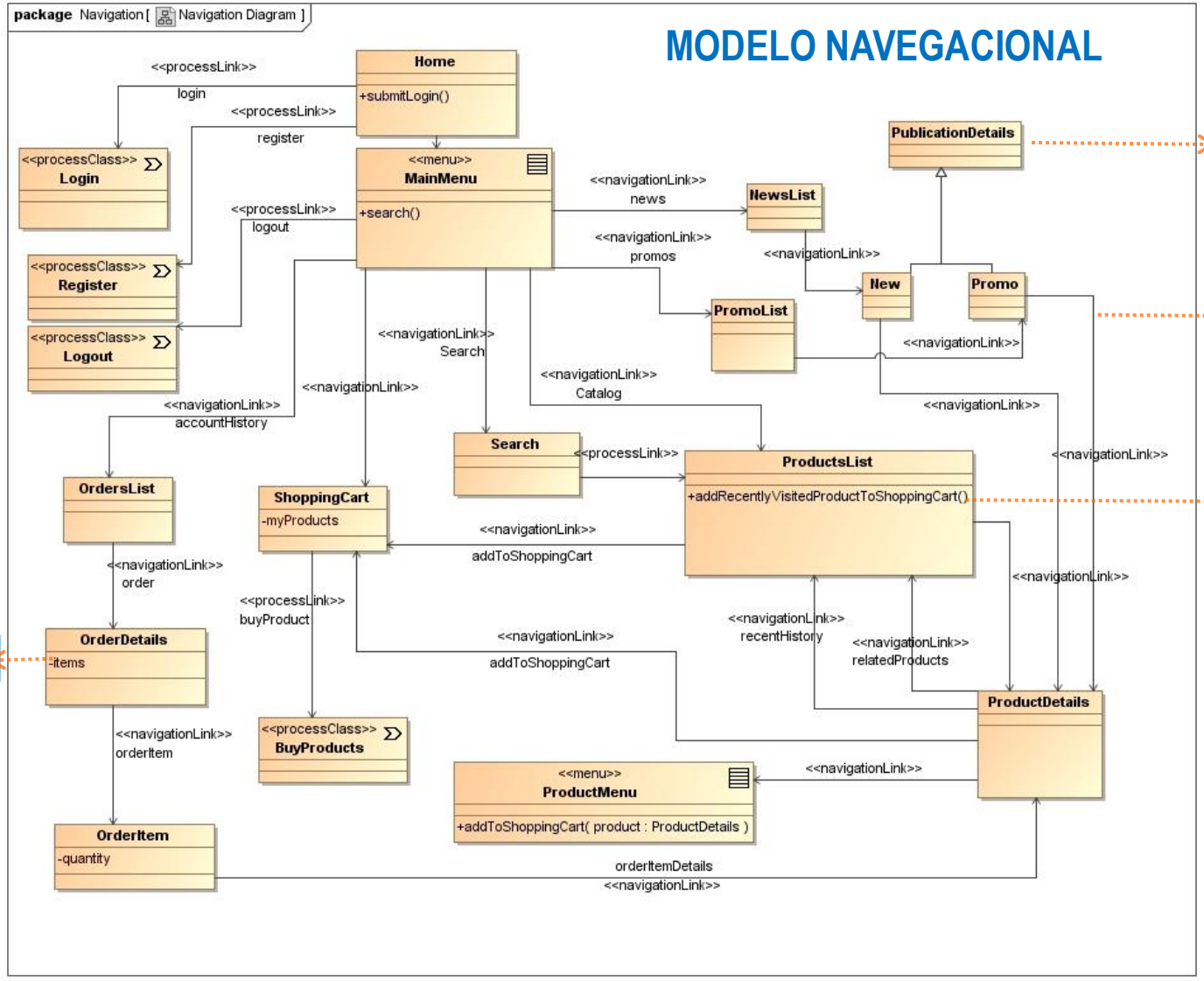
- Caso de estudio
- Add Node Operation
- Move Node Attribute
- Move Node Operation
- Rename Node
- Add Link
- Split Node Class
- Turn Attribute Into Link

Caso de Estudio

- *Sports Store,* aplicación de ventas online de indumentaria deportiva



Caso de Estudio



Add Node Operation

Motivación

- Nuevas operaciones en etapas posteriores al diseño
  - Nuevos requerimientos
  - Acelerar procesos
- Mejora de diseño (operaciones acopladas para que el usuario no tenga que acceder a un nodo extra)

Mecanismo

- Refactoring atómico
- Precondiciones:
  a) El nodo origen debe existir en el diagrama de navegación
  b) La nueva operación no existe en el nodo origen.
- Ejecución:
  1. Seleccionar el nodo al cual queremos agregar la nueva operación
  2. Darle un nombre a la operación.

Ejemplo en MagicUWE4R

(poder comprar un producto recientemente visitado)

Move Node Attribute

Motivación

- Atributo de un nodo origen debe ser movido a un nodo destino
- Empleado en Split Node Class

Mecanismo

- Refactoring atómico
- Precondiciones:
  a) El nodo origen y el nodo destino deben existir en el diagrama navegacional
  b) El atributo a mover no existe en el nodo de destino.
- Ejecución:
  1. Seleccionar nodo origen y nodo destino
  2. Escoger el atributo a mover
  3. Copiar el atributo elegido en el nodo destino.
  4. Eliminar el atributo del nodo origen.

Ejemplo en MagicUWE4R

(poder mostrar la cantidad de productos según el tipo)

Split Node Class

- Motivación
  - Desacoplar nodo repleto de información
  - Extraer atributos/operaciones a un nuevo nodo linkeado

- Mecanismo
  - Refactoring compuesto
  - Precondiciones:

    a) El nombre del nuevo nodo debe ser distinto a cualquiera de los ya existentes en el modelo de navegación.
  - Ejecución:

    1. Nuevo nodo vacío.
    2. Para cada atributo que se decida, usar el refactoring *Move Node Attribute*.
    3. Para cada operación que se decida, usar el refactoring *Move Node Operation*.
    4. Usar el refactoring *Add Link* para agregar un link entre la clase del nodo original y el nuevo nodo
    5. Opcionalmente renombrar el nodo original (*Rename Node*)

- Ejemplo en MagicUWE4R 

(desacoplar lista de productos con sus detalles)

Turn Attribute Into Link

Motivación

- Necesidad de transformar un atributo que se muestra en la pantalla, en un camino de navegación hacía más detalles sobre lo que ese atributo representa.

Mecanismo

- Refactoring compuesto
- Precondiciones:
  a) El link con nombre del atributo seleccionado no existe previamente entre nodo origen y nodo destino.
- Ejecución:
  1. Seleccionar el atributo en el nodo de origen que deseamos convertir en el enlace.
  2. Usar el refactoring Add Link, con nombre del atributo seleccionado previamente, desde el nodo de origen al destino
  3. En el nodo origen, reemplazar la definición del atributo por la definición de un link o ancla.

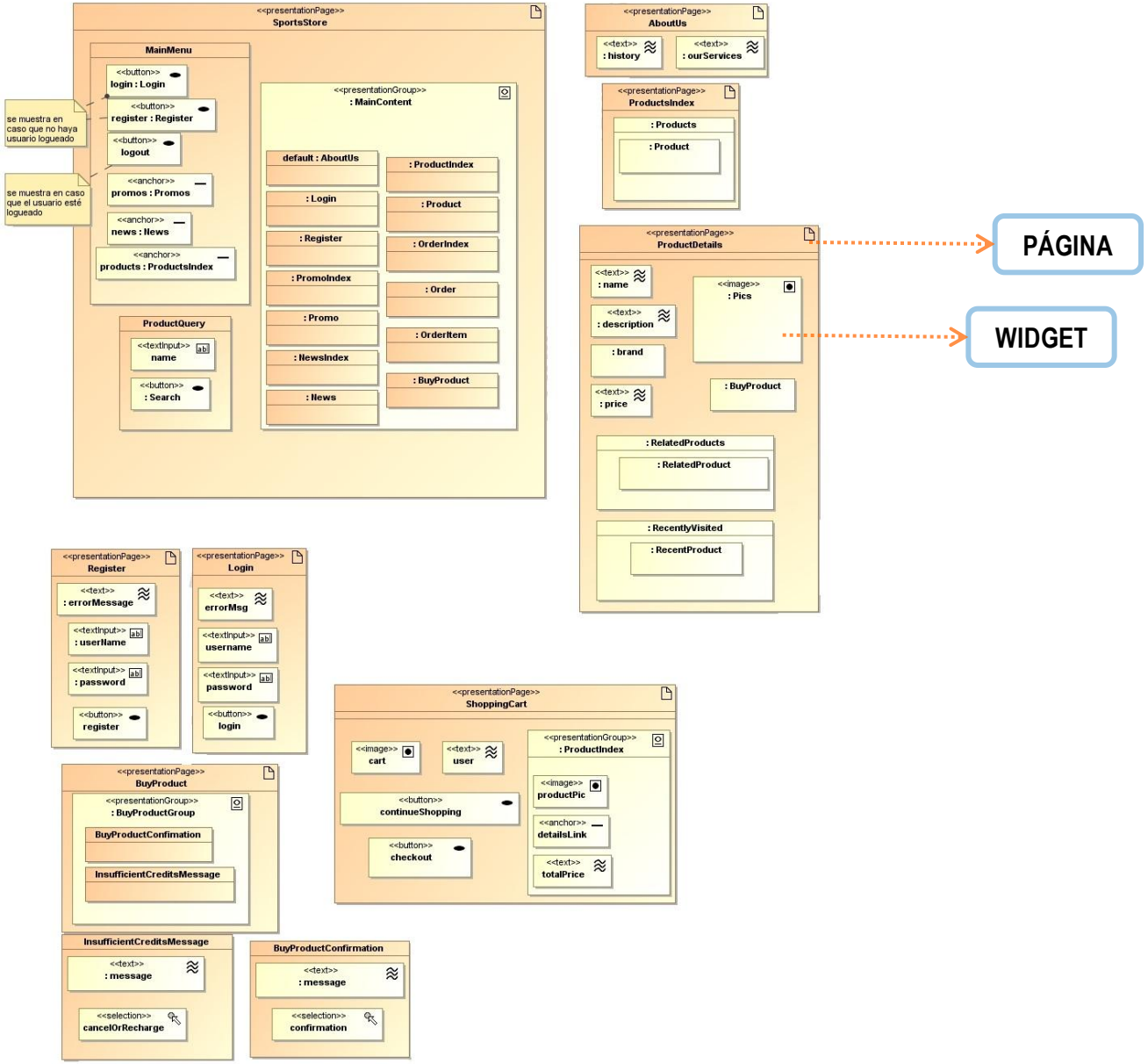
Ejemplo en MagicUWE4R

(acceder a los detalles de los productos del carrito)

Refactorings en el Modelo de Presentación

- Move Widget
- Rename Page
- Add Interface Anchor
- Split Page

Caso de Estudio



Move Widget

- Motivación
  - En la mayoría de los casos, desencadenado por un refactoring en el modelo navegacional
  - Empleado en *Split Page*
- Mecanismo
  - Refactoring atómico
  - Precondiciones:
    a) El nombre del widget no existe en la página destino.
  - Ejecución:
    1. Se seleccionan página origen y página destino.
    2. Se elige el widget a trasladar.
    3. Se copia la definición del widget a la página destino.
    4. Se elimina el widget de la página origen.

- Ejemplo en MagicUWE4R *(integrar los campos de Login al Home)*

Add Interface Anchor

- Motivación
  - Consecuencia de *Add Link* o *Turn Attribute Into Link* en el modelo de navegación
  - Empleado en *Split Page*
- Mecanismo
  - Refactoring atómico
  - Precondiciones:
    a) El widget de tipo anchor (o ancla) con el nombre dado no existe en la página destino
  - Ejecución:
    1. Identificar la página a la cual se debe agregar el widget de tipo anchor.
    2. Agregar el widget con el estereotipo anchor.

- Ejemplo en MagicUWE4R *(accionado por Turn Attribute Into Link)*

Split Page

Motivación

- Como consecuencia de *Split Node Class*
- Página demasiado colmada de información (perder enfoque o scrolling)

Mecanismo

- Refactoring compuesto
- Precondiciones:

  a) El nombre de la nueva página debe ser distinto a cualquiera de las ya existentes en el modelo de presentación.
- Ejecución:

  1. Usar el refactoring Move Widget para mover los widgets seleccionados desde la página original a la nueva.
  2. Usar el refactoring Add Interface Anchor en la página origen para enlazar las dos páginas y permitir al usuario acceder al contenido y las operaciones disponibles en la página original.
  3. Verificar si es necesario cambiar el nombre de la página original y si es así cambiarlo mediante Rename Page.

Conclusiones

Conclusiones

Única herramienta disponible para refactoring de aplicaciones Web

Fuerte impacto de MDD	+	Siempre presente Refactoring	=	MagicUWE4R
Modelo	*vs.*	*Código*		
Abstracción	*vs.*	*Precisión*		

- Plugin para MagicDraw
- Basado en un catálogo de refactorings genéricos (subconjunto de ellos en MagicUWE4R 1.0)
- Fácilmente extensible gracias al uso de patrones de diseño y la composición de refactorings
- Project Managment, Tracking, Source Code **https://magicuwe4r.unfuddle.com**

Publicaciones

- 40º Jornadas Argentinas de Informática JAIIO
- Paper presentado, aprobado y publicado en el simposio EST

Limitaciones

- MagicDraw API tiene poca documentación ✓ Código fuente (debugging) + Foro de devs
- Nuevas versiones de MagicUWE durante el proceso (adaptación de código)
- MagicDraw, aplicación propietaria (Community Edition sin soporte)

Trabajos Futuros

- Sincronización de una capa a otra
  - Reflejar automáticamente un refactoring del modelo navegacional, en el modelo de presentación.

- Construcción instantánea base de una aplicación Web a partir de la generación automática de código.
  - Construir una implementación base *(POJO)* de manera automática a partir de los modelos diseñados
  - Refactorings también se vean a nivel de código de manera automática.
  - *UWE4JSF*

- Implementación de otros refactorings del catálogo


THANK
YOU
GRACIAS
ARIGATO
SHUKURIA
TASHAKKUR ATU
SUKSAMA
EKHMET
MEHRBANI
GRAZIE
BOLZÏN
MERCI
BÏYAN
SHUKRIA
DANKSCHEEN
JUSPAXAR
GOZAIMASHITA
EFCHARISTO
KOMAPSUMIDA
MAAKE
PALDIES
YAQHANYELAY
TINGKI
SPASSIBO
SNACHALHUYA
NURUN
CHALTU
WABEEJA
MAITEKA
DHANYABAAD
ATTO
AMHA
MERSI
SPASIBO
DENKAUJA
NENACHALHYA
UNALCHEESH
HUI
GUI
HATUR
EIKOJU
SIKOMO
MAKETAI
MINMONCHAR
SANCO
MERASTAWHY
GAEJYHO
LAH
TAVTAPUCH
MEDAWAGSE
AGUYJE
FAKAAUE
BAJKA

¿ Preguntas ?