Departamento de Sistemas Informáticos y Computación

Universitat Politècnica de València

Comparativa entre herramientas MDD enfocada en la versatilidad del lenguaje con respecto a la implementación de requerimientos. Caso práctico Integranova - WebRatio

Trabajo Fin de Máster

Máster Universitario en Ingeniería y Tecnología de Sistemas Software

Autor: Daniel Fernando Pineda Alvarez

Tutores: Óscar Pastor López – Ignacio Panach Navarrete

2016-2017

Agradecimientos

Agradezco en primer lugar a Dios, como ente creador, por ser mi fuerza e inspiración en cada paso que doy en mi vida y ayudarme en cada momento a superar los obstáculos que me presenta la misma.

Al Gobierno de mi país Ecuador, el cual, por medio del Instituto de Becas SENESCYT, me brindó la oportunidad a mí y a muchos otros jóvenes ecuatorianos de alcanzar nuestras metas académicas, mi saludo y entera gratitud a esta confianza en quienes estamos en un período de formación profesional y que buscamos día a día el crecimiento de nuestra amada nación, muchas gracias.

De igual manera agradezco esto a mis tutores de tesis, Óscar e Ignacio, que sin su invaluable guía y soporte habría sido imposible realizar este trabajo, gracias por permitirme tener el honor de trabajar junto a Ustedes.

A mi esposa Andrea, gracias por ser la amiga, compañera y amante con quien tengo el gusto de compartir mi vida, mis derrotas y mis victorias, mis aciertos y mis errores, gracias por estar junto a mí en cada momento y enseñarme nuevamente a volar, ha sido, es y será la razón de mi vida.

A mi madre Judith por permitirme llegar a donde estoy ahora, no sería nada sin su ayuda madrecita adorada, gracias por creer en mi aun cuando nadie lo hacía y por no permitir jamás que me hunda en mis errores, gracias por perdonar, por aceptar, por corregir, Usted y solo Usted tiene la gracia de saber enseñar con amor y lealtad. Gracias madre por todo y por tanto, esto es suyo como cada uno de mis logros que sin su ayuda y apoyo jamás pudiera alcanzarlos. A mi padre Manuel, pese a no tenerlo a mi lado jamás olvidaré sus enseñanzas y fortaleza, un agradecimiento hasta el cielo papá, gracias por no dejarme nunca.

A mis tías Blanca y Loli, por ser mis segundas mamás en la tierra, gracias por el gran amor que me tienen y por estar ahí siempre, las amo muchísimo. A mi tío Héctor, un agradecimiento lleno de gratitud por ser tan gran amigo y mentor.

A mis hermanos David y Charo, sin su ejemplo diario muchos de mis objetivos perderían sentido, gracias por enseñarme con actos cuanto se debe hacer para ser una excelente persona, siempre luchando por conseguir llegar a ser todo lo que me han enseñado, los amo muchísimo.

A mis suegros Claudia y Luis gracias por ser quienes son, por enseñarme el valor de la gratitud y el trabajo honrado y diligente, que a pesar de que todo parezca sombrío, la estoicidad y la cortesía son la mejor arma del valiente; siempre dispuesto a la lucha y no claudicar ante ningún tribulo, esta nueva meta es dedicada a todo ese gran esfuerzo y ejemplo que he recibido. Gracias por todo su apoyo en todo sentido, son las mejores personas que he tenido el gusto de conocer.

A mis cuñados Gabriel, Jenny, Jorge, Gabriela, Andrés, Melina y Estefanía, ciertamente los hermanos que la vida me ha regalado, gracias por permitirme ser parte de sus vidas y confiar en mí, cada palabra de aliento, cada detalle entregado, ha sido grabado en mi corazón, nunca cambien y mantengan siempre ese enorme corazón.

A mis amigos, Pablo, Rafael, Mauricio, Marco, Geovanny, Alina, Daniel, Valeria y Karina, gracias por compartirme un espacio tan importante en sus vidas y no dejarme caer en la tristeza,

por mostrarme que no todo está oscuro y siempre hay una luz al final del camino, los quiero muchísimo. De manera especial y pese a no estar físicamente a tu lado te dedico esto amigo y hermano mío, siempre te he sentido a mi lado en cada momento, gracias por no abandonarme y permitirme estar a tu lado de una u otra manera, un abrazo hasta el cielo Sebastián.

Finalmente, y no por ello menos importante, un agradecimiento a mis sobrinos Gabriel Andrés, María José, Isabella Estefanía, José Felipe, David Francisco y Joaquín Andrés, ellos, como la prolongación de la existencia y en la candidez de su inocencia, han sabido llenar de dicha cada uno de mis días, encerrando en su alegría, la esencia misma de la vida.

Resumen

Dentro del análisis y desarrollo de software, se ha considerado un aspecto fundamental la implementación de estándares que nos permitan implementar, diseñar e integrar diferentes elementos para que una aplicación cumpla o se encuentre enmarcada dentro de diferentes características. Los requerimientos de los usuarios se han convertido con el pasar del tiempo en verdaderos retos a cumplir por parte de los desarrolladores de software y se ha buscado la necesidad de abaratar costos con el objetivo de masificar la producción de contenidos.

Es por esta razón, que se ha incursionado en nuevas tecnologías que manejan este tipo de paradigmas con el objetivo de optimizar los recursos necesarios para implementación de proyectos tecnológicos, se han encontrado herramientas como Integranova, WebRatio, Genexus, que han adoptado el paradigma o parte de él, ofreciendo una alternativa de desarrollo ágil en un mercado cada vez más competitivo.

El objetivo de la presente investigación es analizar y realizar una comparativa entre dos herramientas basadas en un paradigma Model-Driven y determinar cuál de las dos es más conveniente para su uso por las características que posea, así como también por que tan bien cumpla los parámetros para ser considerada una herramienta basada en modelos.

En el presente estudio se realizará la comparativa entre dos herramientas en un escenario determinado, mediante el mismo se plantea analizar las ventajas y desventajas que posee cada herramienta en el desarrollo de los diferentes requisitos del problema inicial. Se evaluará la versatilidad de la herramienta y las facilidades que presentan a los usuarios para su correcta implementación y de igual manera se evaluará la capacidad de asimilación de la herramienta como la base de un factor de productividad aplicada a un campo real.

Habrá diferentes normativas que satisfacer partiendo de la definición del modelo Entidad–Relación, servicios, transacciones, roles y permisos de usuario, etc. Se evaluarán como se manejan estos requisitos en ambas herramientas, así como el producto final obtenido con cada una de ellas.

Palabras clave: Modelado Conceptual, Desarrollo Dirigido por Modelos (MDD), Ingeniería de Sistemas de Información

Abstract

Within the analysis and development of software, it has been considered a fundamental aspect the implementation of standards that allow us to implement, design and integrate different elements for an application to meet or be framed within different characteristics. The requirements of users have become over time in real challenges to be met by software developers and has sought the need to lower costs with the goal of mass production of content.

For this reason, that has been penetrated in new technologies that handle this type of paradigms with the objective of optimizing the necessary resources for the implementation of technological projects, we have found tools such as Integranova, WebRatio, Genexus, that have adopted the paradigm or part of it, offering an agile development alternative in an increasingly competitive market.

The objective of the present research is to analyze and to make a comparative between two tools based on a Model-Driven paradigm and to determine which of the two is more convenient for the use by the characteristics that it possesses, as well as by that well it complies the parameters to be considered a Model-Based tool.

In the present study the comparison between two tools will be carried out in a given scenario, through the same one it is proposed to analyze the advantages and disadvantages that each tool has in the development of the different requirements of the initial problem. It will evaluate the versatility of the tool and the facilities that they present to the users for their correct implementation and likewise will evaluate the capacity of assimilation of the tool as the basis of a factor of productivity applied to a real field.

There will be different regulations to satisfy based on the definition of the model Entity-Relationship, services, transactions, roles and user permissions, etc. We will evaluate how these requirements are handled in both tools, as well as the final product obtained with each of them.

Keywords: Conceptual Model, Model-Driven Development, Information Systems Engineering

Tabla de Contenidos

Índice de Figuras

1. Introducción

Dentro del análisis y desarrollo de software, se ha considerado un aspecto fundamental la implementación de estándares que nos permitan implementar, diseñar e integrar diferentes elementos para que una aplicación cumpla o se encuentre enmarcada dentro de diferentes características. Los requerimientos de los usuarios se han convertido con el pasar del tiempo en verdaderos retos a cumplir por parte de los desarrolladores de software y se ha buscado la necesidad de abaratar costos con el objetivo de masificar la producción de contenidos.

Parte de estos esfuerzos se encuentran centrados en la reutilización del software y los modelos que nos permitan potenciar el reúso de diferentes elementos de software y facilitar los diferentes roles que participan en el proceso [1].

La implementación de modelos ha tenido inconvenientes, al ser considerada en sus inicios como una plataforma que implicaba documentación netamente, el querer realizar la transformación a código tenía la problemática de su implementación, al ser tan genérica, imposibilitaba el poder diseñar requerimientos específicos y dificultaba tenerlo como herramienta de desarrollo industrial quedándose en un aspecto teórico únicamente.

Estos problemas se consideraron debido a los cambios y poca claridad al momento de capturar requerimientos de parte de los ingenieros del software, por lo que las debilidades de arquitecturas y desarrollos MDD empezaron a tomar importancia para un manejo más controlado de estos cambios de requerimientos. Manejar este tipo de situaciones es un verdadero reto puesto que implicaba horas de programación que eran difíciles de manejar, una arquitectura basada en modelos permitía que el uso de componentes sean reutilizados de tal forma que soporte cambios en tiempo real y no altere con el tiempo programado de entrega del producto final [2].

En la actualidad se han analizado los beneficios que aportan la realización de desarrollo por medio de tecnologías enfocada a modelos, puesto que, en función del paradigma que se utilice y de la fase del proceso de diseño que se está llevando a cabo, el entorno ofrezca una vista especializada del sistema, la cual presenta de una forma precisa y coherente la información sobre la que se está decidiendo [3].

Es por esta razón que se han incursionado en nuevas tecnologías que manejan este tipo de paradigmas, con el objetivo de optimizar los recursos necesarios para la implementación de proyectos tecnológicos, se han encontrado herramientas como Integranova, WebRatio, Genexus, que han adoptado al paradigma o parte de él, ofreciendo una alternativa de desarrollo ágil en un mercado cada vez más competitivo.

Los estudios comparativos de estas herramientas se han visto centrados en una relación entre programación basada en modelos contra programación tradicional, siendo la más común la basada en objetos. Al ser un paradigma nuevo, los estudios buscan analizar su factibilidad con respecto a herramientas existentes y de probada capacidad, sin embargo, la evolución de esta nueva modalidad de desarrollo ha generado nuevas herramientas permitiendo competir entre sí y alcanzando una madurez lo suficientemente estable como para permitirse un estudio comparativo detallado entre las mismas.

En el desarrollo de la presente investigación se realizará un estudio comparativo entre dos herramientas específicas, WebRatio e Integranova, enfocándose en su capacidad de resolver un universo de requerimientos establecido en un ejercicio determinado. Para ello se buscará encontrar la forma de capturar los requerimientos y como los mismos son procesados por cada uno de las herramientas para obtener así el producto requerido.

1.1. Objetivo

El objetivo de la presente investigación es analizar y realizar una comparativa entre dos herramientas basadas en un paradigma basado en modelos y determinar cuál de las dos es más conveniente para su uso por las características que posea, así como también por que tan bien cumpla los parámetros para ser considerada una herramienta basada en modelos.

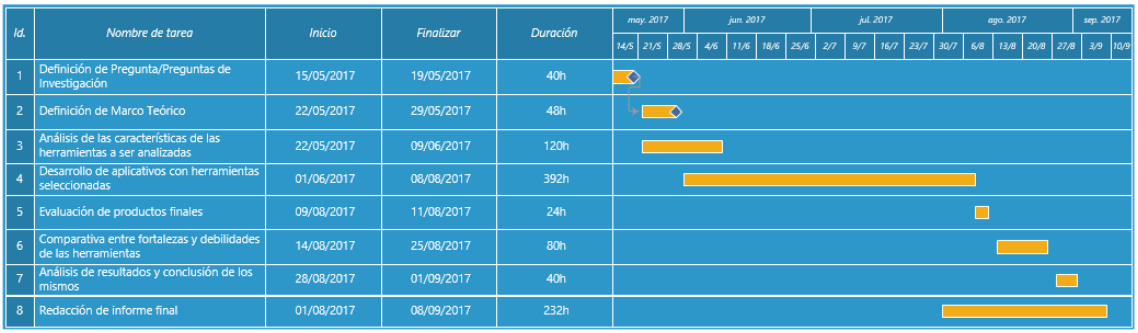
1.2. Planificación

Para la planificación del desarrollo del presente Trabajo de Fin de Máster se han considerado los siguientes elementos:

- Definición de Pregunta/Preguntas de Investigación
- Definición de Marco Teórico relacionando potencial de ambas herramientas frente a un ejemplo establecido.
- Análisis de las características de las herramientas y que tan bien se adoptan al paradigma orientado a modelos.
- Desarrollo de los aplicativos con las herramientas seleccionadas y que van a ser evaluadas.
- Evaluación de productos finales obtenidos con las dos herramientas.
- Comparativa de fortalezas y debilidades técnicas de las dos herramientas a ser evaluadas.
- Análisis de resultados y conclusión de los mismos.
- Redacción de informe final.

Para la realización del correspondiente diagrama de Gantt e ha considerado un promedio de trabajo de 8 horas diarias para el desarrollo del presente trabajo de fin de máster y no se han considerado los fines de semana, esto se ha configurado de esta manera por motivos de organización dado que se trabajaron algunos fines de semana para compensar los días en los que no se pudieron avanzar las 8 horas diarias por concepto del horario de clases del máster.

A continuación, se presenta el diagrama de Gantt que ilustra la planificación realizada para el desarrollo de la presente investigación:



Id.	Nombre de tarea	Inicio	Finalizar	Duración
1	Definición de Pregunta/Preguntas de investigación	15/05/2017	19/05/2017	40h
2	Definición de Marco Teórico	22/05/2017	29/05/2017	48h
3	Análisis de las características de las herramientas a ser analizadas	22/05/2017	09/06/2017	120h
4	Desarrollo de aplicativos con herramientas seleccionadas	01/06/2017	08/08/2017	392h
5	Evaluación de productos finales	09/08/2017	11/08/2017	24h
6	Comparativa entre fortalezas y debilidades de las herramientas	14/08/2017	25/08/2017	80h
7	Análisis de resultados y conclusión de los mismos	28/08/2017	01/09/2017	40h
8	Redacción de informe final	01/08/2017	08/09/2017	232h

Figura 1. Diagrama de Gantt que indica la planificación de la investigación. Gráfico del autor.

1.3. Pregunta de Investigación

PI-1. ¿Se puede considerar a la herramienta WebRatio como una integradora del paradigma orientado a modelos de manera que masifique los resultados y lo haga respetando las estructuras basadas en arquitecturas model driven?

1.3.1. Subpreguntas de investigación

- **SPI-1.** ¿Es posible considerar a WebRatio como una herramienta más potente y actual que Interanova para el desarrollo e implementación de proyectos basados en arquitecturas o paradigmas model driven?
- **SPI-2.** ¿Webratio es una herramienta que mantiene y respeta los lineamientos establecidos por la OMG para una implementación basada completamente en modelos?
- **SPI-3.** ¿WebRatio presenta mejoras o inconsistencias con respecto a otras herramientas basadas en modelos para la gestión de implementaciones de software?

1.4. Estructura del documento

Este documento se encuentra estructurado en seis capítulos resumidos a continuación:

1. **Capítulo Primero.** Aquí se encontrará desarrollada la introducción del presenta trabajo, el trazado de objetivos, preguntas de investigación y planificación del desarrollo de la tesis.
2. **Capítulo Segundo.** Desarrollo del estado del arte con respecto a herramientas Model-Driven, sus principales características y comparaciones en el mercado actual.
3. **Capítulo Tercero.** Desarrollo de marco teórico introductorio acerca de la metodología OO-Method (en el cuál se evaluará Integranova como caso práctico) y WebRatio.
4. **Capítulo Cuarto.** Comparativa de las herramientas seleccionadas acorde a un ejemplo práctico establecido en los Anexos.
5. **Capítulo Cinco.** Establecimiento de resultados obtenidos de la comparativa de las herramientas, así como las respuestas a las preguntas de investigación planteadas.

6. **Capítulo Sexto.** Conclusiones obtenidas de la realización de la presente investigación.

Posterior a la presentación de los capítulos encontraremos la sección de referencias normalizadas por la IEEE, así como los Anexos correspondientes.

2. Estado del Arte

2.1. Introducción

En los últimos años, la arquitectura de desarrollo de software basada en modelos ha adquirido un espacio fundamental en la creciente rama de la ingeniería del software, sin embargo, al ser éste un nuevo paradigma de desarrollo ha inquietado en cuanto a su versatilidad y eficacia con respecto a métodos tradicionales de programación. La visión abstracta que presenta la arquitectura Model – Driven permite al desarrollador mirar al problema de una forma más genérica y resolver el mismo en base al establecimiento de modelos lógicos que permitirán a su vez llegar al producto final deseado. El desarrollo impulsado por modelos [4], nos permite tener la visión de crear sistemas de software complejos a partir de modelos abstractos, para así poder transformarlos en implementaciones concretas [5]. Existen sin embargo conjeturas en las que se plantea que la derivación de una implementación no trivial, completa a partir de modelos por sí misma no es factible [6].

Sin embargo, las técnicas empleadas de MDD en la realización del estudio [7], requieren que los desarrolladores de herramientas integren el código generado que se encontraba diseñado manualmente. Para ello se han implementado varios mecanismos, sin existir uno que sea considerado como un estándar; es por esta razón que siempre se va a depender del contexto en el que se deba llevar a cabo esta integración de código y de las necesidades concretas para poder así emplear los diversos mecanismos.

Por esta razón se han implementado diversas alternativas, acorde a las situaciones requeridas para la implementación o fusión de herramienta MDD con códigos a mano, estos criterios han sido basados a una década de experiencias en ingeniería de software orientada a objetos e investigación de MDD [8] [9], desarrollo de generadores de código e investigación de integración de código [10] y robótica [11].

Se debe considerar que en la última década se ha impulsado el desarrollo por modelos y se han realizado grandes avances, mejorando significativamente las herramientas de apoyo con las que cuentan los modelos. Estas herramientas forman parte de una segunda generación de instrumentos que buscar mejorar el desarrollo impulsado por modelos. A esta nueva realidad, se suma una gran falta de comparativas de nuevas técnicas de desarrollo impulsado por modelos con el desarrollo de software centrado en código en términos de eficiencia, calidad de código y esfuerzo de tiempo [12].

La ingeniería del software se ha caracterizado por continuos aumentos en el nivel de abstracción, esto ha facilitado el camino para permitir resolver problemas de mayor tamaño y complejidad por parte de los desarrolladores, de esta manera la arquitectura dirigida por modelos propuesta por el OMG[1], separa el diseño de la arquitectura y de las tecnologías de la construcción del software, esto favorece una modificación que resultaría independiente en ambas partes [13].

[1] **OMG.** Object Management Group

Las herramientas que soportan un enfoque o arquitectura Model-Driven, permiten a su vez la elaboración de modelos conceptuales como la generación de código automático, basado en el uso de estándar impulsados por OMG. El enfoque MDD[2] nos recomienda que se debería automatizar la generación de código en el mayor grado posible. Sin embargo, existen en el mercado una gran cantidad de enfoques diversos que permiten aplicar este paradigma como son Integranova, WebRatio, Genexus y OOHDM; los mismos generan sistemas totalmente funcionales por medio de transformaciones automáticas. Existen, sin embargo, otras propuestas que generan gran parte del sistema como NDT [14] que nos permite generar el código que soporta el comportamiento y la persistencia, sin embargo, gran parte de la interfaz del usuario debe ser implementada de una forma manual [15].

Cabe denotar que existen estudios relacionados a comparativas entre herramientas Model-Driven, sin embargo, aún queda mucho camino por recorrer en este apartado, se reconoce que es un campo de la industria informática que presenta grandes retos a futuro y brillantes resultados en el presente.

Se puede enumerar ciertos estudios como la comparativa entre WebRatio cuando aún poseía un enfoque WebML con la herramienta jABC de jETI, en la que se denota que ambas soluciones se encuentran enfocadas en modelos, esto permite a su vez diseñar el intérprete en un lenguaje de modelado gráfico de alto nivel que permita una derivación ejecutable entre ambos modelos [16].

La comparativa se realizó basado en el flujo de trabajo del sistema, manejo del modelo de datos, manejo de servicios, heterogeneidad de los servicios, manejo de mensajes XML, ejecución del sistema modelado y, finalmente, control de mecanismos y de estados. Como conclusiones a esta comparativa se pudo determinar que se presentaron las dos soluciones en un escenario Web Semántico. Se determinó que ambas herramientas presentaban similitudes, sin embargo, ofrecen dos puntos de vista diferentes sobre el problema del intérprete, tanto en modelado en tiempo de diseño de la solución como de la plataforma de la ejecución. Se pudo concluir que jABC ofrece una visión más abstracta y sintética de la solución dejando de lado algunos detalles de la conexión de la comunicación. Su modelado se encuentra manejado por el control y apunta a la integración transparente de los servicios en entornos heterogéneos. Para el analista le resulta igual qué tipo de servicios, sean estos locales o remotos, se emplean o cómo se implementan los mismos, todos ellos aparecen de manera uniforme.

En el aspecto de WebML se concluye que el mismo ofrece una cobertura más amplia de los detalles técnicos con respecto a la eficiencia de trabajo en tiempo de ejecución. El enfoque de WebML se encuentra basado en la ingeniería de software y en las prácticas de ingeniería web, mientras que jABC está más centrado en los campos de diseño de servicios SOA y Web. Ambos métodos no son originalmente destinados a hacer frente a las aplicaciones de Web Semántica, pero ambos demostraron adaptarse bastante bien a esta nueva clase de problemas [16].

Cabe reconocer que la antigüedad del estudio anteriormente citado no pudo ver los avances en la herramienta de WebRatio, el cual tiene grandes avances en lo referente a Web Semántica e Internet de las Cosas, esto lo ha realizado evolucionando su lenguaje

[2] **MDD.** Model Driven Development

de modelado a una versión IFML, la misma se encuentra adoptada y reconocida por la OMG.

Existen otros estudios que lejos de ser comparativas, realizan una evaluación exhaustiva de usabilidad en términos de eficiencia, eficacia y satisfacción [17], estos estudios se encuentran centrados en la facilidad y adaptabilidad de herramientas Model-Driven así como en la reducción de costos que estas representan así como en abaratamiento en tiempos de desarrollo. En este estudio se presente un análisis de la usabilidad de la herramienta de Integranova con respecto al desarrollo tradicional, cabe denotar que concluye que en procesos sencillos no existe una mejora consistente con el uso de una herramienta Model-Driven como Integranova, esta diferencia se ve marca en procesos más complejos y que no solo causaron problemas en usuarios inexpertos sino también en aquellos que presentaban conocimientos en el área del desarrollo del software.

Este estudio permite establecer ciertos parámetros para la realización de nuestra comparativa, puesto que se toma como punto de partida la facilidad que debería ofrecer las dos herramientas a los analistas, estos deberían potenciar los trabajos sencillos y monótonos para así, dar mayor importancia, a aquellos que posean un nivel de complejidad más elevado.

Esto permitiría al analista distinguir la lógica que desee aplicar a determinado procesos y centrar sus esfuerzos en la obtención de resultados de objetivos complejos y de gran capacidad de requerimientos.

En el estudio en concreto del análisis de la herramienta Integranova, parte de una serie de recomendaciones en varias áreas como es la prevención y manejo de errores, consistencia y estética, interacción con la aplicación, modelado de objetos y de funcionamiento en general, que van de moderados hasta cambios críticos. Cabe mencionar que estos parámetros se encuentran focalizados netamente en un estudio enfocado hacia la usabilidad [17].

2.2. Retos en la Ingeniería Model-Driven

2.2.1. Escalabilidad

El problema de la escalabilidad está muy ligado al desarrollo de grandes empresas que requieren sistemas complejos, esto implica la necesidad de modelos cada vez más específicos que forman la base de la representación y razonamiento con la cuál fue implementada la solución. Sumado a ello, el desarrollo suele llevarse a cabo en un contexto distribuido e involucra a muchos desarrolladores con diferentes roles y responsabilidades. Las problemáticas encierran que a estas grandes capacidades de desarrollo, se requiera de herramientas que soporten los diseños y que además permita realizar cambios incrementales por eventuales cambios y que estos no generen toda la solución nuevamente sino que se incorporen a su funcionamiento actual [18].

Otra de las problemáticas consiste en el trabajo colaborativo, esto influye en el ámbito de sincronización de cambios locales generados por los diseñadores y que los mismos

puedan ser subidos al aplicativo central. Esta situación debería ser solventada en caso de existir algún conflicto con la copia maestra antes de ser incorporado al sistema general.

Estos problemas fueron solventados por medio del entorno de desarrollo denominado JDT[3], que proporciona un entorno en el que los desarrolladores pueden gestionar enormes bases de código que consisten en decenas de miles de archivos de código fuente Java. JDT soporta la comprobación y compilación incrementales manteniendo la consistencia de la programación. Esto se da de tal manera que solo se vuelve a validar y compilar esa clase y las que se encuentren afectadas por el cambio más no todas las clases de proyecto o espacio de trabajo. Finalmente, JDT es ortogonal al control de versiones, desarrollo colaborativo y maneja herramientas multitarea tales como CVS[4], SVN[5] y Mylyn[6] [18].

2.2.2. Manejo del incremento de volumen

Una de las problemáticas más comunes del manejo de aplicaciones basadas en modelos se encuentra en su crecimiento, puesto que, a medida que estas crecen, las herramientas que los diseñan deben escalar proporcionalmente. Una problemática que se puede ver claramente se encuentra centrada en el compilador Java, puesto que no permite métodos cuyos cuerpos excedan los 64 Kb, sin embargo, en el código de desarrollo de dominios esto genera un inconveniente. La razón de esto es debido a que, en el desarrollo de código, manejar toda la programación embebida en un solo método es considerado como una mala práctica. Sin embargo, cuando hablamos de modelos, es razonable que un modelo encapsule varios miles de elementos en un mismo archivo [18].

Para poder tener un mejor control al tamaño creciente de los modelos y sus aplicaciones, existen marcos de trabajo como EMF[7] que permite realizar una carga *lazy*[8] así como también existen otros enfoques como Teneo[9] y CDO[10] para modelos persistentes de bases de datos. Aunque son enfoques útiles para la práctica, resultaron ser soluciones temporales que intentan compensar la falta de construcciones para encapsulación y modularidad en los lenguajes de modelado. El problema que se tiene previsto a largo plazo no será gestionar modelos monolíticos grandes sino llegar a separarlos en modelos

[3] **JDT.** Java Development Tools
[4] **CVS.** Concurrent Versions Systems. Aplicación informática que implementa un control de versiones almacenando el registro de todo el trabajo y los potenciales cambios en los ficheros que forman parte de un programa y permite que distintos desarrolladores colaboren.
[5] **SVN.** Apache Subversion. Es una herramienta que permite controlar versiones considerada como Open Source. Esta se encuentra basada en un repositorio cuyo funcionamiento es similar al de un sistema de ficheros
[6] **Mylyn.** Es un entorno de administración del ciclo de vida de aplicaciones para Eclipse que facilitan el trabajo del desarrollador permitiendo mantener su eficiencia en trabajos multitareas.
[7] **EMF.** Eclipse Modeling Framework. Es una herramienta de modelado y generación de código para la contrucción de aplicaciones basadas en un modelo de datos estructurado [51].
[8] **Lazy.** Es un patrón de diseño que permite retrasar la carga o inicialización de un objeto hasta el momento mismo de su utilización, esto mejora la eficiencia de los programas puesto que no carga objetos que podrían no llegar a utilizarse.
[9] **Teneo.** Es una solución que en base a persistencia se emplea para EMF que utiliza Hibernate. Es compatible con la creación automática de EMF hasta sus mapeos relacionales [53].
[10] **CDO.** Es una tecnología para los modelos EMF compartidos y distribuidos, funciona también como una solución cartográfica O/R basada en el servidor [52].

modulares pequeños y reutilizables, de manera similar a las prácticas seguidas en desarrollos tradicionales definidas hace más de 40 años [19].

2.2.3. Desarrollo Colaborativo

Un requisito para un entorno MDE con enfoque industrial es permitir el desarrollo colaborativo de modelos. Más específicamente, se espera que cada desarrollador pueda ver una parte arbitraria del modelo, modificarlo y luego devolver los cambios a la copia maestra o repositorio. Una vez más, la formulación de este requisito es impulsada por el estado actual que normalmente implica la construcción y el trabajo con grandes modelos monolíticos. Con mayor modularidad y encapsulación, los modelos grandes se pueden separar en modelos más pequeños, que pueden ser gestionados utilizando herramientas de desarrollo colaborativo existentes, como CVS y SVN. Dada la criticidad de los sistemas de control de versiones en el contexto empresarial, los usuarios enfocados en un ámbito industrial, son particularmente reacios a cambiar a un nuevo sistema de control de versiones. Por lo tanto, las soluciones radicalmente diferentes como los repositorios de modelos dedicados, que no se basan en una base sólida y probada, son muy poco probables de ser utilizados en la práctica [18].

2.2.4. Capacidad Incremental

La capacidad incremental en la gestión del modelado se busca principalmente por medio de los enfoques de transformación puramente declarativas [20]. Esto implicaría que una transformación puramente declarativa puede ser analizada automáticamente para determinar los efectos de un cambio en el modelo fuente al modelo objetivo. La experiencia ha demostrado que las transformaciones incrementales son de hecho posibles, pero su aplicación se limita a escenarios donde los lenguajes de origen y de destino son similares entre sí, y la transformación no implica cálculos complejos. JDT logra esta capacidad incremental sin usar un lenguaje declarativo para compilar el código fuente de Java a bytecode; en su lugar, mantiene el componente Java, que es un lenguaje imperativo. A diferencia de la mayoría de los lenguajes de modelado, Java tiene un conjunto de reglas de modularidad y encapsulación bien definidas que, en la mayoría de los casos, evitan que los cambios introduzcan efectos de variación extensos [18].

2.2.5. Modularidad

Para conseguir enfrentar el reto de la escalabilidad se denotaron en los puntos anteriores la importancia de la modularidad y encapsulación. Existen dos aspectos relacionados con la modularidad de un modelado: por un lado, las capacidades ofrecidas por la herramienta establecida, mientras que, por otro lado, el diseño del lenguaje del modelado utilizado. A continuación, se podrá apreciar como cada uno de estos aspectos afectan a la modularidad e imaginamos las capacidades que desearíamos de un framework de desarrollo de modelos.

2.2.5.1. Capacidades de herramientas de modelado

En las herramientas de modelado actuales, existen tres formas de obtener las relaciones entre dos elementos de un modelo: capacidad contenedora, referencias directas y referencias indirectas. La capacidad contenedora es la relación natural de un elemento que es resulta ser la composición de otra relación, una referencia directa es aquella que se encuentra basada en un identificador único que puede ser resulto directamente por la herramienta de modelado y, finalmente, una referencia indirecta es aquella que necesita de un algoritmo de resolución explícita para poder ejecutada [21].

Para poder permitir a los usuarios dividir a los modelos en varios archivos físicos, las herramientas de modelado contemporáneos admiten la contención entre modelos (es decir, su capacidad para contener otro a pesar de estar almacenado en archivos físicos diferentes). Con respecto a las referencias directas que no se encuentren embebidas, se las suele ofrecer porque pueden ser resueltas automáticamente por la herramienta de modelado y, por lo tanto, permiten una navegación fluida sobre los elementos del modelo. Sin embargo, se ha considerado en algunos estudios [18] que las referencias directas son poco recomendadas para la modularidad dado que permiten el acoplamiento entre diferentes partes del modelo y evitan que los diseñadores puedan trabajar de manera independiente en varias partes. Por otra parte, las referencias indirectas permiten una separación correcta de los fragmentos del modelo, sin embargo, necesitan de algoritmos de resolución personalizado que deben implementarse desde cero a cada momento que se los requiera.

2.2.5.2. Diseño del lenguaje

Con la llegada de tecnologías como EMF y GMF[11], la implementación de un lenguaje de modelado específico y soportar editores gráficos y textuales es un proceso relativamente sencillo y muchas personas y organización han comenzado a definir lenguajes de modelado personalizados para aprovechar las ventajas del contexto específico de las DSL[12]. Al diseñar un nuevo lenguaje de modelado, la modularidad se convierte en su principal reto. Los diseñadores del lenguaje deben asegurarse de que los modelos capturados por medio de DSL se puedan separar en modelos más pequeños proporcionando construcciones de empaquetado de elementos de modelado adecuadas. Estas construcciones no pueden ser parte del dominio, por lo tanto, no pueden ser fácilmente previsibles. Por ejemplo, cuando se diseña una DSL para modelar bases de datos relacionales, es común que se descuide la forma de como se está empaquetando, dado que las bases de datos relacionales son típicamente una lista plana de tablas. Sin embargo, cuando se emplea el lenguaje para diseñar una base de datos con cientos de tablas, tener la capacidad de agruparlas en paquetes conceptualmente coherentes se convierte en un punto clave para la manejabilidad y comprensibilidad del modelo [18].

[11] **GMF.** Graphical Modeling Framework. Es una herramienta que nos permite establecer en enfoque basado en modelos mediante un entorno gráfico [54].
[12] **DSL.** Lenguaje de dominio específico.

2.2.6. Usabilidad

En lo que refiere al avance de las aplicaciones web, se ha tenido grandes limitaciones con respecto a la usabilidad e interactividad de sus interfaces de usuario, para superar estas limitaciones han aparecido nuevos tipos de aplicaciones web conocidas como RIA[13], estas aplicaciones permiten proporcionar componentes gráficos más eficientes, muy similares a las aplicaciones tradicionales de escritorio. Sin embargo, las RIA son bastante complejos y su desarrollo requiere de tareas de implementación y diseño que consumen gran cantidad de tiempo y son propensos a errores. El desarrollo de RIA es un nuevo desafío para las nuevas metodologías de ingeniería web que requieren su modificación y la introducción de nuevas necesidades.

Existen, en este contexto, algunas alternativas en la actualidad que proponen nuevos enfoques como OOH4RIA que propone un proceso de desarrollo impulsado por modelos que extiende la metodología OOH. Esta introduce nuevos modelos estructurales y de comportamiento para representar un RIA completo y aplicar transformaciones que reducen el esfuerzo y aceleran su desarrollo implementado en el framework de GWT[14].

2.3. OOH4RIA

Para definir la herramienta OOH4RIA se ha basado en un estudio en el que se propone un proceso de desarrollo basado en un modelo que permite especificar una RIA casi completa, a través de la extensión de las metodologías tradicionales de la Web como OOH.

El proceso comienza con el diseñador de OOH que define el modelo de dominio respectivo para representar las entidades de dominio y las relaciones entre ellas. A partir de este modelo, el diseñador representa la navegación a través de los conceptos de dominio y establece las restricciones de visualización utilizando el modelo de navegación. Con la definición de ambos modelos, OOH comienza la parte del proceso que se define a continuación: primero transforma el modelo de navegación en el modelo de presentación mediante la transformación PIM2PSM denominada Nav2Pres. Este modelo de presentación está específicamente definido para la plataforma GWT, permitiéndonos capturar claramente los diferentes widgets que constituyen una interfaz GWT. El Nav2Pres es una transformación modelo-modelo definida en QVT que establece las diferentes capturas de pantalla de la presentación del modelo.

Después de obtener las capturas de pantalla del contenedor del modelo de presentación, el diseñador de interfaz de usuario las completa colocando los widgets, definiendo el estilo y estableciendo la configuración espacial por medio de paneles. Vale la pena señalar que estos widgets pueden estar relacionados con un elemento de navegación, mostrando así el contenido dinámico procedente del lado del servidor en la interfaz de usuario.

Dado que el RIA posee una interfaz de usuario interactiva rica similar a las aplicaciones de escritorio, debemos completar las características estáticas de widgets con un modelo

[13] **RIA.** Rich Internet Application
[14] **GWT.** Google Web Toolkit

que nos permitirá especificar la interacción entre estos widgets y el resto del sistema. Este modelo ha sido llamado modelo de orquestación y está representado como un perfil UML del diagrama de estado. El modelo de orquestación no tiene que definirse a partir de este punto dado que una transformación de modelo a modelo llamada Pres & Nav2Orch nos permite obtener el esqueleto del modelo de orquestación a partir de los modelos de navegación y presentación.

La transformación de modelo a modelo comienza estableciendo los estados de comportamiento de las capturas de pantalla y sus transiciones desde los nodos de navegación y las asociaciones, respectivamente. También define los estados de comportamiento del widget correspondiente a los widgets representados en el modelo de presentación. En este punto, el diseñador completa el modelo de orquestación que presenta los eventos, operaciones y disparadores de diferentes estados.

El último paso consiste en definir las transformaciones de modelo a texto que nos otorgarán la implementación RIA. La transformación GWT Server Side genera el código del servidor del dominio OOH y los modelos de navegación, mientras que el lado del cliente genera el código utilizando un marco GWT específico. Ambas transformaciones se escriben en el lenguaje MOFScript que sigue a la RFP de ModelToText de OMG para la representación de transformaciones de modelo a texto [22].

2.4. WebML

La especificación de una aplicación Web en WebML [23] consiste en un conjunto de modelos ortogonales: el modelo de datos de la aplicación (un modelo Entidad-Relación extendido), uno o más modelos de hipertexto (es decir, vistas de sitio diferentes para diferentes tipos de usuarios) por medio de los cuales se pueden expresar los caminos de navegación y la composición de la página; y el modelo de presentación, que nos permite describir el aspecto visual de las páginas. El modelo de presentación resulta ser muy interesante, ya que permite "vestir" un modelo de hipertexto para obtener páginas Web con el diseño y apariencia deseados para cualquier tecnología.

Un modelo de hipertexto consiste en una o más vistas de sitio, cada una de ellas orientada a una función de usuario o dispositivo cliente específico. Una vista de sitio es una colección de páginas (posiblemente agrupadas en áreas con fines de modularización). El contenido de las páginas se expresa mediante componentes para la publicación de datos (denominados unidades de contenido). La lógica de negocio activada por la interacción del usuario se representa en su lugar por secuencias de unidades de operación, que denotan componentes para modificar datos o para realizar acciones empresariales arbitrarias (por ejemplo, enviar correo electrónico). Las unidades de contenido y operaciones están conectadas mediante enlaces, que especifican el flujo de datos entre ellos y el flujo de proceso para calcular el contenido de la página y para ejecutar la lógica de negocio, en reacción a los eventos de navegación generados por el usuario [24].

Se considera, por ejemplo, un escenario simple enfocado al caso práctico que se va a realizar, partimos de la necesidad de crear nuevos tipos de pago, iniciando de una página donde se encuentran listados los diferentes tipos de pago, por medio del mismo listado se alimentan dos diferentes caminos que resultarían ser la modificación del tipo de pago y la eliminación de un tipo de pago. Adicional al listado que se encuentra en la página, se

tiene también la acción de crear un nuevo tipo de pago y también un mensaje de confirmación.

Como se puede observar, el siguiente ejemplo contiene unidades para la publicación de contenidos (unidades de datos e índices), que muestran algunos atributos de una o más instancias de una entidad determinada. Por cuestión de sintaxis, cada tipo de unidad tiene un ícono distinguido y el nombre de la entidad se especifica en la parte inferior de la unidad, debajo del nombre de la entidad, en caso de ser necesario, se cargan todos los predicados (conocidos como selectores) que expresan condiciones que filtran las instancias de la entidad que se mostrarán. En el ejemplo de la Figura 2, se puede observar que se manejan unidades de contenido estático, que muestran contenido fijo que no proviene de los objetos del modelo de datos, tal es el caso de la unidad de *Actions* que nos permite almacenar los botones de acción como *New* que nos permitirá crear una nueva instancia de información, otro ejemplo de contenido estático es la unidad de mensaje *Confirmation message* en la que se presentará un mensaje de información cuando el contenido haya sido almacenado correctamente.

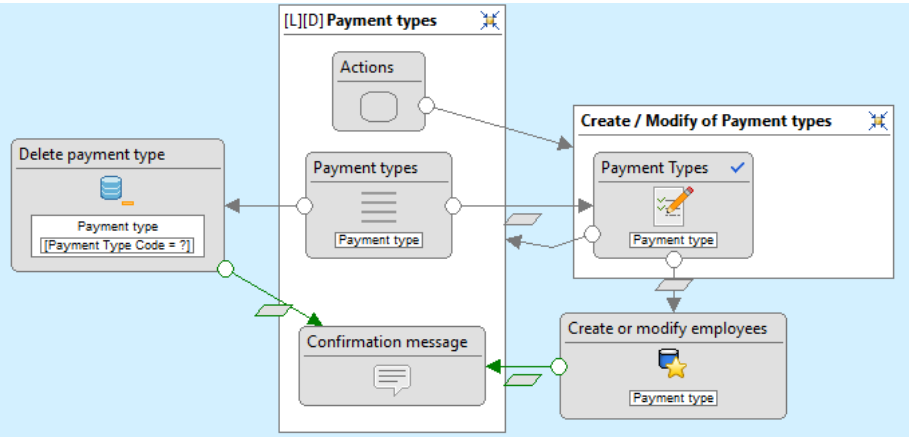


Figura 2. Ejemplo de mantenimiento de Tipos de Pago.

El modelo de hipertexto también nos permite ilustrar la utilización y empleo de las unidades de operación, en nuestro ejemplo en particular, podemos observar dos unidades de operación como son las de creación o modificación, *Create or modify employees*, como se puede apreciar en esta unidad en la parte superior se le añade un título informativo que nos indica que es lo que realiza la unidad, se colocó una etiqueta diferente para denotar que este proceso es netamente informativo puesto que la creación o modificación a llevarse a cabo es de un tipo de pago más no de un empleado, esta información viene indexada en la parte inferior de la unidad, como se había explicado anteriormente.

También existe una unidad de eliminación, como se puede notar, aquí ya se pone de manifiesto el uso de un predicado o selector, el cual va a recibir por medio del enlace el identificador de la tupla a ser eliminada para poder así realizar la ejecución del script en la base de datos.

Como dato adicional, cabe mencionar que la unidad de creación y modificación permite realizar dos acciones dependiendo de la información que le ingrese, en caso de no enviarse un identificador como clave primaria se realizaría una inserción nueva, pero si

se envía como parámetro un identificador, cargaría primero esa tupla y actualizaría la información contenida.

WebML distingue entre enlaces normales, de transporte y automáticos. Los enlaces normales (indicados por flechas sólidas) permiten la navegación y se representan como anclajes de hipertexto o botones de formulario, mientras que los enlaces de transporte (indicados por flechas discontinuas) permiten sólo el paso de parámetros y no se representan como widgets navegables. Los enlaces automáticos son enlaces normales, que son automáticamente recorridos por el sistema en carga de página.

Ortogonalmente, los enlaces pueden clasificarse como contextuales o no contextuales: los vínculos contextuales transfieren datos entre unidades, mientras que los enlaces no contextuales permiten la navegación entre páginas, sin parámetros asociados. Las unidades de operación también demandan otros dos tipos de enlaces: enlaces OK y enlaces KO, respectivamente, que denotan el curso de acción tomado después del éxito o fracaso en la ejecución de la operación. En el ejemplo de la Figura 2 tenemos que:

- El enlace que va de *Actions* hacia una nueva ventana no es contextual, puesto que no debe enviar información alguna al formulario para que este pueda ser creado dado que sería una nueva instancia.
- En contrapartida, el enlace de modificación que va desde el listado hasta el formulario si es un enlace contextual puesto que debe precargar los valores originales que tenía la tupla en el formulario para que el usuario pueda realizar los cambios que requiera y mandarlos a la unidad de Guardado.
- El enlace de eliminación también es un enlace contextual puesto que está enviando el identificador a la unidad de eliminación para que la tupla sea eliminada.
- En lo referente a la unidad de formulario podemos ver que tiene dos enlaces, un enlace contextual y uno no contextual. El enlace contextual es el que va hacia la unidad de crear o modificar un nuevo tipo de pago, es contextual puesto que la unidad requiere de la información que va a insertar en la base de datos.
- En enlace no contextual que tiene el formulario es el de *Cancel* o cancelar, puesto que este permite regresar a la pantalla original y no necesita enviar ningún tipo de información sino más bien conserva la estructura original sin realizar cambio alguno.
- También se pueden observar los enlaces de OK al finalizar el proceso realizado por las unidades de operación como lo que son la eliminación y el guardado, estos enlaces permiten enviar un mensaje de confirmación a la pantalla inicial en el que se indique que el tipo de pago ha sido guardado o eliminado satisfactoriamente en caso de haber sido así.

El contenido de una unidad depende de sus enlaces de entrada y selectores locales. En general, pueden usarse condiciones lógicas arbitrarias, pero las expresiones conjuntivas se presentan fácilmente en los diagramas, donde cada conjunto es un predicado sobre el atributo de una entidad o rol de relación.

Como ya se mencionó, WebML se asocia con un algoritmo de cálculo de páginas derivado de la definición formal de la semántica del modelo (basada en los diagramas de estado [25]). El punto esencial es el algoritmo de cálculo de páginas, que describe cómo se determina el contenido de la página después de un evento de navegación producido por el usuario. El cálculo de páginas equivale a la evaluación progresiva de las distintas

unidades de una página, partiendo de parámetros de entrada asociados con la navegación de un enlace. Este proceso implica la propagación ordenada del valor de los parámetros de enlace, desde un conjunto inicial de unidades, cuyo contenido es computable cuando se accede a la página, a otras unidades, que esperan la entrada de enlaces automáticos o de transporte que salen de las unidades ya calculadas de la página.

En WebML, las páginas son la unidad fundamental de cálculo. Una página WebML puede contener varias unidades enlazadas entre sí para formar un gráfico complejo, y se puede acceder mediante varios enlaces diferentes, procedentes de otras páginas, desde una unidad dentro de la propia página o desde una operación activada desde la misma página.

3. Marco Teórico

3.1. Introducción

Integranova es una herramienta que permite la edición y validación de modelos conceptuales OO-Method, contando entre sus principales características al soporte de los cuatro modelos de UML (objetos, dinámico, funcional y presentación), soporte al modelado de vistas legadas, asistente para la escritura de fórmulas, validación automática de todos y cada uno de los modelos generados, soporte al modelado cooperativo, generación automática de documentación acerca de un modelo y de métricas para medir el tamaño funcional asociado a un modelo [13].

El enfoque OO-Method no es más que un método de ingeniería Model-Driven. Que permite generar sistemas completamente funcionales partiendo de un modelo conceptual. A partir del mismo y mediante las transformaciones correspondientes, un compilador de modelos permite generar el código necesario para el sistema. Por medio de las cuatro vistas complementarias que se enumeraron anteriormente, se puede desarrollar que todos los aspectos funcionales de una aplicación puedan ser descritos de una manera abstracta gracias a las denominadas primitivas conceptuales que tienen una semántica precisa. Luego de su análisis se puede notar que gran parte de estas primitivas conceptuales poseen una notación gráfica basada en UML, esto permite ocultar la complejidad y formalismo de la especificación OASIS subyacente.

Hay que entender, de la misma manera, que se requieren de entornos avanzados de desarrollo de software que garanticen la generación ágil de productos software que sean confiables y se encuentren diseñados mediante esquemas conceptuales. Esto representaría un salto gigantesco a la masificación de desarrollo de aplicativos industriales basadas en un entorno model-driven.

OO-Method se encuentra basado en las ideas abajo expuestas y que permite proporcionar un enfoque metodológico para la construcción de un esquema conceptual que se encuentre orientado a objetos, con su respectiva representación en un entorno de desarrollo comercial, obteniendo de esta manera un producto de software de calidad. Para ello OO-Method se encuentra basado en los siguientes principios [26]:

- Priorizar las nociones de modelo conceptual orientadas a objetos y realizar este proceso en un marco en el que todas las primitivas conceptuales requeridas se encuentren definidas de manera precisa.
- Integración de métodos formales y convencionales de aceptación industrial.
- Proporcionar un entorno avanzado de producción de software que incluya la generación de código completo para la arquitectura de software proporcionada por el entorno de desarrollo comercial. Por "completo", queremos decir que los aspectos estáticos, dinámicos y de presentación que se especifican en el Esquema Conceptual se encuentren representados de una forma completa y correcta en el producto final correspondiente.

En el proceso de desarrollo sugerido por OO-Method se relaciona el conjunto de propuestas que, en estos días, favorecen la noción de agilidad, dando una principal importancia a la creación de procesos de software que proporcionan soluciones pragmáticas, las mismas que son fáciles de usar en entornos de producción de software industrial. El proceso de desarrollo se encuentra comprendido por las siguientes fases principales:

1. Modelado Conceptual (Espacio del Problema)
2. Generación del producto de software (espacio de la solución) mediante la aplicación de un modelo de ejecución abstracta.

En la primera fase del modelo conceptual se busca obtener y representar el conjunto de propiedades esenciales del sistema de información en estudio, creando así el correspondiente esquema conceptual. Dado que los patrones conceptuales o primitivos, conceptuales utilizados en su creación, tienen un apoyo formal, podemos obtener la especificación formal orientada a objetos (escrita en el lenguaje OASIS) en cualquier momento. En la práctica, esta especificación constituye un repositorio de alto nivel o diccionario de datos del sistema.

En la segunda fase, se ofrece un modelo de ejecución preciso, en el que se incluye una estrategia de generación de código que determina qué representaciones de software corresponden a cada patrón conceptual de un esquema conceptual. Las implementaciones de estas correspondencias permiten la creación de un compilador de esquemas conceptuales que, para una fuente dada del mismo, producirá automáticamente el producto de software correspondiente. Este producto de software incluirá todos los aspectos estáticos, dinámicos y de presentación que se especifican en la fase de modelado conceptual. Por lo tanto, podemos concluir que el modelo de ejecución determina cómo los conceptos capturados en el esquema conceptual están representados en un entorno computacional.

3.2. OO-Method

OO-Method es un método que se encuentra orientado a objetos de producción automática de software, cuyo principal objetivo consiste en la unión de las propiedades positivas de los lenguajes de especificación formales con el nivel de abstracción de los métodos informales de análisis y diseño de software. El mismo realiza este proceso mediante diagramas gráficos basados en notación UML, esto requiere una semántica precisa y que se encuentre soportada por el lenguaje formal OASIS [27] [28]. Mediante esta metodología, es posible ocultar la complejidad que incide en la especificación formal textual por medio de diagramas específicos. También se ha reconocido que los analistas son mucho más reacios a emplear herramientas textuales que no tienen un buen escalado en proyectos grandes, mientras que muestran mayor empatía en lo referente al uso de herramientas gráficas de especificaciones [29].

Para conceptualizar de una manera más sistemática partimos de que el modelo conceptual de OO-Method se compone de cuatro puntos de vista complementarios [30]:

- **El modelo de objetos:** Donde se especifica la estructura del sistema en términos de clases de objetos (con atributos y servicios) y sus respectivas relaciones.
- **El modelo dinámico:** Representa las secuencias válidas de eventos para y entre los objetos.
- **El modelo funcional:** Donde se especifica cómo los eventos cambian los estados de los objetos.
- **El modelo de interacción:** Modeliza la interacción entre el sistema y el usuario mediante dos vistas: el Modelo de Interacción Abstracta y el Modelo de Interacción Concreta [31]. El Modelo de Interacción Abstracta representa la interfaz independientemente de los tipos de interacción y las peculiaridades de la plataforma. El modelo de interacción concreta especifica la representación de la interfaz en términos de elementos que pueden ser percibidos por el usuario final.

OO-Method permite realizar la tarea de construcción de software en dos fases.

- **Modelo Conceptual.** Se encuentra situado en el espacio del problema, para la definición de este modelo se brinda al analista de un lenguaje gráfico que permite recoger las propiedades de los requisitos del problema en cuestión. Mediante este modelo es posible cubrir la fase de análisis del problema de una forma completa. Finalmente, y partiendo de esta información, es posible la generación automática de una especificación formal completa en OASIS, tomando en cuenta también el uso de mecanismos de traducción determinados por el método.
- **Modelo de Ejecución.** El modelo de ejecución se encuentra situado en el mundo de la solución, el mismo fija los detalles de implementación en lo referente a la interacción del usuario, la persistencia de los objetos, el control de acceso, etc. Esto lo realiza para cada uno de los elementos del Modelo Conceptual en un entorno de desarrollo particular. Finalmente se obtiene un producto funcionalmente equivalente a todas las especificaciones recogidas del problema en el modelo conceptual.

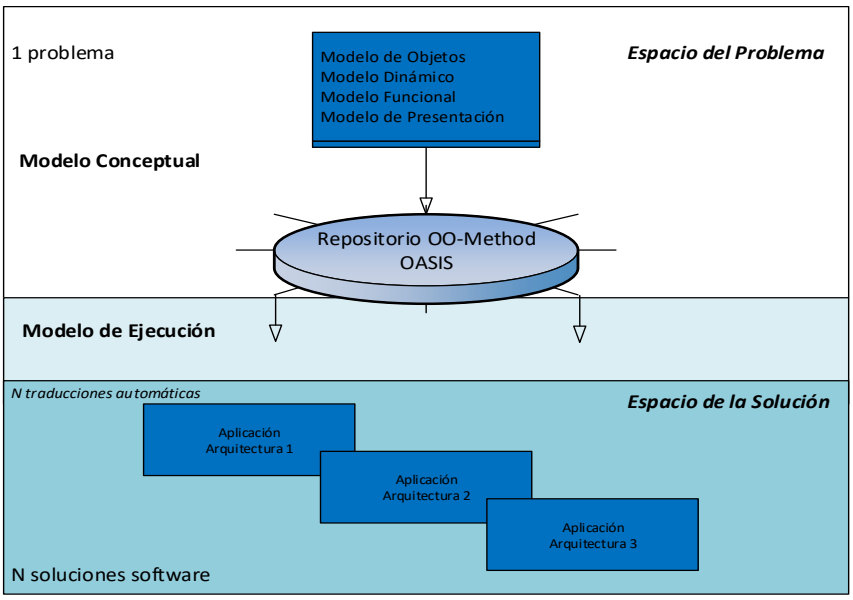


Figura 3. Fases de OO-Method para la producción de aplicaciones según Molina [29]

En la figura 3 se puede observar la relación existente entre los distintos modelos del método. Aquí se encuentran detallados los cuatro modelos que componen el OO-Method en el primer recuadro, modelos que revisaremos a continuación:

3.2.1. Modelo Conceptual

El modelo conceptual es considerado como la primera etapa del proceso de construcción de software. En la misma se permite obtener la información del dominio del problema, esto se encuentra fuera de lo que refiere a aspectos de implementación. Una vez que se ha determinado esto es posible construir los tres modelos que permiten especificar claramente la estructura y el comportamiento de los componentes de la sociedad de Objetos estudiada que son el modelado de objetos, el modelado dinámico y el modelado funcional.

3.2.1.1. Modelado de Objetos

Es un modelo semántico de datos extendido por medio de la declaración de eventos y de agentes activadores de los mismos; estos fijan el marco arquitectónico del sistema desde una perspectiva estática. En este modelado se encuentran declaradas las clases componentes tanto elementales como complejas (herencia y agregación) así como su estructura interna (atributos y servicios). La declaración de agentes especifica dentro del modelado de objetos a los usuarios que se encuentren con el acceso y permiso correspondiente para activar como desactivar servicios ofrecidos por las clases.

3.2.1.2. Modelado Dinámico

En el modelado dinámico se establece la persistencia de los objetos en una clase determinada, esto se realiza mediante el proceso de ordenación temporal de la secuencia de ocurrencia de servicios, tanto eventos como transacciones, así como también especifica la interacción inter-objetual.

3.2.1.3. Modelado Funcional

En el modelado funcional se permite especificar la funcionalidad de cada evento, reconocido como un método atómico de una clase, definiendo cual vendría a ser el efecto sobre el estado del objeto implicando en términos de los atributos que pudiera llegar a modificar.

Como conclusión a los anteriores apartados, podemos denotar que existe una estrecha relación de los modelados puesto que el modelo de objetos se complementa con el dinámico y funcional, de esta manera establece la estructura y comportamiento de los objetos. Los tres modelos definen, desde un punto de vista complementario, las propiedades, tanto estáticas como dinámicas de los objetos identificados en el sistema estudiado [29].

3.2.2. Modelo de Ejecución

El modelo de ejecución es la segunda fase del OO-Method, donde se busca la generación automática de código en un entorno de desarrollo dado, donde se ha implementado la estructura y el comportamiento del sistema, estos aspectos fueron recogidos ya en la fase de análisis del problema. El código producido sigue una pauta común en la que se permite asegurar la existencia de una equivalencia funcional entre las implementaciones con respecto al modelo diseñado originalmente. Son estas pautas las que consideran en el modelo de ejecución [32] [33].

En el modelo de Ejecución se considera al sistema como una Sociedad de Objetos, que no es más que un sistema vivo por medio del cual, los objetos interaccionan entre sí por medio de paso de mensajes. El mismo usuario hace las veces de objeto al ser considerado como una instancia de la clase agente. Para poder alcanzar este comportamiento es necesario realizar los siguientes pasos:

1. **Identificación del Usuario.** Nos permite realizar una identificación del usuario por medio del login del sistema, luego de un proceso de autenticación un objeto agente asignará al usuario conectado ante la sociedad de objetos. Por medio de esto, se le podrá asignar una vista a medida del mismo, dependiendo de los permisos otorgados, así como visibilidad de atributos, servicios o inclusive otros roles, de la clase agente considerada.
2. **Autorización para activación de servicios.** Con todos los atributos visibles, así como también los accesos requeridos, el usuario podrá activar o desactivar cualquier servicio que tenga disponible.

Se debe tomar en cuenta que entre los servicios se incluyen observaciones sobre las consultas y transacciones que ofrecen los otros objetos. Estas activaciones de servicio tienen dos pasos, la construcción del mensaje, así como la ejecución del mismo (en caso de ser posible). Para la fase de la construcción del mensaje, el usuario deberá proporcionar la información siguiente:

1. **Identificación del objeto servidor.** La existencia del objeto es una condición que se encuentra implícita para la ejecución de cualquier servicio, aunque cabe la posibilidad de que nos encontremos frente a un servicio de creación. De ser este el caso el objeto servidor sería el meta-objetivo o clase que oferta el servicio de creación como método constructor para la creación de la instancia. Hay que tomar en cuenta que existen un meta-objeto para cada clase. Este meta-objetivo contiene como atributo principal la población de la clase, el siguiente OID y los citados servicios de creación.
2. **Proporcionar de argumentos al servicio.** Cuando un servicio requiera de argumentos adicionales, estos deben ser proporcionados siempre y cuando sean necesarios.

Cuando hemos creado y enviado el mensaje, la ejecución del servicio se encuentra caracterizada por una ocurrencia de la siguiente secuencia de acciones que se encuentren en el objeto servidor, para ello se realizan los siguientes pasos:

1. **Comprobación de la transición del estado.** En este paso se va a verificar el diagrama de transición de estados (DTE) que se encuentra asociado al objeto

servidor, en donde debería existir una transición válida, en caso de no hacerlo se producirá una excepción y el mensaje será ignorado.

2. **Satisfacción de precondiciones.** La precondición asociada al servicio debe cumplirse, caso contrario, se producirá una excepción y el mensaje sería ignorado.
3. **Satisfacción de Valoraciones.** Las modificaciones que se invocan por medio de los eventos y que se encuentran indicadas en el modelo funcional, son satisfechas, alterando de este modo, el estado del objeto servidor.
4. **Comprobación de restricciones de integridad en el nuevo estado.** Es una forma de confirmación cuando la ejecución de un servicio conduce a un estado válido. Estas restricciones de integridad son verificadas sobre el estado final del objeto y en caso de no ser cumplidas se lanzará una excepción y los cambios realizados en el estado son completamente ignorados descartando el mensaje original, esto permitirá al sistema regresar a su estado original previo a la recepción del mensaje.
5. **Comprobación de acciones automáticas.** Luego de producirse la ejecución de un servicio, las reglas de condición-acción que representan la actividad interna del sistema (triggers), deberán ser comprobadas. En caso de que alguna condición se satisfaga, el servicio especificado será ejecutado.

En los pasos que enumeramos anteriormente se constituye la guía de implementación de cualquier programa derivado de una especificación OO-Method, en esta guía se asegura una equivalencia funcional entre la especificación del sistema de objetos descrito en el esquema conceptual con su correspondiente ejecución en una implementación dada. Uno de los factores claves a resaltar del modelo de ejecución es que el mismo es independiente de su entorno de desarrollo, esto quiere decir que se podrán obtener implementaciones particulares de este modelo para cualquier entorno de producción de software independientemente de la arquitectura de la aplicación destino.

3.3. WebRatio

3.3.1. Introducción

Uno de los principales retos en lo que refiere a la definición de un enfoque basado en modelos se encuentra focalizado en el desarrollo del front-end, puesto que el mismo sigue siendo un proceso costoso e ineficiente, dado que la codificación manual es el enfoque de desarrollo predominante, la reutilización de artefactos de diseño es baja y la portabilidad entre plataformas aún sigue siendo un problema. Es por esta situación que se consideró que la disponibilidad de un lenguaje de modelado de flujo de interacción independiente de la plataforma, puede llegar a aportar varios beneficios al proceso de desarrollo de los front-ends de una aplicación determinada. Esto permitiría la especificación formal de las diferentes perspectivas como son el contenido, composición de interfaz, interacción y opciones de navegación. De igual manera permitiría una conexión con la lógica de negocio y la presentación, separando las preocupaciones de las partes interesadas al aislar la especificación del front-end de sus problemas específicos de implementación. La realización de este proceso permitiría una mejora en el proceso

de desarrollo, fomentando la separación de las incidencias en el diseño de la interacción del usuario, otorgando así una máxima eficiencia en las diferentes funciones de desarrollo; permitiendo una comunicación entre la interfaz y el diseño de interacción hacia partes interesadas no técnicas que validen prematuramente la consolidación de los requisitos.

Para la realización de este proceso se desarrolló el Lenguaje de Modelado de Flujos de Interacción (IFML[15]), el cuál ha sido adoptado como un estándar por el Grupo de Gestión de Objetos (OMG[16]), tomando especial consideración la usabilidad y comprensibilidad del modelo permitiendo que el usuario pueda aprenderlo rápidamente, sea fácil de usar, susceptible de implementación y abierto a la reutilización y extensibilidad.

Dentro de las características que posee IFML se encuentran las siguientes:

- Es conciso, permite evitar la redundancia y reduce el número de tipos de diagramas y conceptos necesarios para expresar las decisiones de diseño de interacción más destacadas.
- Proporciona reglas de inferencia de modelo en el nivel de modelado lo que nos permite aplicar automáticamente patrones de modelado por defecto y permite a su vez generar una inferencia automática de detalles cuando sea necesario.
- Incluye extensibilidad en la definición de nuevos conceptos (por ejemplo, nuevos componentes de interfaz o tipos de eventos).
- Asegura la implementabilidad, es decir, soporta la construcción de frameworks de transformación de modelos y generadores de código para transformarlas en aplicaciones ejecutables para una amplia gama de plataformas tecnológicas y dispositivos de acceso.
- Soporta la definición de patrones de diseño reutilizables que pueden ser almacenados, documentados, buscados y recuperados y reutilizados en otras aplicaciones.

IFML puede ser visto como la consolidación del Web Modeling Language (WebML) [34] el cual ha sido definido y patentado hace unos 15 años como un modelo conceptual para las aplicaciones web de datos intensivos.

Se considera como la principal innovación de WebML la notación de modelos de hipertexto, que permitían la especificación de páginas web consistentes en componentes conceptuales (unidades) interconectados por enlaces conceptuales. El modelo de hipertexto se dibuja en una notación visual simple y bastante intuitiva, pero tiene una semántica rigurosa, que permite la transformación automática de diagramas en el código completo de ejecución de una aplicación web de datos intensivos.

El enfoque del diseño que manifestaba WebML se concentró en la definición de un poderoso conjunto de unidades; con el pasar del tiempo, se evaluó que las unidades son sólo componentes específicos, que pueden definirse y adaptarse a las necesidades de cualquier nuevo desarrollo tecnológico; sin embargo, la esencia del modelo de hipertexto de WebML radica en las reglas para ensamblar componentes con diferentes enlaces en un gráfico y para inferir todas las posibles reglas de paso de parámetros de las interfaces de componentes y los tipos de vínculo.

[15] IFML. Interaction Flow Modeling Language
[16] OMG. Object Management Group

Un gráfico bien formado garantiza el flujo de datos correcto entre las unidades y dicta el orden de ejecución del componente adecuado al calcular el contenido de las páginas.

WebML sufrió un conjunto de extensiones a lo largo de los años, para seguir las principales tecnologías y tendencias de desarrollo de software:

- Servicios web y arquitecturas orientadas a servicios [35].
- Integración con los procesos de negocio [36].
- Personalización, adaptación, conciencia de contexto y movilidad [37].
- Web Semántica y Servicios Web Semánticos [38].
- Aplicaciones de Internet ricas [23].
- Aplicaciones basadas en la búsqueda.
- Soporte de reutilización y modularización.

3.3.2. WebRatio como agente integrador

La plataforma de desarrollo WebRatio ha adoptado IFML a partir de su versión 7.2 como notación oficial. Desde 2001, WebRatio trabajó junto con gerentes de TI, arquitectos web y desarrolladores de Java para generar un apoyo en el desarrollo de nuevas aplicaciones empresariales, la actualización de las existentes y la integración de sistemas abiertos. En la actualidad grandes expertos se encuentran trabajando en mejorar la herramienta enfocada en arquitectura MDD, interesados en el desarrollo impulsado por modelos. Estos esfuerzos están siendo proyectados para clientes corporativos, principalmente en las áreas de finanzas, energía, transporte, gobierno y minoristas.

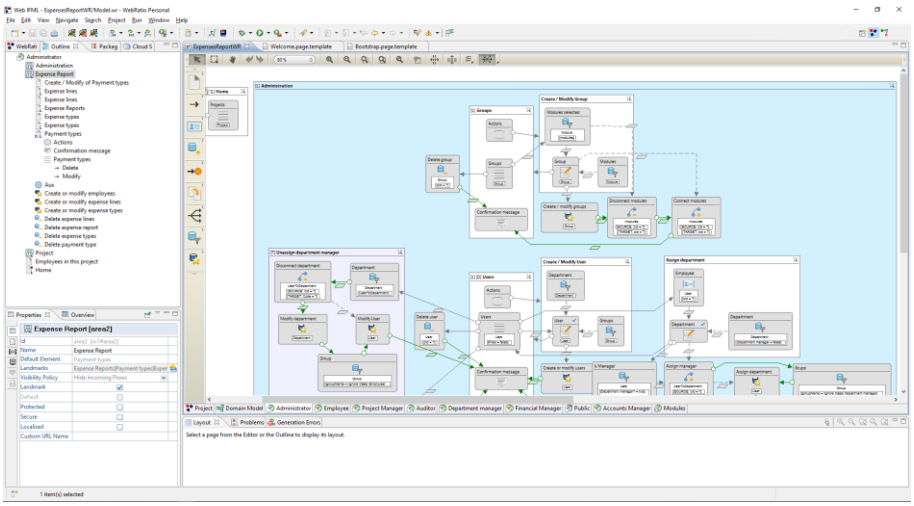


Figura 4. Interface de WebRatio

La interfaz de usuario del editor IFML de WebRatio se muestra en la Figura 4, por medio de ella se permite generar automáticamente tanto aplicaciones Web como móviles con la característica de que las mismas serán rápidas, seguras, escalables y robustas basándose en los modelos IFML y BPMN. WebRatio Platform puede implementar estas aplicaciones, con una interacción personalizada y consistente con el usuario. Produce código tanto del lado del cliente y como código optimizado del lado del servidor.

WebRatio Platform es escalable, fiable y cumple con las más estrictas políticas de seguridad corporativa para desarrollar aplicaciones empresariales B2C[17] y B2E[18] (como portales, sitios de comercio electrónico, soluciones de venta de boletines de asistencia, sistemas de autoservicio de clientes, sistemas de administración de suscripciones de Aplicaciones, etc.). Las aplicaciones generadas son compatibles con HTML5, CSS3 y Java para mantenerse al día con los cambios continuos de código y tecnologías. Las aplicaciones Web construidas con WebRatio Platform cumplen con el estándar Java / JSP 2.0+ y se pueden implementar en cualquier servidor de aplicaciones. Las aplicaciones pueden ser desplegadas automáticamente en las instalaciones o en la nube (pública o privada). Los planes de implementación son totalmente personalizables.

WebRatio Platform se integra con las herramientas de gestión del ciclo de vida de las aplicaciones (Atlera JIRA, IBM Rational Team Concert, etc.). Todos los recursos del proyecto se comparten a través de un servidor de trabajo colaborativo y un servidor de control de versiones (CVS o SubVersion). También ayuda a enfrentar los desafíos de DevOps (Desarrollo y Operaciones) facilitando la colaboración entre los equipos de desarrollo y operación.

Con los componentes personalizados de WebRatio, es posible integrar la aplicación con una variedad de sistemas y servicios como SAP, Tibco, IBM Mainframe y aplicaciones SaaS como Salesforce.com, Dropbox, etc., incluyendo su propio código heredado personalizado. La Plataforma WebRatio facilita la integración de sistemas de TI con soluciones externas y ayuda a invocar y publicar Servicios Web (REST, SOAP), así como con los servicios de inicio de sesión de Single Sign On, compartir contenido, gestión de contactos y redes sociales como Facebook, Twitter, LinkedIn, G+, etc [24].

3.3.3. Personalización a Nivel de Usuario y Conocimiento del Contexto

WebML se ha aplicado también al diseño de aplicaciones Web adaptables y sensibles al contexto, es decir, aplicaciones que explotan el contexto y adaptan su comportamiento a las condiciones de uso ya las preferencias del usuario [37].

En estas aplicaciones, el proceso de diseño se amplía mediante un paso preliminar dedicado al modelado de los perfiles de usuario y de la información contextual.

Los requisitos de usuario y contexto se describen mediante tres modelos diferentes, complementando los datos de aplicación:

- El **modelo de usuario**. Es quien describe los datos sobre los usuarios y sus derechos de acceso a los objetos de dominio. En particular, la entidad Usuario expresa un perfil de usuario básico, la entidad Grupo habilita los derechos de acceso para grupos de usuarios y la entidad Módulo permite a usuarios y grupos obtener acceso selectivamente a cualquier elemento de hipertexto (vistas de sitio, páginas, unidades de contenido individuales e incluso enlaces).
- El **modelo de personalización.** Es quien asocia entidades de aplicación con la entidad de usuario mediante relaciones que denotan preferencias de usuario o

[17] **B2C**. Business to Consumer, es decir, del negocio al consumidor.
[18] **B2E.** Business to Employee, es decir, del negocio al empleado.

propiedad. Por ejemplo, la relación entre las entidades User y Group en la ilustración anterior permite asignar a un usuario un grupo determinado para que pueda tener acceso a las acciones realizadas por ese determinado grupo y que han sido asignadas a su vez por medio de su relación con módulos que pueden contener tanto vistas como módulos de trabajo, siempre y cuando los mismos se encuentren protegidos.

- El **modelo de contexto**. Es el que incluye entidades como Dispositivo, Ubicación y Actividad, que describen propiedades de contexto relevantes para la adaptabilidad. Las entidades de contexto están conectadas a la entidad Usuario para asociar cada usuario con su contexto personal.

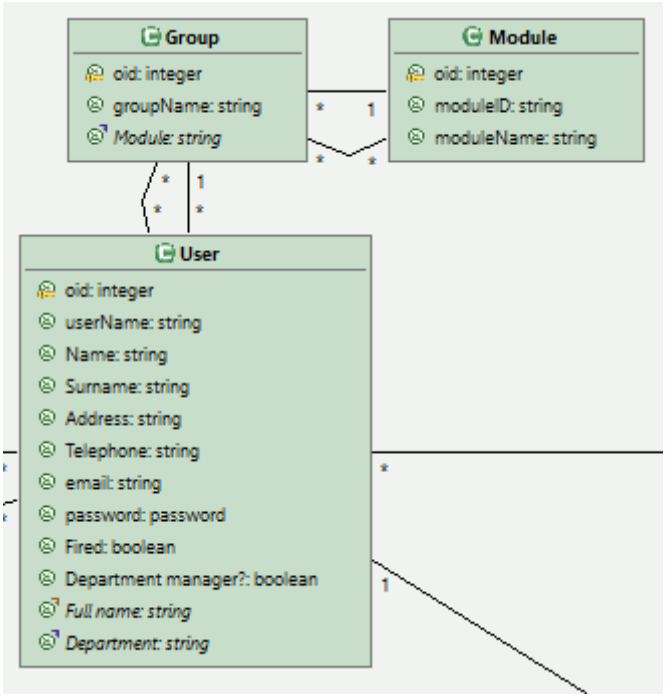


Figura 5. Modelo que representa la gestión de usuarios con roles

Durante el diseño del hipertexto, la conciencia del contexto puede estar asociada con páginas seleccionadas, y no necesariamente con la aplicación completa. Las páginas con reconocimiento de contexto se etiquetan con una etiqueta C (que significa "contextual") para distinguirlas de las páginas convencionales. Las acciones de adaptación se agrupan dentro de una nube de contexto que debe ejecutarse antes del cálculo de la página. Normalmente, las nubes incluyen operaciones WebML que leen los datos de personalización o de contexto y luego personalizan el contenido de la página o modifican el flujo de navegación definido en el modelo.

3.3.4. Web Semántica y Servicios

Tradicionalmente, el solicitante de servicios y el proveedor de servicios están diseñados conjuntamente y luego se encuentran estrechamente unidos cuando se crea una aplicación. El campo de la Web Semántica y los Servicios Semánticos proporcionan paradigmas para enriquecer semánticamente las descripciones sintácticas existentes de contenido, sitios y servicios Web. Para este entorno el solicitante puede buscar, ya sea en el diseño o en tiempo de ejecución, entre una variedad de proveedores habilitados para la Web, eligiendo el servicio que mejor se adapte a los requisitos del solicitante. Esta

vinculación flexible del solicitante y de los proveedores permite crear aplicaciones dinámicas y en evolución, utilizando el descubrimiento automático de recursos, la selección, la mediación y la invocación [24].

Para esto se amplió WebML [39] para generar, además de modelos convencionales (procesos, datos, servicios e interfaces) una gran parte de las descripciones semánticas requeridas por el paradigma de la Web Semántica de forma semiautomática, integrando así la producción y el mantenimiento de la información semántica en el ciclo de generación de aplicaciones.

De igual manera se definió un proceso para el diseño de recursos semánticos extendiendo el proceso de diseño SOA con dos tareas adicionales:

- **Importación de Ontologías**, que permiten reutilizar las ontologías existentes que pueden ser explotadas para describir el dominio de la aplicación web en desarrollo.
- **Anotaciones Semánticas**, para especificar cómo las páginas o servicios de hipertexto pueden anotarse utilizando el conocimiento ontológico existente.

Las primitivas WebML básicas fueron ampliadas con componentes para la consulta y navegación ontológica, explotando el poder expresivo de los lenguajes ontológicos. Estas unidades permiten consultas sobre clases, instancias, propiedades y valores, de igual manera se puede comprobar la existencia de conceptos específicos, para que, luego de una posterior verificación, se permita establecer si una relación se mantiene entre dos recursos. Otras unidades importan contenido de una ontología y devuelven la descripción RDF de una porción dada del modelo ontológico. También han sido introducidas operaciones tales como levantar y bajar, ampliando los componentes de mapeo XML2XML ya desarrollados en el contexto de SOA.

3.3.5. Rich Internet Applications (RIA)

Debido a los requisitos cada vez más complejos de las aplicaciones, las tecnologías Web actuales están empezando a mostrar límites de usabilidad e interactividad. Las aplicaciones de Internet ricas (RIAs) se han propuesto recientemente como la respuesta a tales inconvenientes. Son una variante de sistemas basados en Web que minimizan las transferencias de datos cliente-servidor y que mueven las capas de interacción y presentación del servidor al cliente.

Mientras que en las aplicaciones Web tradicionales de uso intensivo de datos el contenido reside únicamente en el lado del servidor, en forma de tuplas de base de datos o como objetos de memoria principal relacionados con la sesión del usuario, en RIAs el contenido también puede residir en el cliente, como objetos de memoria principal con el mismo nivel de visibilidad y la duración de la aplicación cliente, o incluso, en algunas tecnologías, como persistencia de objetos de lado del cliente. Además, en RIAs son posibles patrones de comunicación más potentes, como el empuje de mensajes de servidor a cliente y el procesamiento de eventos asíncronos. WebML ha sido ampliado usado con el objetivo de reducir la brecha entre las metodologías de desarrollo Web y el paradigma RIA, aprovechando las características comunes de RIAs y aplicaciones web tradicionales [39].

El proceso de diseño se amplía definiendo la asignación al lado del cliente o del servidor de elementos de datos (entidades y relaciones) y componentes de hipertexto (páginas, contenido y unidades de operación) y estableciendo los patrones de comunicación cliente-servidor notificación de eventos relevantes, filtrado de destinatarios y procesamiento de eventos síncronos / asincrónicos).

En el modelo de contenido, los conceptos se caracterizan por dos dimensiones diferentes: su ubicación, que puede ser el servidor o el cliente, y su duración, que puede ser persistente o temporal.

Del mismo modo, la noción de página en WebML se ha ampliado, agregando páginas de cliente, que incorporan contenido o lógicas gestionadas por el cliente. En estos casos su contenido se puede calcular en el lado del servidor o del cliente, mientras que la presentación, representación y el manejo de eventos ocurren exclusivamente en el lado del cliente.

3.3.6. IFML – Lenguaje de Modelados de Flujos de Interacción

El Lenguaje de Modelado de Flujos de Interacción o IFML[19] [40] se ha separado de la experiencia WebML, y tiene como objetivo generalizar a cualquier tipo de plataforma de software.

IFML ha sido diseñado para expresar el contenido, la interacción del usuario y el comportamiento de control del front-end de las aplicaciones pertenecientes a los siguientes dominios:

- Aplicaciones web tradicionales basadas en HTML y HTTP
- Rich Internet Applications (RIAs), tal como está soportado por el próximo estándar HTML 5.
- Aplicaciones móviles.
- Aplicaciones cliente-servidor.
- Aplicaciones de escritorio.
- Interfaces humanas integradas para aplicaciones de control. Aplicaciones multicanal y contextual.

Cabe señalar de nuevo que IFML no cubre el modelado de los problemas de presentación (por ejemplo, diseño, estilo, etc.) de una aplicación front-end y no se ocupa de la especificación de gráficos bidimensionales y tridimensionales basados en computadora, videojuegos y otras aplicaciones altamente interactivas. Está dirigido principalmente a aplicaciones orientadas a los negocios y que requieren mucha información [24].

Las notaciones y los conceptos no varían mucho con respecto a los de WebML, excepto por alguna terminología. Las dos grandes evoluciones con respecto a WebML son:

- La definición del concepto de evento como usuario de primera clase de la norma, con el fin de cubrir la gestión de un amplio espectro de tipos de eventos, abarcando eventos de usuario y eventos o procesos automáticos también.

[19] IFML. Interaction Flow Modeling Language

- La eliminación de las cadenas de los componentes de la lógica de negocios, para hacer que el lenguaje se centre más precisamente en el diseño de front-end (y delegar el modelado de orquestaciones a otros modelos, por ejemplo, secuencia UML o diagramas de actividad).

IFML ha sido adoptado como estándar por la OMG en versión Beta en marzo de 2013 y ha sido finalizado en la versión IFML 1.0 en marzo de 2014.

4. Comparativa entre OO-Method y WebRatio

4.1. Introducción

Para realizar la comparativa entre OO-Method y Webratio se analizarán como responden frente a los diferentes requerimientos implementados mediante el análisis de diferentes primitivas que se enumeran a continuación:

- **Modelo de Datos.** Se realiza la comparativa en base a cómo las herramientas manejan su modelado de datos, se toma en cuenta el parámetro que se utiliza para su diseño, así como también las estructuras y características que permiten, dando mayor importancia a las acciones de herencia, generalización y agregación. Es una visión general de cómo se permite diagramar la base misma de la programación enfocada en modelos. De igual manera se analiza la conexión con bases de datos físicas tanto en su facilidad como en versatilidad para adaptarse a diversas versiones de bases de datos que se encuentran en el mercado actual.
- **Clases.** En este apartado se buscará realizar un análisis acerca de la forma de implementación de las clases, como se permite graficar las mismas, así como sus propiedades de persistencia de la información, si estas son manejadas en la misma instancia o se requieren de otras normativas para especificarlo.
- **Atributos.** Hace referencia a la forma de incluir los atributos dentro del modelado de datos, su creación, modificación y eliminación. Se valora la usabilidad de las herramientas para su gestión.
- **Tipos de atributos.** Hace referencia a los tipos de atributos que manejan las herramientas clasificándolos en constantes, variables y derivados. Cabe mencionar que los atributos derivados serán estudiados en un apartado aparte de este mismo capítulo.
- **Tipos de datos de atributos.** Se evalúan los tipos de datos que soportan los atributos y si estos son específicos y soportados por las bases de datos actuales, permitiendo un mapeo transparente del modelo a la base y optimizando así la información obtenida de la misma.
- **Atributos derivados.** Apartado específico de un tipo de atributo, examina como manejan los atributos derivados ambas herramientas, si requieren de un proceso largo o es por otra parte algo intuitivo. Por otra parte, se evalúa la relación que existe entre los atributos derivados y su funcionalidad sobre la base de datos física.
- **Servicios.** Hace referencia a las operaciones básicas con las que puede interactuar el modelo, aquí se establecen los procesos de creación, modificación y eliminación de la información desde una perspectiva enfocada a modelos. Como se pueden diseñar estos procesos está establecido en este apartado, esto se realiza sobre las dos herramientas para su mejor análisis.
- **Transacciones.** En este apartado se busca realizar una comparativa para procesos más complejos que requieren de más de una operación o servicio para

poder ser ejecutada, se evalúa también las políticas que estos tengan para soportar errores y excepciones en tiempos de ejecución del aplicativo final.

- **Precondiciones.** Apartado en el que se evalúa el manejo de las condiciones que el sistema deberá cumplir para ejecutar una determinada acción, estos procesos son establecidos dentro del modelado de diferentes maneras por lo que la comparativa de las herramientas es esencial en este apartado.
- **Control de Agentes.** Hace referencia a la gestión del módulo de seguridad que tendrá el aplicativo, por medio de esta sección se permitirá generar o desarrollar los módulos que brindarán accesos al objeto usuario para que pueda activar servicios por medio de la visualización o no de sus atributos.

4.2. Modelo de Datos

El objetivo del modelado de datos es la especificación de los activos de información relevantes que constituyen el dominio de la aplicación de una manera formal pero comprensible y legible. La correcta interpretación de esta información permite generar un modelo de dominio, también llamado esquema conceptual por los diseñadores de bases de datos. Esto representa el conocimiento disponible sobre los conceptos relevantes, sus propiedades y relaciones, y, en el modelado orientado a objetos, las operaciones aplicables a ellos.

Por lo tanto, el modelado de dominio naturalmente interactúa con el modelado de la lógica de negocio y el front-end de la aplicación. El modelo de dominio producido también impulsa la implementación de las estructuras físicas para el almacenamiento, actualización y recuperación de datos.

El modelado del dominio es una de las disciplinas más consolidadas del manejo de la información. El modelo de dominio resultante puede considerarse como un modelo de contenido, que enfatiza la descripción de los activos de información utilizados por la aplicación.

Muchos lenguajes y directrices han sido propuestos y ahora están consolidados para el modelado de dominios. Por esta razón, no se busca proponer un nuevo lenguaje de modelado de dominio, sino que se explotan los diagramas de clases de UML. Por lo tanto, utilizando diagramas de clase se permite que una notación basada en modelos encaje tanto en el dominio como en el front-end de modelado. Como una alternativa más familiar a los diagramas de clase UML, los sistemas de información y los diseñadores de bases de datos pueden utilizar el modelo de entidad-relación, que se centra únicamente en las entidades, atributos y relaciones, pero no tiene en cuenta las operaciones apoyadas por los objetos.

Los ingredientes esenciales del modelo de dominio en UML son las clases, definidas como planos de resúmenes las propiedades comunes de objetos (también conocidos como instancias de clases), y asociaciones, que representan conexiones semánticas entre las clases. Las clases se caracterizan por atributos con sus correspondientes tipos de datos y por las operaciones aplicables a sus instancias. Las clases también pueden ser organizados en jerarquías y generalizaciones, que expresan la derivación de un concepto específico de uno más general e implican la herencia de propiedades y el comportamiento. Las asociaciones se caracterizan por restricciones de multiplicidad que

imponen restricciones al número de instancias de asociación en las que un objeto puede participar [41].

Clases. Una clase representa una descripción de características comunes de un grupo de objetos en un mundo real como una Persona, Producto, Vehículo, etc. Una clase es la unidad básica de modelado que debe tener como características un nombre único, atributos y relaciones.

4.2.1. Integranova

Dentro de Integranova se define a una clase con las propiedades de un nombre único, una función de identificación y un evento de creación, como propiedades adicionales tenemos un alias que es el nombre usado para mostrar al cliente final de la aplicación, comentarios que es información relevante al analista y un mensaje de ayuda que representaría la información interesante al usuario de la aplicación.

Finalmente, la mayoría de elementos que se encuentran contenidos en una clase son los atributos, que incluyen la información acerca de la clase. Servicios, que contienen la funcionalidad de una clase, restricciones definidas en la clase y otros como unidades de interacción que son empleadas para mostrar información al usuario final. Los desencadenadores o Triggers son definidos en la actividad interna de una clase.

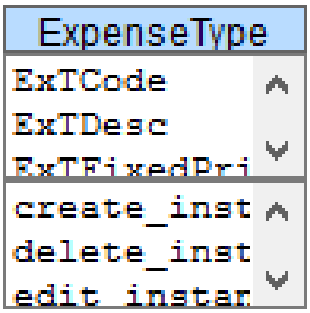


Figura 6. Ejemplo de clase en Integranova, donde se muestran atributos y funciones

Integranova maneja dos tipos de relaciones estructurales entre dos clases que son:

- **Agregación.** Es cuando en dos clases que se encuentran relacionadas, a cada clase se la puede acceder desde la otra en términos de un rol.
- **Herencia.** Es cuando una clase hija es una especificación de una clase padre. La clase hija cubrirá los atributos y servicios de la clase padre pero nunca se podrá realizar esta acción al contrario.

En lo que refiere a las relaciones de agregación se pueden definir las siguientes características:

- **Roles**, que pueden ser estáticos y dinámicos. Los roles son elementos empleados para acceder desde un lado de la relación hacia la otra parte. Los roles son usados en fórmulas para navegar a través de rutas y referir los atributos y servicios de una clase relacionada. En Integranova se emplean las características de Rol, que es el nombre usado para navegar entre los objetos relacionados vistos desde la

otra parte de origen o destino de la relación. El alias del rol es usado para ser mostrado en la interface de usuario de la aplicación.
Los roles estáticos son el lado de la relación cuando se establece que cuando una instancia es creada y no puede ser modificada posteriormente. Los roles dinámicos son el lado de la relación que puede ser modificado.
Las relaciones son dibujadas desde la clase derecha de la ventana hacia el lado derecho, aunque se permite el cambio de la posición por medio del botón Swap.

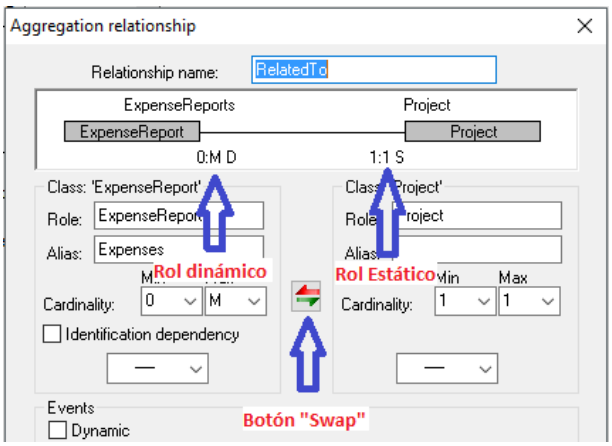


Figura 7. Ejemplo de gestión de roles en Integranova

- **Cardinalidad.** Es el número de entidades con la cual otra entidad puede asociar mediante una relación binaria; la cardinalidad puede ser:
    - **Mínima.** Es el número de instancias relacionadas necesarias, entre ellas están:
        - **0:** No son necesarias instancias relacionadas.
        - **1:** Necesita al menos una instancia relacionada.
        - **N:** Necesita "N" instancias relacionadas.
    - **Máxima.** Es el número de instancias relacionadas permitidas, entre ellas están:
        - **1:** Necesita al menos una instancia relacionada.
        - **N:** Varias instancias relacionadas son permitidas siempre y cuando no excedan el valor definido por N.
        - **M:** Todas las instancias relacionadas que se requieran

Hay que tener en consideración que existen combinaciones que el sistema considera que no tiene sentido como que al menos un rol tenga un valor mínimo de cero, estas validaciones protegen al sistema de inconsistencias sin embargo no resulta intuitivo generando un grave fallo de usabilidad en este sector del sistema.

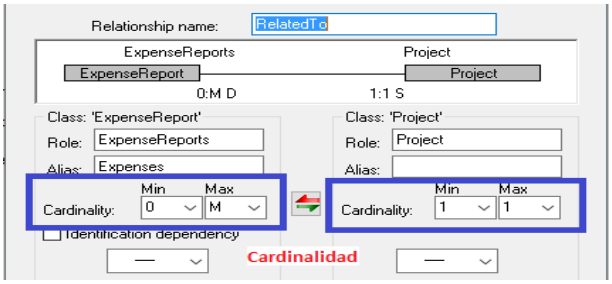


Figura 8. Ejemplo de configuración de cardinalidad en Integranova.

- **Propiedades estáticas y dinámicas.** Para determinar correctamente esta sección se definirá a las propiedades estáticas y dinámicas por separado.
    - **Propiedades estáticas.** Se dice que una relación es estática cuando uno de sus roles es estático. En una propiedad estática, la relación se encuentra establecida cuando una instancia es creada y no puede ser modificada posteriormente. Para eliminar las instancias de la clase independiente, estas no deberían tener ninguna instancia relacionada en la clase dependiente.

En Integranova, el rol que navega desde la parte izquierda hacia la parte derecha de la clase en la ventana puede contener la propiedad estática, la otra dirección pudiera ser dinámica.

Si una relación es estática, entonces debe seguir las siguientes normativas.

1. La relación debe ser establecida en la dirección planteada en tiempo de creación.
2. La eliminación requiere la supresión de todos los objetos relacionados en la otra parte.
3. Una vez establecida la relación, el objeto independiente no puede ser cambiado y la relación es estática.

Para poder definir una relación estática en Integranova se debe tener en consideración que, dentro de las propiedades, la característica de Dynamic no debería estar seleccionada, como se muestra en la figura 9:

Figura 9. Demostración de creación de una propiedad estática en Integranova

- **Propiedades dinámicas.** En Integranova, una relación es dinámica cuando ambos roles son dinámicos. Las relaciones dinámicas emplean eventos compartidos para modificar la relación entre dos clases en cualquier instante de tiempo.
  Para definir una relación dinámica en Integranova, se debe seleccionar la opción de Dynamic. Al seleccionar la opción de Dynamic los eventos compartidos que podemos emplear aparecerán como se ejemplifica en la Figura 10.

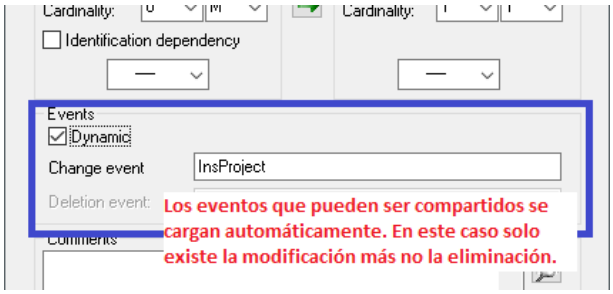


Figura 10. Ejemplo de creación de una propiedad dinámica en Integranova

- **Eventos compartidos**, los cuales están relacionados a la relación dinámica. Los eventos compartidos son los servicios que afectan a más de un objeto simultáneamente. Estos eventos permiten al analista establecer o eliminar relaciones entre objetos en cualquier momento. Los mismos son automáticamente añadidos a las dos clases involucradas en la relación. Estos dependen de una cardinalidad mínima y se podrían mostrar uno o dos eventos compartidos. Esto se puede observar de una mejor manera en la Figura 11:

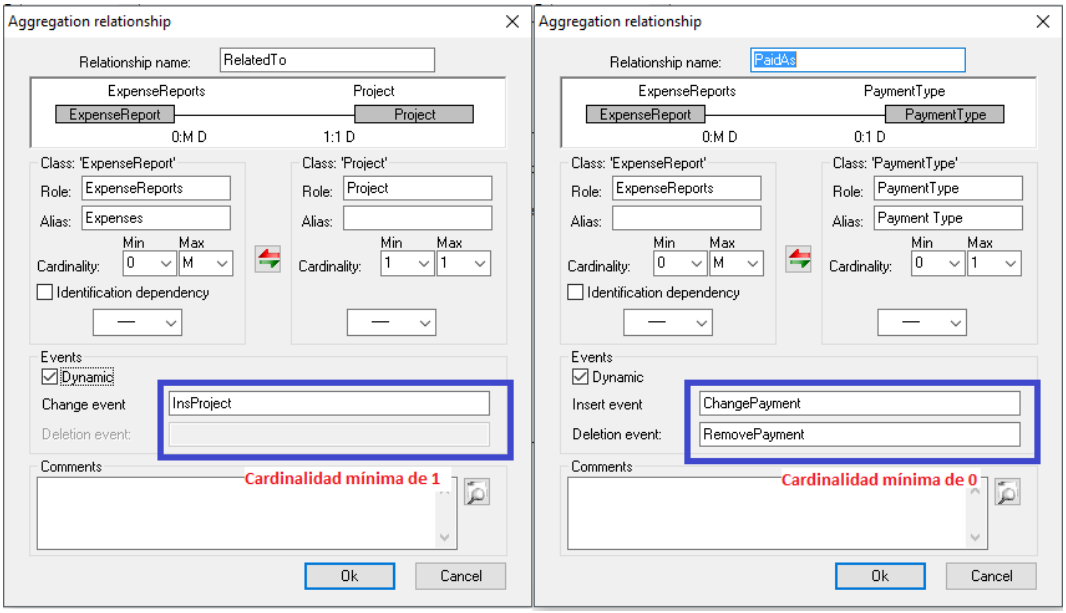


Figura 11. Ejemplo de eventos compartidos en Integranova con dos tipos de cardinalidad en Integranova

- **Identificación de dependencia.** Significa que la función para identificar el objeto estará compuesta por una función de identificación de la clase relacionada más los atributos propios de identificación. Para realizar una identificación de dependencia es necesario que la relación deba ser estática y la cardinalidad en el rol estático sea de 1 a 1. En la Figura 12 se detalla el proceso de creación:

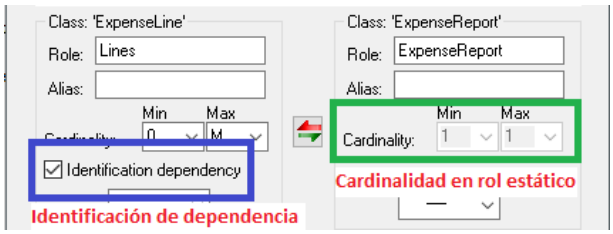


Figura 12. Ejemplo de Identificación de dependencia en Integranova

- **Representación de líneas.** La representación de líneas no tiene ninguna semántica asociada, sin embargo, ayuda a leer de una forma más clara los modelos expresados en los gráficos. Los diferentes tipos de representación son:
  - **Línea.** Representa asociación.
  - **Rombo blanco.** Representa agregación.
  - **Rombo negro.** Representa composición.

  Estas representaciones pueden apreciarse de una mejor manera en la siguiente gráfica:

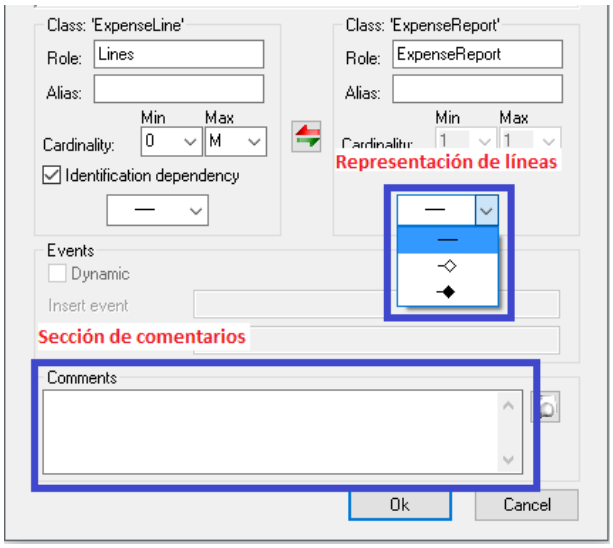


Figura 13. Ejemplo de representación de Líneas y Comentarios en Integranova

- **Comentarios.** Los comentarios son información introducida para una perspectiva de un analista para así poder ayudar a entender mejor el modelo por un nuevo desarrollador. Esta información puede ser usada para la documentación final de la aplicación.

En cuanto refiere los tipos de relaciones tenemos también el apartado de herencia, estas son relaciones que nos permiten "heredar" todas las características (atributos, derivaciones, servicios, precondiciones y valoraciones) de una clase padre hacia una clase hijo, pero no en viceversa. Dentro de las propiedades de la herencia en Integranova tenemos que algunos elementos pueden ser redefinidos por medio de Redefinition. La Cancelación está prohibida, esto quiere decir que no es posible dejar de heredar un determinado elemento, por ejemplo, si la clase padre tiene 4 atributos, no podemos eliminar alguno de ellos en la clase hija, es posible agregar nuevos atributos, pero no eliminarlos.

El tipo de enlace definido en Integranova es llamado Especialización Temporal por Evento. Si analizamos esta expresión obtendremos que por Temporal se entiende que un objeto puede ser especializado después que este haya sido creado. El objeto también puede perder la especialización, pero puede seguir siendo un objeto pariente. Por evento entendemos que el objeto es especializado después de ejecutar un evento y pierde su especialización cuando otro evento "liberador" es ejecutado.

La especialización temporal requiere una compatibilidad de firma, lo cual significa que el comportamiento del objeto especializado puede ser diferente del de la clase padre. La especialización múltiple está permitida, es decir que una clase padre puede tener más de

una clase hijo sin embargo la generalización múltiple no, por lo que una clase hijo puede tener únicamente solo una clase padre. La clase hija puede definir su propia función de identificación. Otra característica de la especialización es que una clase padre puede ser a su vez hija de otra clase sin embargo los ciclos no son permitidos en una jerarquía de especialización.

Otras características de la especialización temporal son:

- **Evento portador.** Es el evento o eventos definidos en la clase padre, esta es una característica obligatoria.
- **Evento liberador.** Es el evento o eventos definidos en la clase padre o hija, pero esta es opcional.

Los eventos de Nuevo y Eliminar no se encuentran permitidos en la clase hija dado que los mismos se encuentran reservados para la clase padre.

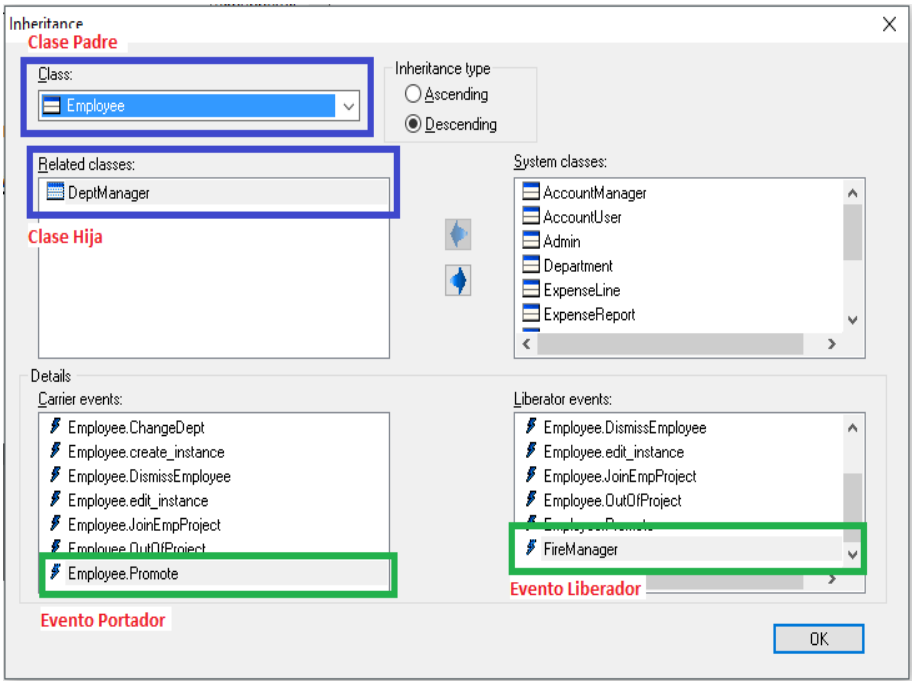


Figura 14. Ejemplo de especialización temporal por eventos en Integranova

4.2.2. WebRatio

Según la ideología de WebRatio para que una aplicación web pueda administrar y mostrar información al usuario, debe tener una base adecuada y coherente, que represente los conceptos clave de la aplicación Web, esta base se encuentra plasmada y diseñada en el modelo de datos.

El modelo de datos describe la organización de los mismos en las aplicaciones WebRatio y no es más que un contenedor de elementos de contenido a los que los diferentes componentes pueden tener acceso para recuperar información, procesarla y mostrarla al usuario. El modelo de datos también almacena la información capturada del usuario.

4.2.2.1. Modelo Entidad-Relación

Se supone que el modelo de datos está representado en UML y, por lo tanto, consiste en un conjunto de elementos de modelo UML, pero también es compatible con el modelo Entidad-Relación. WebRatio utiliza tanto el diagrama Entidad-Relación como la representación UML de los datos.

Un Modelo de Entidad-Relación es una representación abstracta y conceptual de los datos. Es un método de modelado de base de datos, utilizado para producir un tipo de esquema conceptual o modelo de dominio semántica de un sistema, a menudo una base de datos relacional, y sus requisitos de una manera descendente [41].

El esquema Entidad-Relación se basa en 3 conceptos principales:

- **Entidad.** Representa una descripción de las características comunes de un conjunto de objetos del mundo real. Una entidad tiene una población, que es el conjunto de objetos que son descritos por la Entidad. Estos objetos también se denominan instancias de entidad. Un ejemplo de entidad se define a continuación:

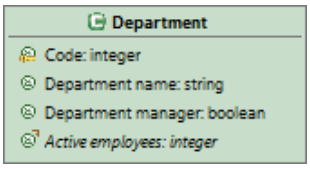


Figura 15. Ejemplo de Entidad en WebRatio

  Como se puede apreciar en la gráfica anterior, la entidad viene representada por un nombre y por atributos (de los cuales se pueden observar el atributo con identificador principal o *key* y un atributo derivado que se evaluará posteriormente).

  En las propiedades de la entidad tenemos las siguientes características [42]:

  - **Identificador.** Es un identificador único que se le asigna a cada entidad al momento de su creación, este es un valor que no puede ser ingresado o modificado manualmente por el analista, sino que se lo incorpora automáticamente.
  - **Nombre.** Es el nombre que se le asigna a la entidad, WebRatio permite la inclusión de caracteres especiales para generar nombres y le asigna automáticamente nombres nemotécnicos al momento de ser mapeado a una base de datos física. Este nombre es utilizado por el sistema como un alias y es el que se verá en la intefaz de usuario, sin embargo, este alias puede ser modificado en la capa de diseño si así el usuario lo considera conveniente o requiere referirse a la misma instancia de diferentes maneras. Por ejemplo, la entidad Persona pudiera ser llamada como Cliente o Empleado dependiendo de la Especialización que se use.
  - **Super Entidad.** Es un espacio donde se cargará el nombre de la Entidad Padre en caso de ser esta una entidad hija por medio de una especialización.
  - **Duración.** La duración hace referencia a la persistencia de los datos, tiene como opciones la persistente, que implica que los datos serán

almacenados físicamente en la base de datos. También tiene la opción volátil en tiempo de sesión que nos permite almacenar datos de una forma virtual y los mismos serán eliminados una vez que se finalice una sesión; también están los volátiles en tiempo de aplicación que serán almacenados hasta que la aplicación sea cerrada.

- **Derivación.** La derivación hace referencia a cuando empleamos datos que se encuentran en otras tablas, esta derivación se lo realizaría de una forma manual para interactuar con tablas adicionales y que contengan información relevante para el sistema y no forme parte del diseño conceptual de datos.
- **Orden de los atributos.** Nos permite ordenar los atributos de la manera que deseamos que los mismos sean mapeados a la base de datos física.

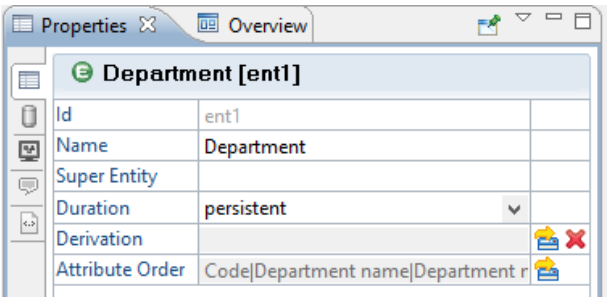


Figura 16. Ejemplo de propiedades de una entidad en WebRatio

- **Atributo.** Representa una propiedad de objetos del mundo real que es relevante para propósitos de la aplicación web. Los atributos están asociados con el concepto de Entidad, con el significado de que todas las instancias de Entidad se caracterizan por el mismo conjunto de Atributos. Los atributos serán evaluados con mayor profundidad en la sección de Atributos.
- **Relación.** Una Relación representa una conexión entre Entidades. El significado de la asociación se transmite por el nombre de la relación y la cardinalidad. Cada relación tiene dos roles, que son las formas en que una relación puede ser leída.

  La cardinalidad especifica cuántas instancias de entidad pueden participar en una relación. Hay tres tipos de cardinalidad:
  - **Relaciones.** Ambos roles de relación tienen máxima cardinalidad 1.
  - **Relaciones 1-N**. Un rol de relación tiene máxima cardinalidad 1 y el otro tiene cardinalidad máxima N.
  - **Relaciones N-N**. Ambos roles de relación tienen máxima cardinalidad N. Cuando existen estos casos, el mapeo en la base de datos genera automáticamente la tabla auxiliar necesaria para poder interactuar con la misma relación.

Figura 17. Ejemplo de relaciones en WebRatio

Dentro de las propiedades de una relación tenemos las siguientes:

- **Identificador.** Es el nombre nemotécnico que WebRatio asigna a todos sus componentes.
- **Nombre.** Es el nombre que se le asigna a la relación.
- **Entidad origen.** Es el nombre de la entidad de la cual va a partir la relación.
- **Nombre del rol directo en inverso.** Es el nombre con el que se le asigna al rol directo e inverso, automáticamente el sistema le asigna un nombre con la unión de las dos entidades, sin embargo, es una buena práctica asignar nosotros mismos el nombre de nuestras relaciones para una mejor comprensión del modelo.
- **Cardinalidad máxima directa e inversa.** Es la cardinalidad que tendrá el rol directo y el inverso, los valores que puede tener es 1 o N.
- **Derivación directa e inversa.** Es una sentencia de derivación que define a los miembros del rol de la relación. Tiene provisto de un agente guía que nos da asistencia en la edición de la sentencia de derivación.
- **Eliminación directa e inversa en cascada.** Al seleccionar esta opción permitimos que, de darse una eliminación, todas las instancias relacionadas sean eliminadas de una forma directa o inversa.
- **Entidad objetivo.** Es el nombre de la entidad que recibe la relación.

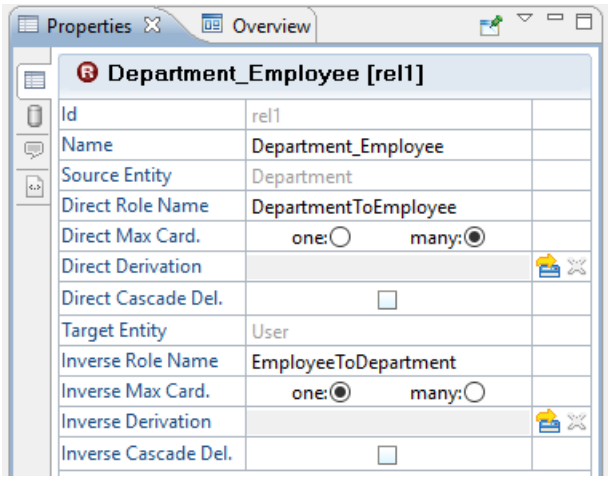


Figura 18. Propiedades de una relación en WebRatio

No es posible conectar dos Entidades con una duración diferente (por ejemplo, una persistente y la única volátil) [41].

Herencia

WebRatio permite organizar las clases de una forma jerárquica para resaltar y reutilizar sus características comunes. Una generalización conecta a una superclase con una o más subclases, las cuales representan la especialización de la súper clase. Se puede observar que mantiene las mismas propiedades de redefinición y cancelación que maneja Integranova, así como mantener una jerarquía en la que una subclase puede convertirse en una superclase de otras entidades hijas [42].

Cada subclase hereda las características (atributos, operaciones y asociaciones) definidas en la superclase y puede añadir localmente características definidas. Cuando el modelado de datos tiene el propósito de especificar las clases persistentes que forman el nivel de datos de una aplicación, es habitual asumir algunas hipótesis restrictivas que simplifican

la forma de jerarquías de generalización y las hacen más fáciles de implementar con la tecnología de bases de datos convencionales, entre ellas tenemos [41]:

- Cada clase se define como la especialización de como máximo una superclase. En términos técnicos, se evita "herencia múltiple".
- Cada instancia de una superclase está especializada exclusivamente en una subclase.
- Cada clase aparece en como máximo una jerarquía de generalización.

4.3. Atributos

Los atributos son las propiedades de los objetos que son relevantes para los propósitos de la aplicación [41].

Un atributo es un elemento del modelo de datos que contiene un valor determinado con un tipo de valor respectivo al cual se encuentra asociado. Sumado a ello tenemos el concepto del estado del objeto que corresponde al grupo de valores de los atributos en un momento en concreto.

Dentro de las instancias de una clase se permite tener valores nulos para uno o varios atributos, sin embargo, un valor nulo puede representar diferentes situaciones de modelado y generar ambigüedades en la interpretación de las propiedades de una instancia [41], por ejemplo:

- Un valor nulo podría denotar que la certeza de un atributo no puede aplicarse a una instancia de clase específica. Por ejemplo, si hablamos de un número telefónico local de un usuario, este puede tenerlo como no.
- Un valor nulo podría también denotar que el valor de un atributo puede ser simplemente desconocido y es por esta razón que no es colocado como la edad o el estado civil de un usuario.

Primary Key. Una primary key o llave primaria es un atributo que puede ser usado para identificar las instancias de la clase de una forma única. Un ejemplo de llave primaria pudiera ser el número del seguro social o la placa de un vehículo.

4.3.1. Integranova

Para realizar el estudio de cómo se manejan los atributos en la herramienta de Integranova, vamos a partir de la siguiente figura:

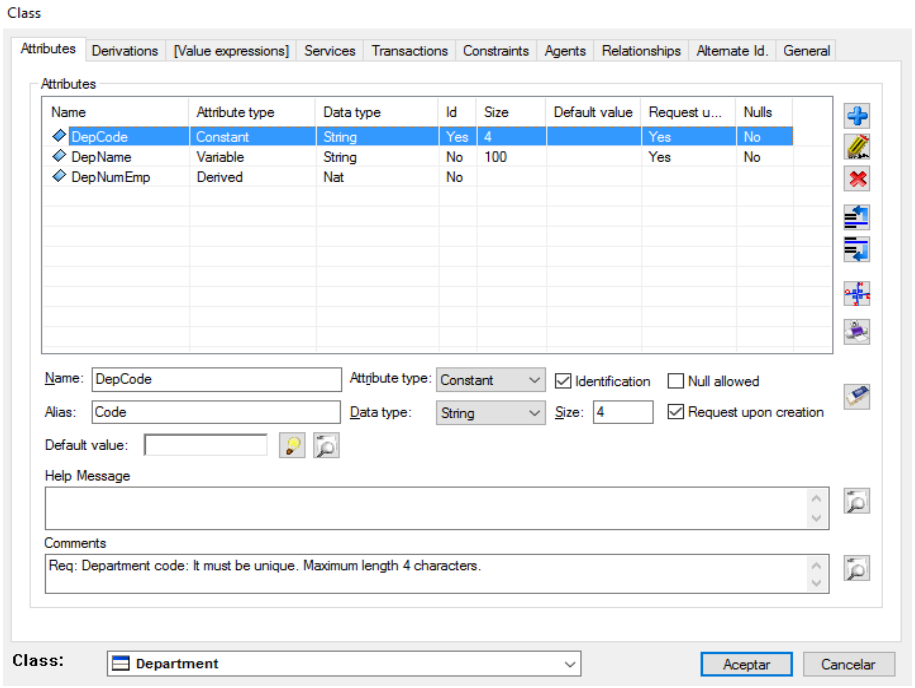


Figura 19. Modelo de Atributos en Integranova

Como se puede apreciar, los atributos aparte de su nombre pueden tener un alias, comentarios y un mensaje de ayuda. Para la creación de un nuevo atributo nos dirigiremos al botón de + que se encuentra en la esquina superior derecha de color azul y eso nos permitirá crear un nuevo atributo al cuál le daremos un nombre, un tipo, un alias, el tamaño del atributo si así correspondiera, de igual manera validaremos si permite valores nulos y si requiere el proceso de creación para validarlo posteriormente como un proceso de mantenimiento. Para determinar si el atributo es una llave primaria lo gestionamos directamente en las características señalado la propiedad Identification como activa, finalmente tenemos también la opción de crear un valor por defecto en la opción Default Value. Integranova maneja estos procesos a nivel de creación de atributos de variable, WebRatio lo hace a nivel de modelado por medio de IFML permitiendo un mayor control sobre el despliegue de mantenimientos.

4.3.2. WebRatio

De manera análoga al análisis de la herramienta Integranova, para el estudio de cómo se manejan los atributos en WebRatio partimos de la siguiente figura:

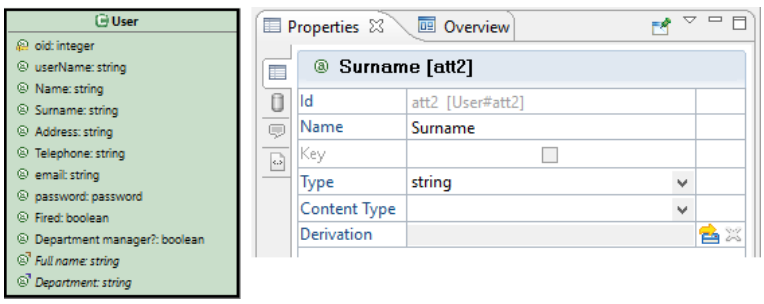


Figura 20. Modelo de Atributos en WebRatio

Se pueden apreciar dos gráficas en la Figura 20, en la parte izquierda tenemos la vista de una clase con los atributos que lo componen, mientras que en la parte izquierda tenemos la pestaña de propiedades donde definimos el nombre, para ello no importa el uso de caracteres nemotécnicos puesto que WebRatio acepta caracteres especiales en la definición de los nombres, los mismos pueden ser utilizados como alias dentro del aplicativo final si así el programador lo decidiera, caso contrario, podrá modificar el alias libremente en el diseño de la interfaz más adelante. También tiene la opción de validar si el atributo es una llave primaria, el tipo de archivo, el tipo de contenido (texto plano o enriquecido por medio de HTML) y si es un atributo derivado. En contraposición a lo utilizado en Integranova, WebRatio maneja la opción de los valores por defecto dentro del apartado de atributos derivados, generando un atributo constante que irá por defecto, también se tiene la opción de enviar los valores por defecto directamente en el modelado IFML.

4.4. Tipos de Atributos

4.4.1. Integranova

Los tipos de atributos que soporta Integranova son:

- **Constantes.** Son valores que no pueden ser cambiados una vez que han sido asignados en el momento de la creación del objeto.
- **Variables.** Son aquellos tipos de datos que pueden ser controlados e ingresados por el usuario de una forma manual. Pueden cambiar en cualquier momento dependiendo de la ejecución de los eventos que modifican este valor.
- **Derivados.** Son aquellos que son calculados de los valores de otros atributos que se encuentran en la base de datos y se tenga una relación directa con los mismos.

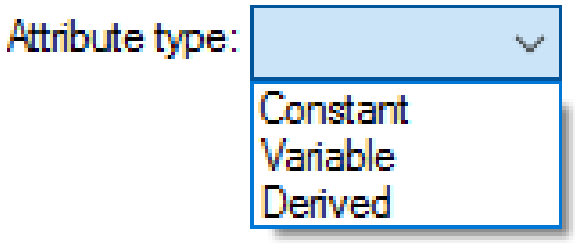


Figura 21. Tipos de atributos en Integranova

En la gráfica anterior se puede observar cómo se puede seleccionar el tipo de dato del atributo por medio de un combo desplegable que nos permite definir las tres posibles opciones que se pueden escoger.

4.4.2. WebRatio

- **Constantes.** WebRatio maneja el uso de atributos constantes de una manera implícita, es decir, las claves primarias las maneja por medio de valores enteros

auto incrementales, en caso de querer definir un algoritmo para generación de código propio del sistema este se lo pudiera modelar y almacenar directamente sobre la clave primaria sin ningún inconveniente.

- **Variables**. WebRatio maneja los atributos variables de manera análoga a Integranova.
- **Derivados**. WebRatio maneja los atributos derivados de varias maneras dependiendo el grado de derivación que el atributo posea, este apartado se lo explicará más ampliamente en el apartado de Atributos Derivados.

4.5. Tipos de datos de atributo

4.5.1. Integranova

Los tipos de datos de atributos que maneja Integranova se maneja a manera de un combo que permite definir el tipo de dato que va a manejar el atributo al momento de su creación. Para ejemplificarlo de una manera más gráfica se podrá observar la gráfica a continuación:

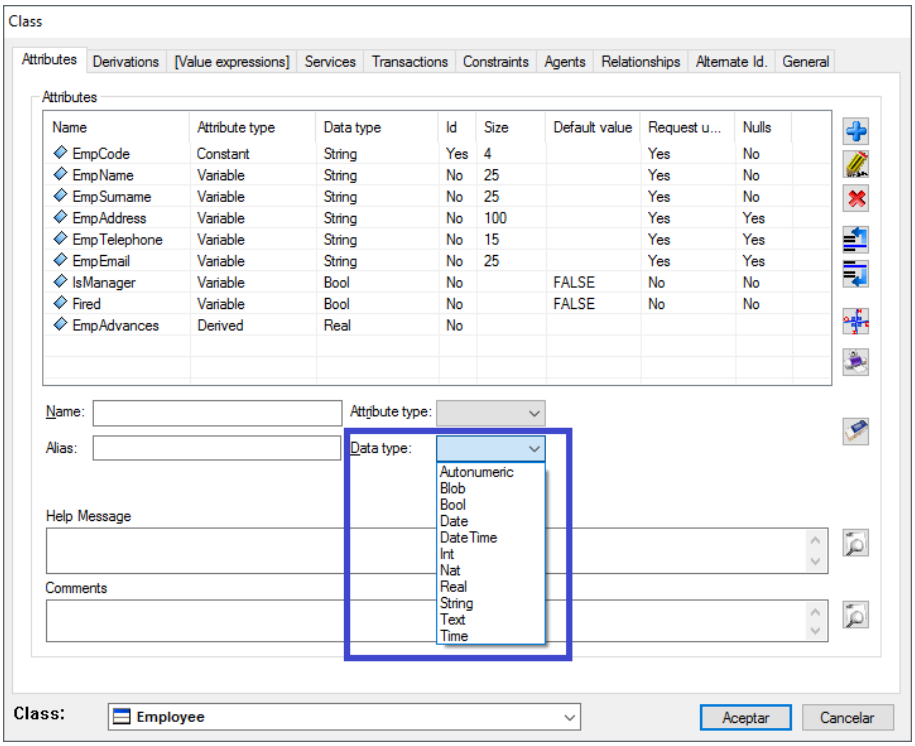


Figura 22. Ejemplo de Tipos de datos que maneja Integranova

Como se puede apreciar en la ilustración anterior, los tipos de datos que soporta Integranova son: auto numéricos, blob, booleanos, fecha, fecha y hora, enteros, natural, reales, cadenas, texto y hora.

4.5.2. WebRatio

Los tipos de atributos que maneja WebRatio se asignan por medio de su apartado de propiedades que se enseñó anteriormente y los tipos de datos que soporta son los siguientes:

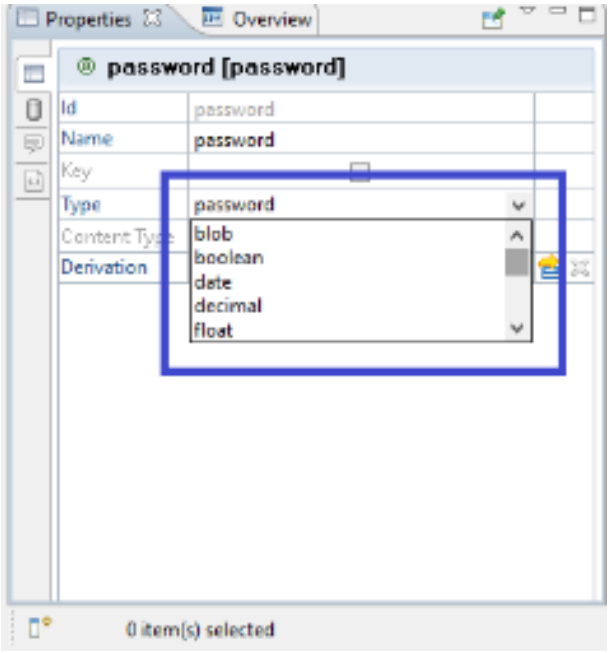


Figura 23. Ejemplo de tipos de atributos de WebRatio

Los tipos de datos que maneja WebRatio son: Blob, booleanos, fecha, decimales, flotantes, enteros, contraseña, cadena, texto, hora, captura de hora y url. Como se puede apreciar, maneja un rango más amplio de tipos de datos que Integranova.

4.6. Atributos Derivados

4.6.1. Integranova

En Integranova se manejan los atributos derivados para obtener datos que se encuentran almacenados en otros lugares del modelado, también son de gran utilidad para el establecimiento de fórmulas puesto que permiten obtener valores para nuevos atributos que no se encuentren almacenados en la Base de Datos, sino que son calculados en tiempo de ejecución. Hay tener mucho cuidado puesto que estos atributos no pueden ser nulos. Para manejar estas fórmulas, Integranova permite realizarla en una pestaña destinada para esta especificación.

Dentro de esta pestaña es posible definir varias derivaciones para el mismo atributo derivado, siempre y cuando las condiciones sean diferentes. El orden es muy importante dado que las condiciones pueden ser evaluadas acorde estas hayan sido definidas. Es requerida, al menos, una condición de derivación vacía al final.

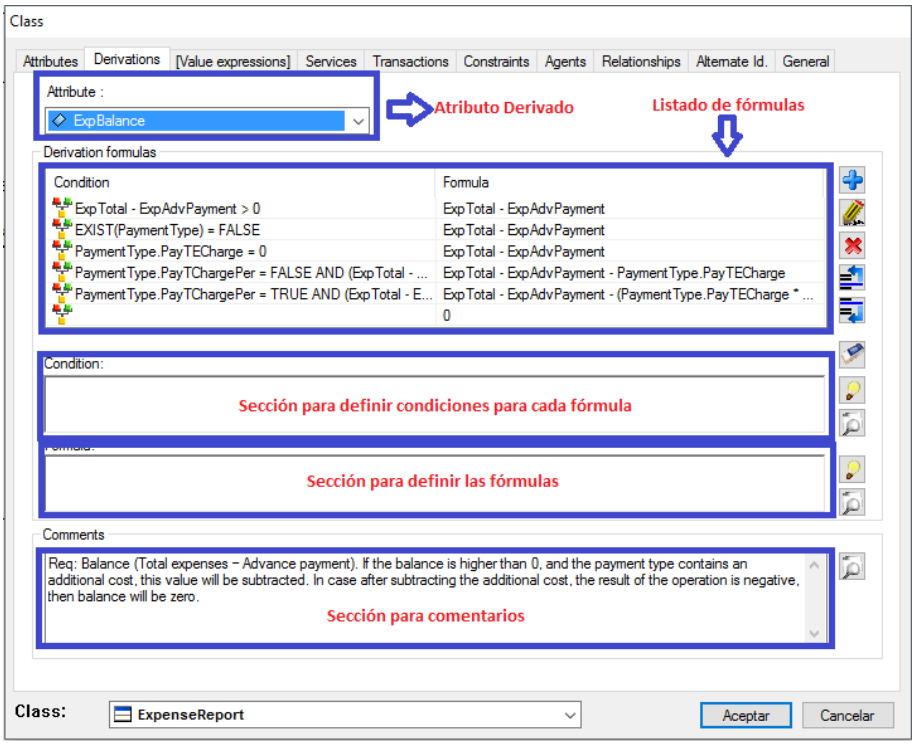


Figura 24. Ejemplo de configuración de atributos derivados en Integranova

Como se puede apreciar en la gráfica anterior, en la primera parte tenemos el listado de las fórmulas de derivación del atributo marcado en la parte superior. Posteriormente se tiene un apartado donde se encuentran las condiciones para que se pueda ejecutar la fórmula planteada en el apartado inferior. Finalmente tenemos una sección de comentarios que le ayudarán al analizador a entender mejor el proceso realizado por cada derivación.

4.6.2. WebRatio

En WebRatio se pueden definir atributos por medio de la opción Derivation que nos dirigirá a una pantalla auxiliar con las siguientes opciones:

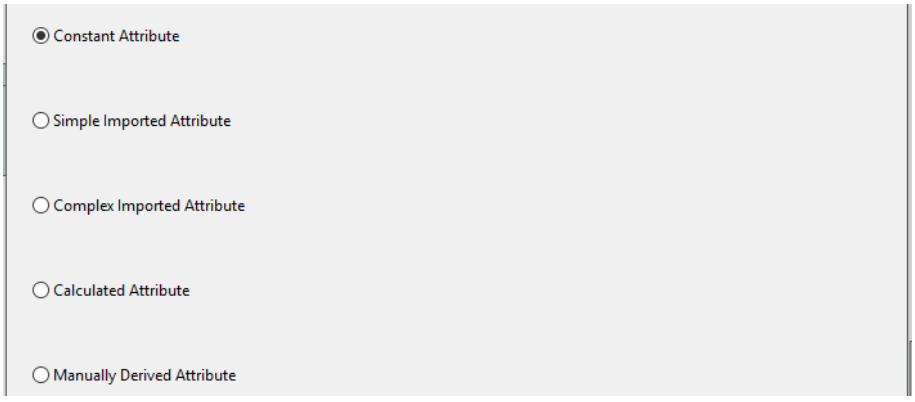


Figura 25. Gestión de atributos derivados en WebRatio

Dentro de las opciones presentadas en la gráfica anterior podemos apreciar que tenemos las opciones de:

- **Atributos constantes:** Nos permitirán dar un valor constante al atributo, es una forma de expresar un valor por defecto que tendrá nuestro atributo.
- **Atributo Importado Simple:** Nos permitirá obtener el resultado de un valor que se encuentre almacenado físicamente en otra tabla al cual se le tenga una relación jerárquica directa.

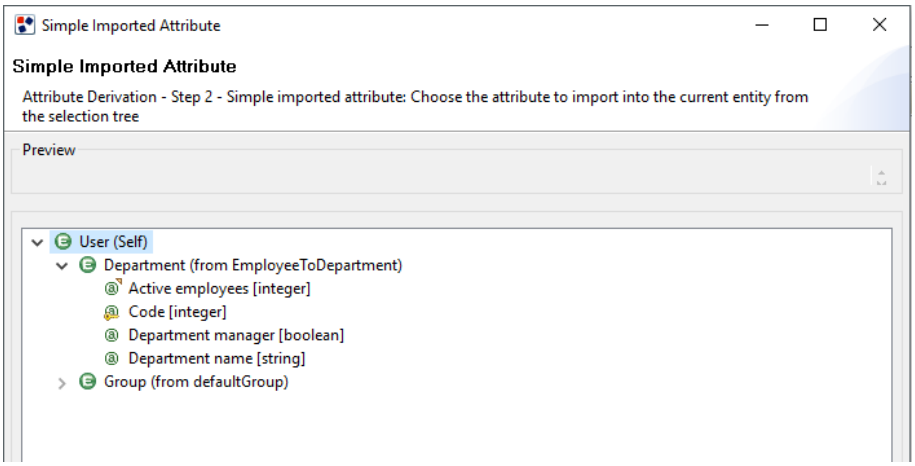


Figura 26. . Ejemplo de atributo importado simple en WebRatio

Como se puede apreciar en la ilustración anterior, se cargarán automáticamente los atributos de las clases con las que se tenga una relación jerárquica, esto no replicará la información física, sino que generará una consulta virtual que permita acceder a la información persistente de la base de datos.

- **Atributo Importado Complejo:** En algunos casos, es posible que se desee consultar información específica sólo cuando se compruebe una condición. En estos casos se emplea el atributo importado complejo puesto que a diferencia del simple nos permite incorporar condiciones al atributo importado. Por ejemplo, si deseamos ver los usuarios que pertenecen a un Departamento que ganen más de €300 pero menos de €1500 tendremos ya una condición que sería imposible de manejar por una importación simple que es demasiado genérica. Dado que este tipo de importación incluye una consulta de derivación, cuando se la cree se tendrá que sincronizarse con la base de datos para que se cree la vista correspondiente a la mencionada consulta. A continuación, la gráfica que explica el proceso anteriormente explicado [43].

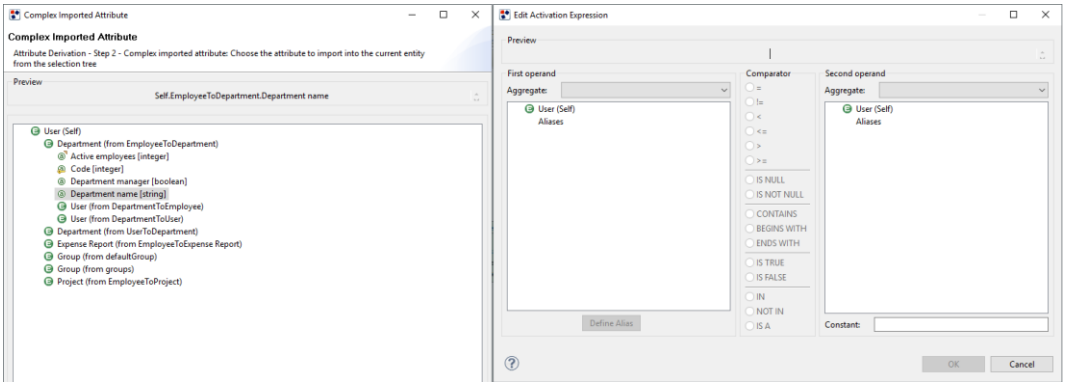


Figura 27. Ejemplo de un atributo importado complejo en WebRatio

- **Atributo Calculado.** Este atributo nos permite realizar operaciones que pueden como no incluir condiciones, estos atributos son empleados en caso de querer conocer cuántos usuarios que hayan realizado operaciones en los últimos tres meses se encuentran registrados en el sistema. Como se puede apreciar, la consulta explicada conlleva una operación que sería el conteo de los usuarios que pertenecen a un departamento determinado, así como la condición que es que hayan realizado operaciones en los últimos tres meses. Dentro de las operaciones que podemos realizar con los atributos en este apartado tenemos el de conteo, suma, promedio, cálculo de valores mínimos y máximos, dependiendo del tipo de atributo seleccionado. Al igual que los atributos importados complejos, este tipo de atributo requiere una sincronización con la base de datos para poder generar la vista correspondiente que realice el proceso definido [43].

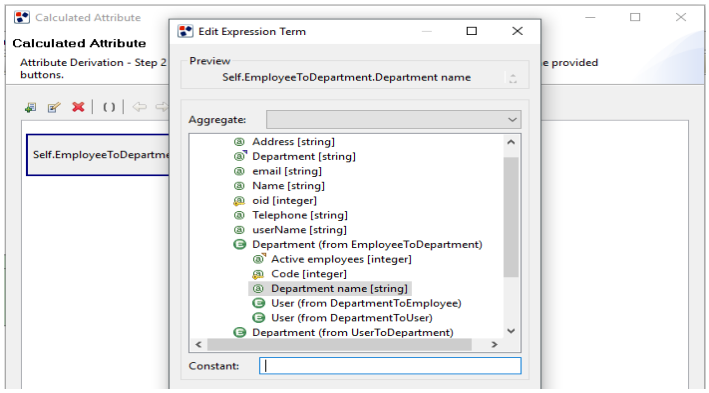


Figura 28. Ejemplo de Atributo Calculado en WebRatio

- **Atributo Derivado Manualmente.** Es una vista que realizamos manualmente sobre la base de datos a la que vamos a importar a nuestro modelado, esto nos permite realizar consultas mucho más definidas directamente en la capa de datos y traerlas a nuestro modelo para poder emplear los valores que se requieran. Esta utilidad es de gran valía en desarrollos sobre bases de datos definidas con anterioridad puesto que permite recuperar el trabajo realizado de una forma transparente y evitando volver a generar el modelo desde cero [43].

4.7. Servicios

Los servicios son componentes básicos que se encuentran relacionados al comportamiento de la clase.

4.7.1. Integranova

Antes de iniciar la explicación de los servicios en Integranova es importante considerar ciertos parámetros como son los argumentos, que son información proveída al servicio antes de la ejecución del servicio o la información resultante que se obtiene luego de una correcta ejecución del servicio.

En Integranova existen dos tipos de argumentos:

- **Argumentos de Entrada.** Datos requeridos como ingreso por el servicio. Los argumentos de entrada, al igual que los atributos, tienen ciertas propiedades que

maneja Integranova como el nombre, tipo de dato, si es un dato nulo, alias, valor por defecto, mensaje de ayuda y comentarios. Todos los servicios, con excepción de los de creación, poseen argumentos de entrada que representan a la instancia actual de la clase.

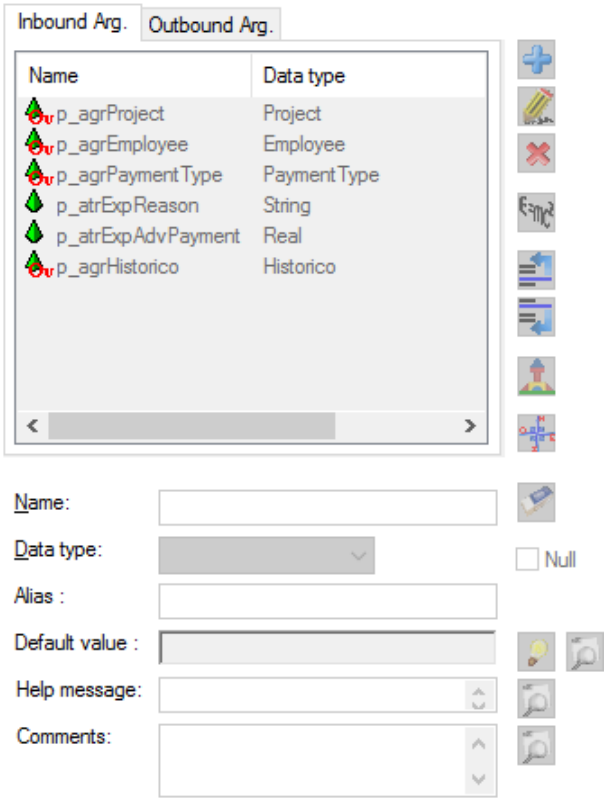


Figura 29. Ejemplo de Argumentos de Entrada en Integranova

- **Argumentos de Salida.** Respuesta del sistema después una correcta ejecución de un servicio. Un argumento de salida es una instancia que no es necesaria para la ejecución del servicio. Permite devolver un valor si el servicio ha sido ejecutado satisfactoriamente y su ejecución ha sido completada. En transacciones u operaciones, se las puede emplear como variables locales que permiten copiar un valor calculado de un servicio como la entrada de otro.
  Los argumentos de salida tienen a su vez propiedades que son: nombre, tipo de dato, verificación si son nulos, alias, valor por defecto, mensaje de ayuda y comentarios, características de las cuales hemos visto sus funcionalidades en apartados anteriores. La información a ser devuelta es asignada por una expresión de valor o la asignación en una fórmula de transacción o de operación.

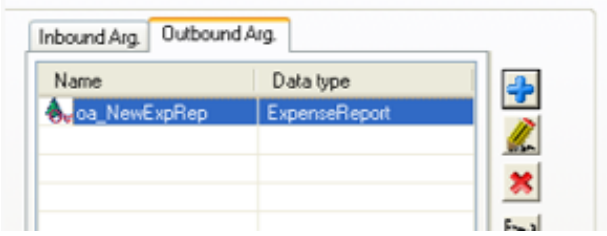


Figura 30. Ejemplo de Argumentos de Salida en Integranova

En Integranova los servicios tienen un nombre que es obligatorio y único, de igual manera en caso de una red de herencias. Las propiedades adicionales que un servicio tiene son: Alias, que es el nombre como el servicio se muestra en la aplicación final, comentarios para aclarar la información acerca del servicio al analista y un mensaje de ayuda que podrá visualizar el usuario final en la interfaz de usuario.

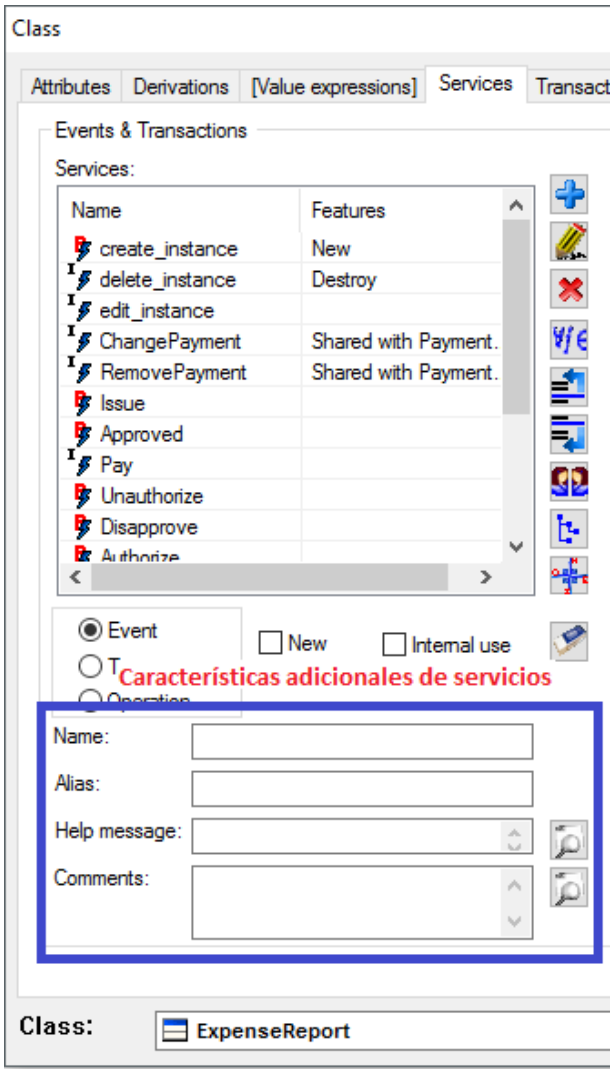


Figura 31. Ejemplo de Servicio en Integranova

Cuando se marca la opción de uso interno (*Internal use*), el servicio podrá ser empleado únicamente en transacciones o fórmulas de operación. Los servicios internos no tienen agentes de relaciones ni unidades de interacción de servicios; esto quiere decir que un servicio interno jamás se mostrará de una forma visual en una aplicación final.

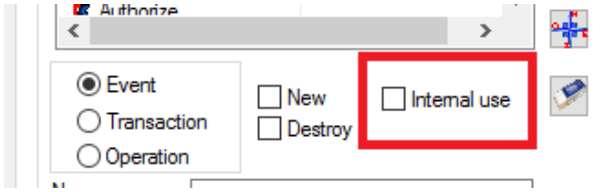


Figura 32. Opción para marcar a un servicio como interno en Integranova

Un servicio puede ser definido como una unidad de proceso, con esta premisa es posible decir que existen tres tipos de servicios que son:

- Eventos (atómicos)
- Transacciones (moleculares)
- Operaciones (moleculares)

Figura 33. Tipos de servicios en Integranova

Eventos. Los eventos son unidades de proceso atómicos que ocurren en un momento específico de tiempo, existen 4 tipos de eventos que son:

- **Creación:** Evento que crea un objeto.
- **Destrucción:** Evento que destruye un objeto
- **Propio:** Evento que afecta el estado de solo un objeto.
- **Compartido:** Evento que afecta el estado de dos o más objetos.

Los eventos están relacionados a las Valoraciones que definen como asignar valor a los atributos.

Figura 34. Tipos de eventos en Integranova

Evento de Creación. Un evento de creación está identificado por el checkbox de New como se puede apreciar en la ilustración anterior, cuando este está marcado. Los argumentos de entrada que son agregados automáticamente a la creación del evento son: la instancia de todas las clases relacionadas, si este rol es estático o la cardinalidad mínima es 1 y que todos los atributos se encuentren con la opción de Creación.

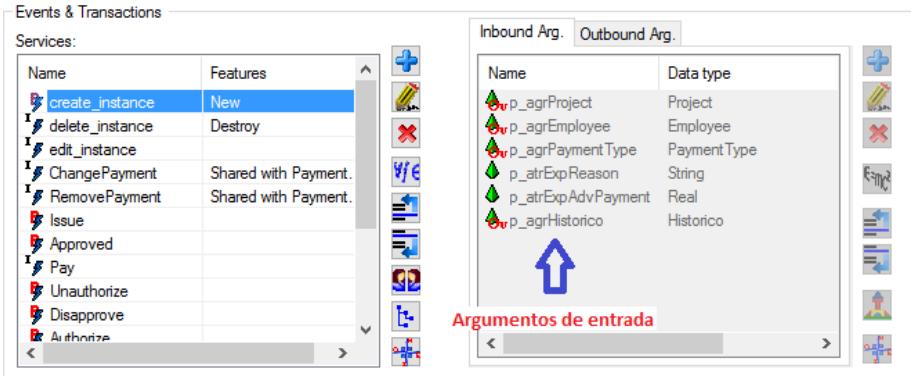


Figura 35. Argumentos de entrada de un Evento de Creación en Integranova

Evento de Destrucción. Un evento de destrucción es identificado por el checkbox de Destroy. Cuando este tipo de evento es marcado, un argumento de entrada es agregado automáticamente para destruir el evento que representa la instancia a ser eliminada.

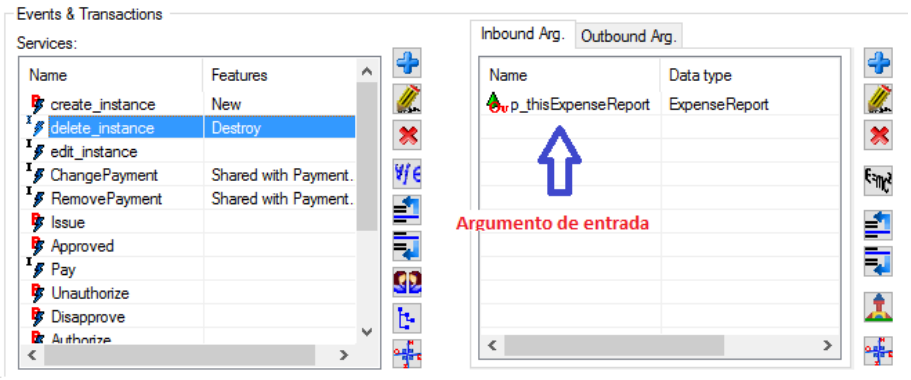


Figura 36. Argumentos de entrada de un Evento de Destrucción en Integranova

Evento de Edición. Un evento de edición es un evento creado automáticamente desde el asistente de creación extendida. Para utilizar este tipo de evento no hay que marcar ninguna de las opciones de New o Destroy. Para un evento de edición, se agrega automáticamente un argumento de entrada que la instancia que representa el objeto THIS y todos los atributos que fueron marcados como agregar un evento de edición (Add to Edit Event).

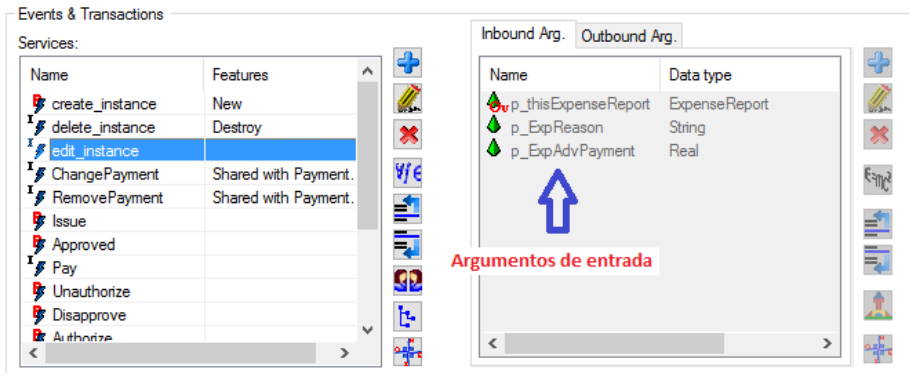


Figura 37. Argumentos de entrada de un Evento de Edición en Integranova

Eventos propios y compartidos. Los eventos propios son todos los eventos que afectan el estado de un solo objeto, por ejemplo, la creación y la destrucción son eventos propios. Los eventos compartidos son cualquier evento que afecta el estado de dos o más objetos. Los eventos crean o modifican una relación dinámica entre eventos compartidos, por esta razón, es posible definir eventos compartidos que solo modifican atributos y que no se encuentren enlazados con relaciones dinámicas.

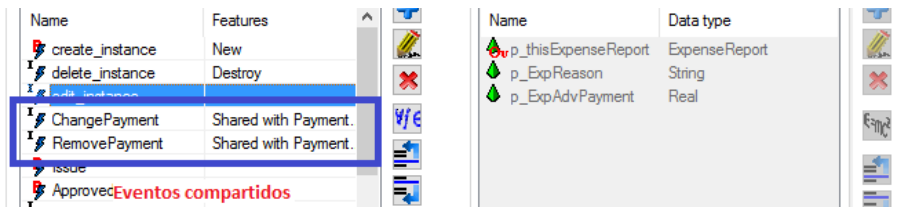


Figura 38. Ejemplo de eventos compartidos con relaciones dinámicas en Integranova

4.7.2. WebRatio

Los eventos son ocurrencias que pueden afectar el estado de la aplicación, y son un subtipo de elementos de interacción de flujo. Los eventos se clasifican en dos categorías principales: Eventos de captura (eventos que se capturan en la interfaz de usuario y que activan un cambio de interfaz posterior) y Eventos de envío (eventos generados por la interfaz de usuario).

Existen tres tipos de Eventos de Captura: Eventos de visualización de Elementos, resultante de una interacción del usuario (con subtipos específicos sobre eventos seleccionados y sobre eventos ejecutados), Eventos de Acción y Eventos del Sistema.

Los Eventos de Visualización de Elementos son pertenecen a los elementos de vista relacionados. Esto significa que los elementos de vista contienen eventos que permiten al usuario activar una interacción en la aplicación, por ejemplo, con el clic en un hipervínculo o en un botón.

Los Eventos de Acción son propiedad de sus Acciones relacionadas. Una Acción puede activar al evento durante su ejecución o cuando termina, sea de una forma normal o con una excepción.

Los eventos del sistema son eventos independientes, que están en el nivel de modelado de interacción de flujo. Los eventos del sistema son el resultado de un evento de terminación de ejecución de una acción o de una expresión activadora o *trigger* tal como un momento específico en el tiempo, o eventos de condición especial como un problema en la conexión de red.

Los eventos de captura poseen un conjunto de enlaces de navegación. Una expresión de interacción de flujo se utiliza para determinar cuál de los flujos de navegación se siguen como consecuencia de la ocurrencia de un evento. Cuando se produce un evento y no se tiene una expresión de interacción de flujo, todos los enlaces de navegación asociados con el evento se siguen sin un orden en particular. Un evento puede tener una expresión para activarse que determina si el evento está habilitado o deshabilitado [44]. Estas características se verán a mayor profundidad cuando se analice el apartado de condiciones de visibilidad.

Los eventos en WebRatio son desarrollados por medio de componentes en un sector diferente al modelado de datos, el mismo comprende las tres operaciones básicas de un mantenimiento que son la creación, modificación y eliminación. Tiene también tres componentes adicionales para trabajar con relaciones de cardinalidad N-N como son el conectar, desconectar y reconectar, los mismos nos permiten interactuar con la tabla auxiliar generada por el mapeo realizado de la relación múltiple.

Para una mejor comprensión de los servicios que maneja WebRatio se procederá a dividirlo en categorías acorde a la clasificación de eventos.

Eventos de Captura

Dentro de los eventos de captura tenemos la siguiente clasificación:

- **Eventos de visualización de elementos.** Para la ejecución de este tipo de eventos, WebRatio maneja un amplio rango de componentes o unidades que son

conocidos como Componentes de vista. Dentro de estos componentes se encuentran: detalles, detalles múltiples, navegador, lista simple, lista, lista seleccionable, lista jerárquica, lista jerárquica recursiva, formulario, formulario múltiple.

Figura 39. Componentes de visualización de WebRatio

Dentro de los componentes de visualización tenemos:

- **Detalles.** Permite mostrar los detalles de una instancia seleccionada, los detalles que deseamos visualizar deberán ser seleccionados por el analista.

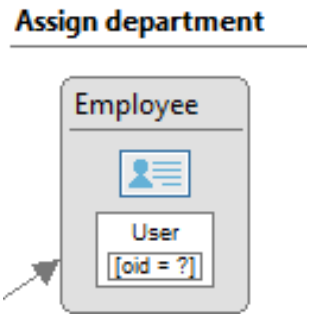


Figura 40. Ejemplo de la unidad Detalles en WebRatio

Como se puede apreciar en la gráfica anterior, se cargarán los detalles de la clase Empleado y tiene un argumento de entrada, por medio del uso de un selector, que es el identificador, esto quiere decir que, de todas las instancias que se encuentren almacenada en esa tabla se cargará la información únicamente del empleado seleccionado.

- **Detalles múltiples.** Permite mostrar los detalles de una o varias instancias seleccionadas permitiendo una navegabilidad entre las mismas.
- **Navegador (Scroll).** Es una utilidad que nos permite navegar entre varias instancias que hayan sido invocadas a la vez.

- **Lista Simple.** La lista simple nos permite mostrar un listado de las instancias que pertenecen a una clase, originalmente nos muestran todas las instancias que pertenecen a una clase determinada, sin embargo, permite el uso de selectores que actúan como filtros y sobre los cuales podemos colocar las condiciones que se requieran para una mejor visualización de los datos.

  Un ejemplo de lista simple sería el siguiente:

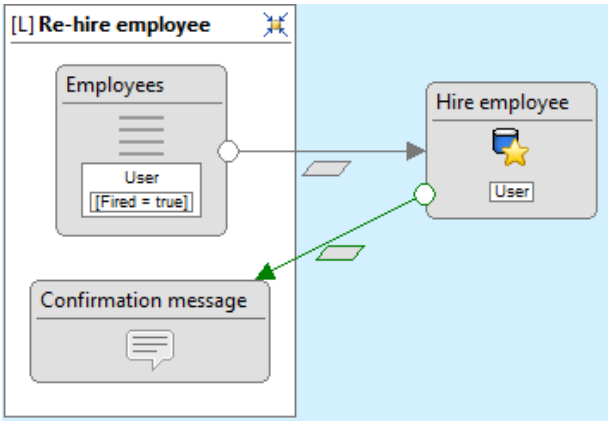


Figura 41. Ejemplo de lista simple en WebRatio

Como se puede apreciar, la lista Empleados no tiene ningún parámetro de entrada, pero tiene un selector precargado que nos permite cargar únicamente a los empleados que hayan sido despedidos por medio de la sentencia Fired = true.

- **Lista.** Tiene un funcionamiento similar a lista, pero con más propiedades como la paginación y permitir cambiar de ser una lista simple a una seleccionable, una muestra de sus propiedades se muestra a continuación:

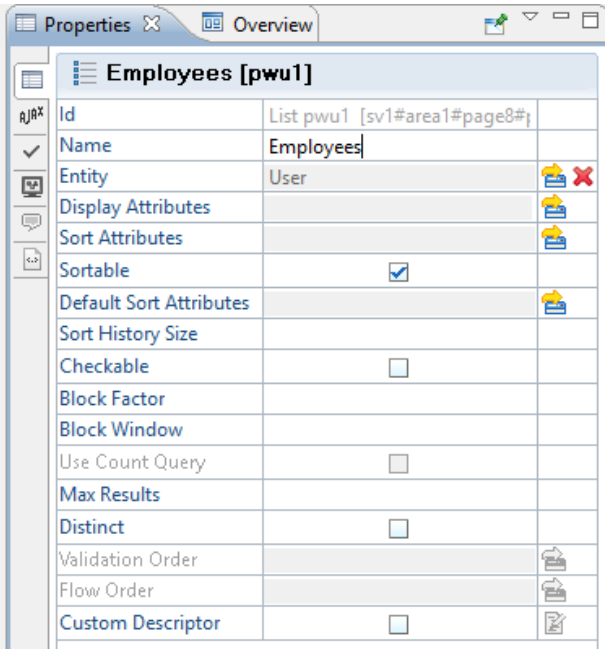


Figura 42. Ejemplo de propiedades de una lista en WebRatio

Para mostrar atributos lo primero que debemos realizar es dirigirnos al ícono que se muestra en la gráfica posterior y en la ventana auxiliar que se despliega seleccionamos los atributos que deseamos visualizar como se muestra en la siguiente gráfica:

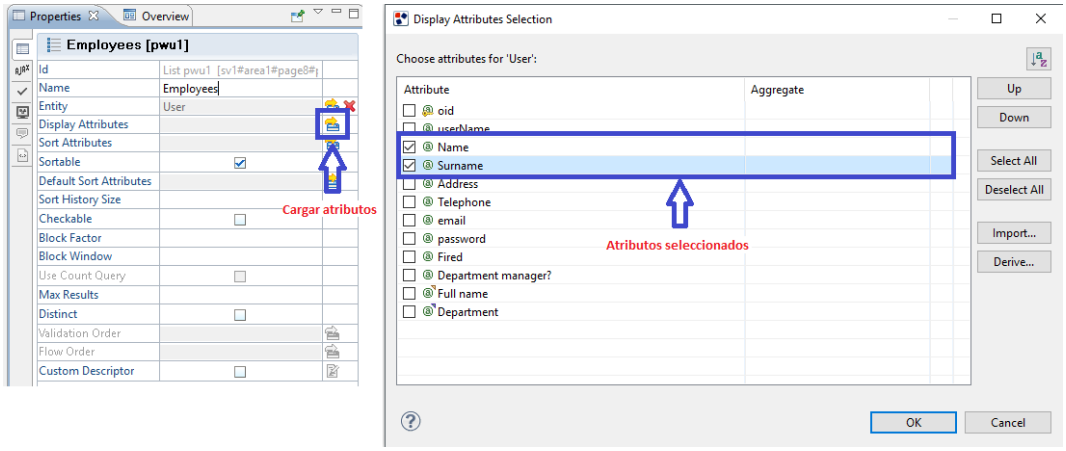


Figura 43. Ejemplo de carga de atributos en una Lista en WebRatio

Como se puede apreciar, se pueden realizar importaciones de nuevos atributos, así como derivaciones a nivel de diseño conceptual IFML acorde a las necesidades del analista.

- **Lista seleccionable.** La lista seleccionable tiene una funcionalidad parecida a la lista, con la característica de que permitirá seleccionar a las instancias cargadas en la misma para poder enviar un envío masivo de instancias hacia otro evento, transacción u operación.
- **Lista jerárquica.** Esta lista es una representación gráfica de una relación establecida en el modelo de datos, por medio de la cual tendremos acceso a la instancia padre a la que pertenecen instancias hijas, esto se verá reflejado en relaciones con cardinalidad 1 a N y de N a N ya sea con otras instancias como consigo misma. Cabe mencionar que los niveles a los cuales se puede llegar son infinitos, pero deben ser expresados manualmente por el analista.
- **Lista jerárquica recursiva.** Tiene una funcionalidad similar a la lista jerárquica con la diferencia de que los niveles jerárquicos son generados de una forma recursiva hasta llegar a la instancia raíz.
- **Formulario.** Es el componente en el que se nos permite ingresar la información de una determinada instancia. Para ello se pueden cargar los campos de una forma manual o por medio de la utilización del asistente. Hay que considerar que el empleo del componente de formulario nos permite crear los tipos de campos acorde sea el tipo de datos, es decir, si tenemos un tipo de dato *blob* el campo que visualizará el usuario final tendrá ya configurado la visibilidad de carga de archivos, si manejamos el tipo de dato *date* o fecha, el componente visualizará en su sección front-end la facilidad de un calendario. Esta facilidad es muy conveniente pues elimina la necesidad de programar este tipo de campos de una forma manual y mejora la usabilidad del sistema.
  Por otra parte, tenemos los valores precargados que le dan la capacidad al formulario de tener campos tanto precargados (para una acción de

modificación) como también campos nuevos para la creación de una nueva instancia.

A continuación, se presenta un gráfico que ejemplifica lo descrito anteriormente en lo que refiere al uso del asistente para la creación de nuevos campos, así como la opción para que los mismos acepten valores precargados.

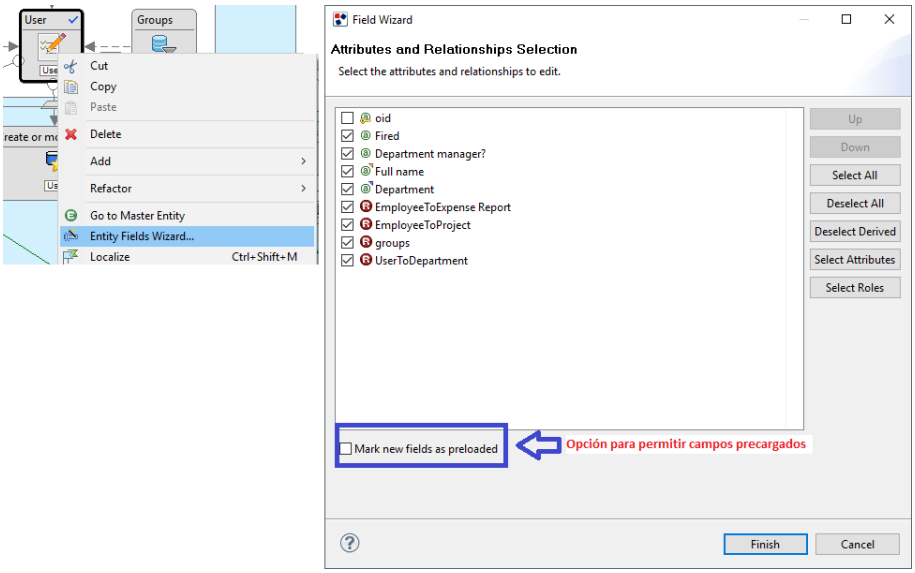


Figura 44. Ejemplo de creación de un formulario con la ayuda de un agente en WebRatio

- **Formulario múltiple.** El formulario múltiple es similar al anteriormente mencionado, con la característica de que permite enviar múltiples instancias en el mismo formulario.

- **Eventos de Acción.** Dentro de los eventos de acción tenemos los siguientes: Creación, modificación, eliminación, conexión, desconexión, reconexión, procedimientos almacenados y de no operación.

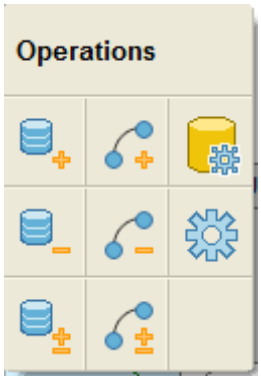


Figura 45. Componentes de Operaciones de WebRatio

- **Creación.** El componente de creación nos permite crear nuevas instancias, tiene la opción *Bulk* que nos permite grabar múltiples instancias al mismo tiempo. Para dar funcionamiento a la unidad de creación se debe crearla y seleccionar la clase a la cual va a estar relacionada. Un componente de creación no puede relacionado a más de una instancia.

- **Modificación.** El componente de modificación permite modificar una instancia en la base de datos, este componente tiene como argumentos de entrada, fuera de los campos de la instancia, el identificador que servirá como condición para poder encontrar la instancia seleccionada y sobre la misma realizar el cambio. El componente de modificación viene con un selector por defecto en el que se especifica el identificador de la instancia, en caso de estar vacío este selector daría un error.

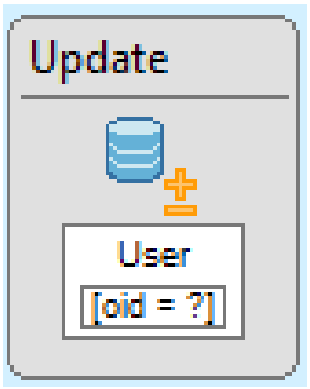


Figura 46. Componente de modificación en WebRatio

Como se puede apreciar en la gráfica anterior, el componente tiene que ser creado para posteriormente enlazarlo a la clase, el selector que carga el identificador es creado automáticamente.

- **Eliminación.** Este componente nos permite realizar una eliminación de una instancia en la base de datos, requiere únicamente un argumento de entrada que es el identificador de la instancia y el resto de datos los elimina con esta condición.
- **Conexión.** La unidad de conexión es una unidad de operación utilizada para habilitar la asociación entre instancias de dos entidades (la entidad de origen y la entidad de destino), conectadas mediante una función de relación predefinida. La unidad de conexión puede administrar instancias únicas o conjuntos de instancias para conectar una instancia de la entidad de origen a un conjunto de instancias de la entidad de destino (o viceversa) o conjuntos de instancias de entidades de destino y de origen con la misma cardinalidad. Las instancias de las entidades origen y destino que se van a conectar se eligen aplicando condiciones [45].

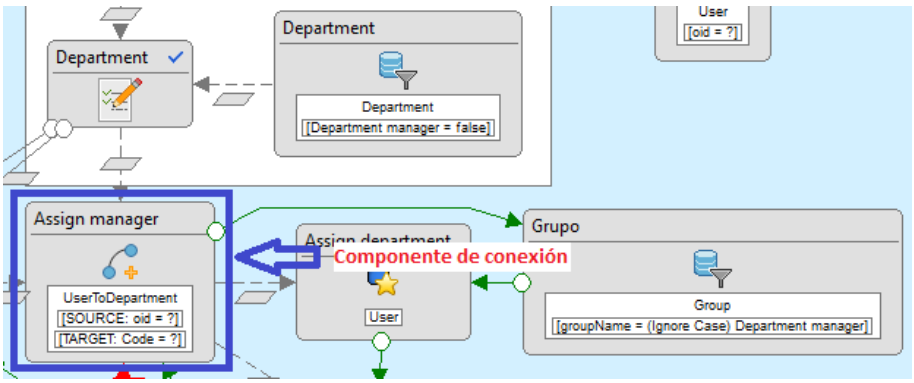


Figura 47. Ejemplo de componente de conexión en WebRatio

- **Desconexión.** La unidad de desconexión es una unidad de operación utilizada para eliminar la asociación entre instancias de dos entidades (la entidad de origen y la entidad de destino), conectadas mediante una función de relación predefinida. La unidad de desconexión puede administrar instancias únicas o conjuntos de instancias para desconectar una instancia de la entidad de origen de un conjunto de instancias de la entidad de destino (o viceversa) o conjuntos de instancias de entidades de destino y de origen con la misma cardinalidad realizando una desconexión por pares. Las instancias de las entidades origen y destino que se van a conectar se eligen aplicando Condiciones [46].
  A continuación, se puede observar un ejemplo del componente de desconexión en el que se eliminan las relaciones para poder quitar la asignación de un administrador de departamento.

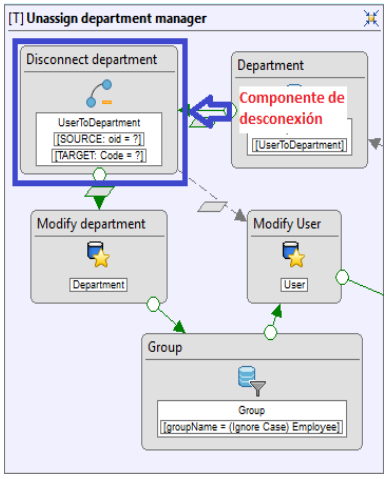


Figura 48. Ejemplo de componente de desconexión en WebRatio

- **Reconexión.** La unidad de reconexión es una unidad de operación que se utiliza para modificar la asociación entre instancias de dos entidades (la entidad de origen y la entidad de destino), conectadas mediante una función de relación predefinida. La unidad de reconexión puede administrar instancias únicas o conjuntos de instancias para conectar o desconectar una instancia de la entidad fuente de un conjunto de instancias de la entidad objetivo (y viceversa) o conjuntos de instancias de las entidades de destino y de origen con la misma cardinalidad. Las instancias de las entidades origen y desconectarán que se conectarán o desconectarán se elegirán aplicando Condiciones.
  La Unidad de Reconexión se comporta como una unidad de desconexión y una unidad de conexión ejecutada inmediatamente después. La unidad de desconexión borrará todas las conexiones entre todas las instancias de entidad de origen y todas las instancias de entidad de destino y la unidad de conexión conectará posteriormente las instancias de entidad de origen recién definidas con las instancias de entidad de destino. Utilizando las condiciones de desconexión de destino, es posible elegir explícitamente las instancias que se van a desconectar [47].
- **Procedimiento Almacenado.** La operación de procedimiento almacenado es una operación que permite llamar a un procedimiento almacenado en una base de datos. El procedimiento se debe crear previamente en la base de datos.

- **No Operación.** Es un componente que nos puede servir como auxiliar para procedimientos como condiciones de visibilidad.

4.8. Transacciones

Una transacción es una unidad de proceso molecular que se encuentra compuesto por servicios.

4.8.1. Integranova

En Integranova se maneja el concepto del objeto ***THIS***, que representa el objeto donde los servicios van a ser ejecutados. La fórmula que emplea puede ser modificada por medio del uso de servicios y esta permite modificar el valor de varios atributos variables del objeto actual o de sus relaciones al igual que el valor de cualquier atributo variable u objetos relacionados a través de una ruta de enlace con el objeto THIS.

En una transacción local no se pueden ejecutar servicios sobre un argumento, ejecutar servicios de clases que no se encuentren relacionadas a la clase de transacción y ejecutar funciones de usuario. En el siguiente ejemplo se muestra un ejemplo de transacciones y las partes que lo conforman.

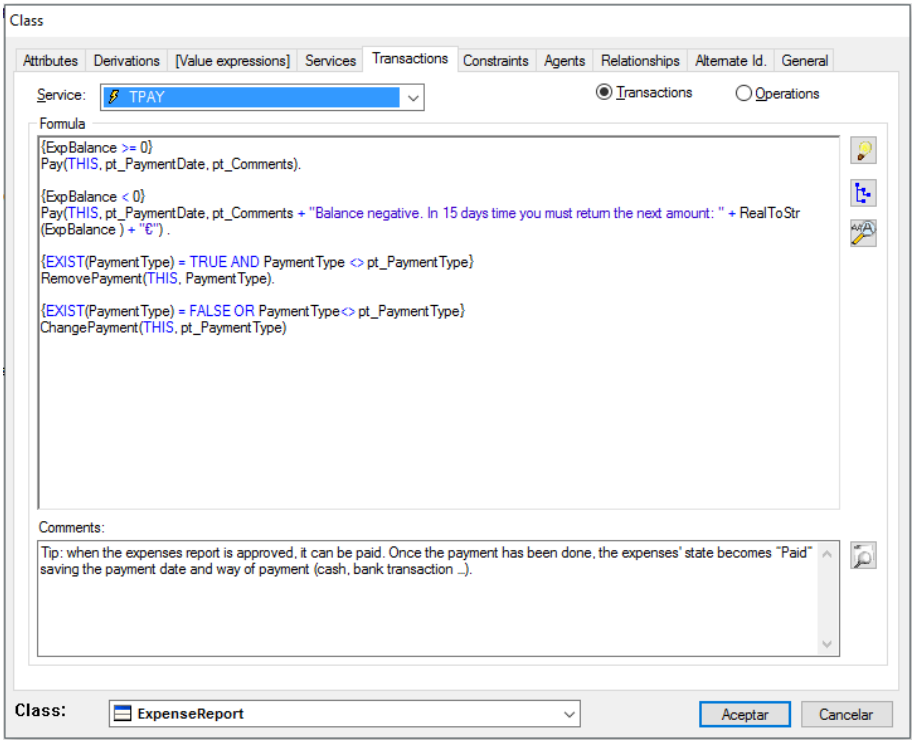


Figura 49. Ejemplo de transacción en Integranova

Transacciones de Creación. Son aquellas transacciones que crean al menos el objeto *THIS*. El primer servicio que se ejecutará en la transacción debe incluir la creación del objeto *THIS*. Si un objeto relacionado es creado en la fórmula, no es necesaria una transacción de creación.

Transacciones de Destrucción. Son las transacciones que elimina el objeto *THIS*, donde el último servicio debería destruir la misma clase de la transacción. En este tipo de transacciones, el servicio de destrucción no puede ser opcional. A continuación, una gráfica que ejemplifica de una mejor manera la forma de las transacciones de destrucción, nótese la destrucción del objeto THIS al finalizar la misma.

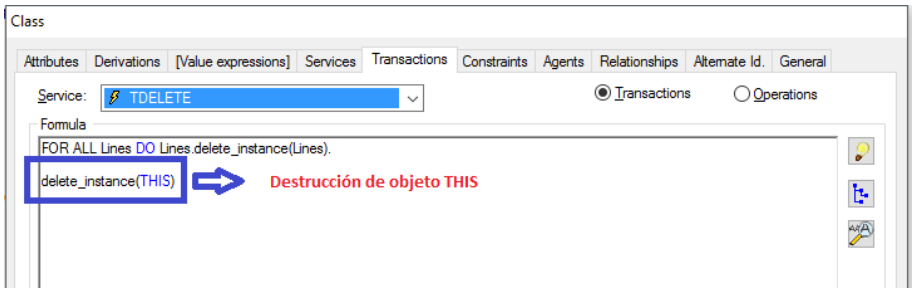


Figura 50. Ejemplo de transacción de destrucción en Integranova

Otras transacciones. Son aquellas transacciones que no crean o eliminan el objeto THIS. Estas transacciones pueden emplear servicios de creación y destrucción, pero siempre sobre objetos relacionados, nunca sobre el objeto *THIS* como tal.

Los elementos que pueden ser empleados en fórmulas de transacción son: atributos, argumentos de entrada y salida, servicios visibles (no del sistema), funciones del usuario, funciones estándar y operadores.

La sintaxis que se emplea en una fórmula de transacción es la siguiente:

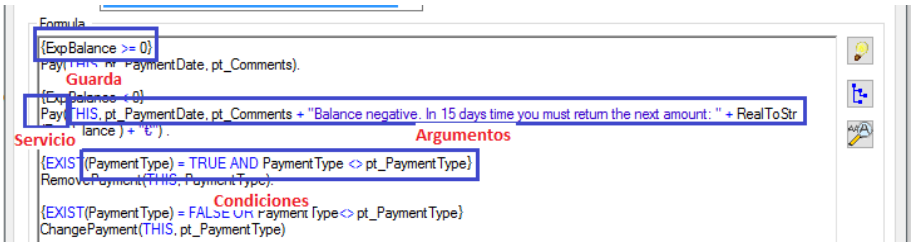


Figura 51. Ejemplo de fórmulas en transacción en Integranova

Guarda. Son las condiciones que pueden ser incluidas dentro de la transacción u operación. Esto se realiza de tal forma que si se cumple la condición el servicio es ejecutado, caso contrario se saltaría la ejecución de ese servicio. Los servicios pueden ser eventos o transacciones. Si existe más de un servicio después de la condición, tiene que ser agrupados dentro de la transacción. Los paréntesis no están permitidos, sino que las agrupaciones se realizan por medio de un punto (.).

Servicios. Los servicios deben pertenecer a la clase donde la transacción es definida. Para ello emplea también **Argumentos** que pueden ser inicializados mediante el uso de: atributos de la clase, argumentos de entrada de la transacción, contantes, fórmulas, etc.

Condiciones. Dentro de las condiciones que se pueden emplear tenemos la de **FOR ALL** que no es más que un bucle que corre un servicio sobre un grupo de instancias que pertenecen a un rol determinado hasta completar una condición.

4.8.2. WebRatio

Trabajar en aplicaciones web del mundo real puede requerir modelar un proyecto grande. En esta situación es importante poder gestionar el proyecto para evitar la pérdida de control y la duplicación de piezas del modelo. WebRatio permite trabajar fácilmente en grandes proyectos gracias a sus capacidades de modelado, que permiten dividir un modelo en partes, y reutilizarlas donde lo necesite. Las principales ventajas de este enfoque son la facilidad de mantenimiento, así como la legibilidad y elegancia del modelo obtenido. De hecho, se puede "centralizar" la parte del modelo de interacción del usuario que se desea reutilizar y colocarla en un solo lugar, y ese será el único lugar donde tendrá que realizar cambios, ahorrando así tiempo y reduciendo errores. También permite recorrer de una mejor manera el modelo IFML [48].

Cuando se emplea WebRatio, el empleo de transacciones se lo realiza mediante el uso de módulos que permiten integrar varios componentes y unidades (servicios en Integranova) para poder llegar a un estado determinado. A diferencia de las Operaciones en Integranova, WebRatio maneja una política de todo o nada, esto quiere decir que en caso de que un componente presente un error, no se pueden seguir usando el resto.

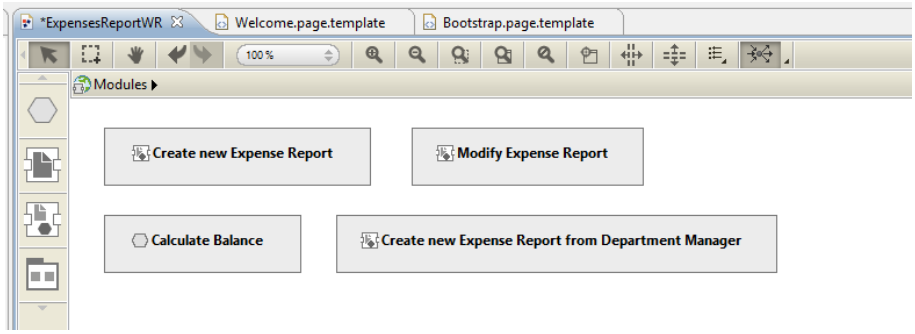


Figura 52. Ejemplo de módulos en WebRatio

En WebRatio se pueden crear tres tipos diferentes de módulos, dependiendo del contenido que desee agregar a ellos: módulos de contenido, híbridos y acciones

Módulo de contenido. Es un tipo de módulo que contiene una colección de componentes visuales que se pueden utilizar como una parte del contenido de una página. Un ejemplo de módulo de contenido en una aplicación web de CRM es el conjunto de componentes visuales necesarios para mostrar la información actual de un usuario. Esta información suele aparecer en todas las páginas de la aplicación Web. Sin el concepto de módulo, se vería obligado a modelar el mismo conjunto de componentes de contenido en cada página de un proyecto web.

Módulos híbridos. Es una definición de módulo que contiene una colección de páginas y acciones. Un ejemplo de módulo híbrido en una aplicación web consiste en la tarea crear un reporte de gastos nuevo. En donde el modelado de datos contiene varias entidades y se requiere que el proceso llame a componentes tanto visuales como de acción en una secuencia de pasos determinado. Como se podrá apreciar en la siguiente gráfica, el módulo contiene páginas, componentes de visualización y de acción ligados entre sí por medio de enlaces, estos enlaces permiten determinar un camino para la obtención del objetivo por medio de una agrupación de operaciones. Esta agrupación

puede ser “refactorizada” para convertirse a su vez en un módulo de Acción que será analizado posteriormente.

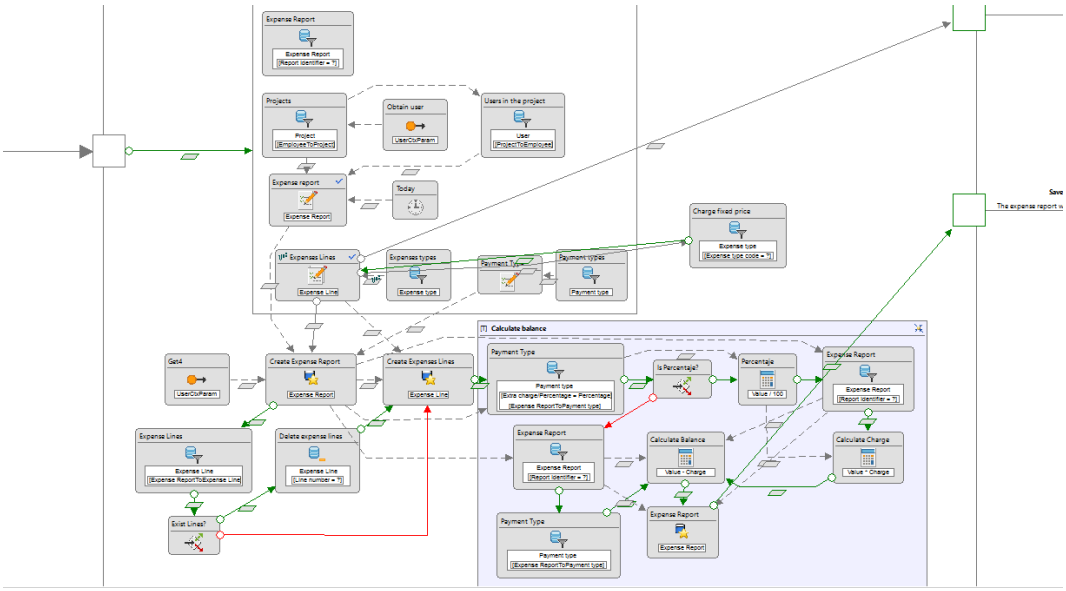


Figura 53. . Ejemplo de módulo híbrido en WebRatio

Acciones. Las acciones nos permiten agrupar componentes de operación, estos componentes son necesarios para la realización de un grupo de operaciones secuenciales para la obtención de uno o varios argumentos de salida concretos. Un ejemplo de nuestro ejercicio de reporte de gastos se encuentra en el cálculo del balance puesto que requiere de cálculos internos para poder ser gestionado, estos cálculos son realizados por componentes o grupos de componentes para obtener de esta manera un grupo de argumentos de salida que satisfaga la operación.

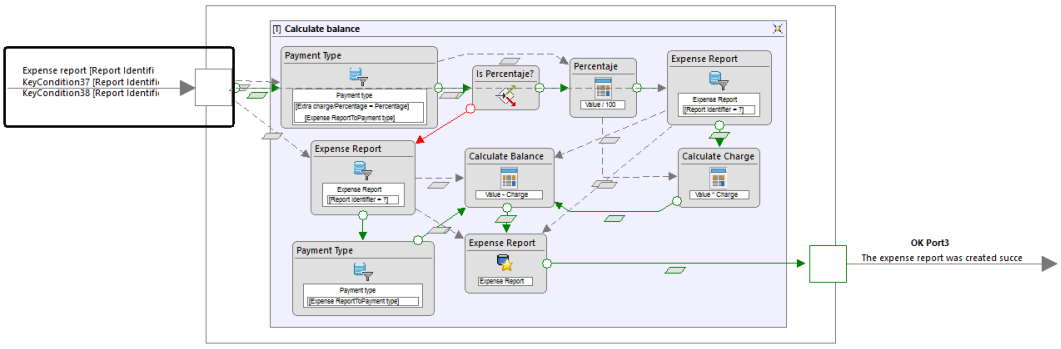


Figura 54. Ejemplo de Acción en WebRatio

Argumentos. Los argumentos en WebRatio son manejados por medio del concepto de puertos. Existen dos tipos de puertos, los de entrada y los de salida [49].

- **Puertos de Entrada.** Un punto de interacción entre un módulo o acción y su entorno que recoge los flujos de interacción y los parámetros que llegan al módulo. En los puertos de entrada, es posible tener parámetros definidos que pueden se pueden enviar por medio y a través de los componentes presentes en el módulo o acción.

- **Puerto de Salida.** Es un punto de interacción entre el módulo u acción y su entorno que recoge los flujos de interacción y los parámetros que salen del módulo. Existen a su vez dos tipos de puertos de salida que son:
  - **Puerto OK.** Es el puerto de salida de una definición de acción que representa el éxito de la ejecución de la acción.
  - **Puerto KO.** Es el puerto de salida de una definición de acción que representa el fallo de la ejecución de la acción.

4.9. Precondiciones

Las precondiciones son condiciones que deben ser satisfechas antes de la ejecución de una acción siendo definidas por fórmulas booleanas bien formadas. Como lógica de las precondiciones se tiene que si la condición es satisfecha el servicio es ejecutado. Cuando la condición no es satisfecha correctamente el servicio no podría ser ejecutado.

4.9.1. Integranova

Una precondición viene determinada por el servicio (evento, transacción u operación) sobre el que se va a evaluar y el agente que lo va a ejecutar, en caso de no tener ningún agente definido, se asumirá que la precondición deberá ser respetada por todos los agentes, esto quiere decir que las precondiciones se encuentran relacionadas con las acciones, por esta razón se puede controlar la ejecución de un servicio dependiendo del tipo de usuario.

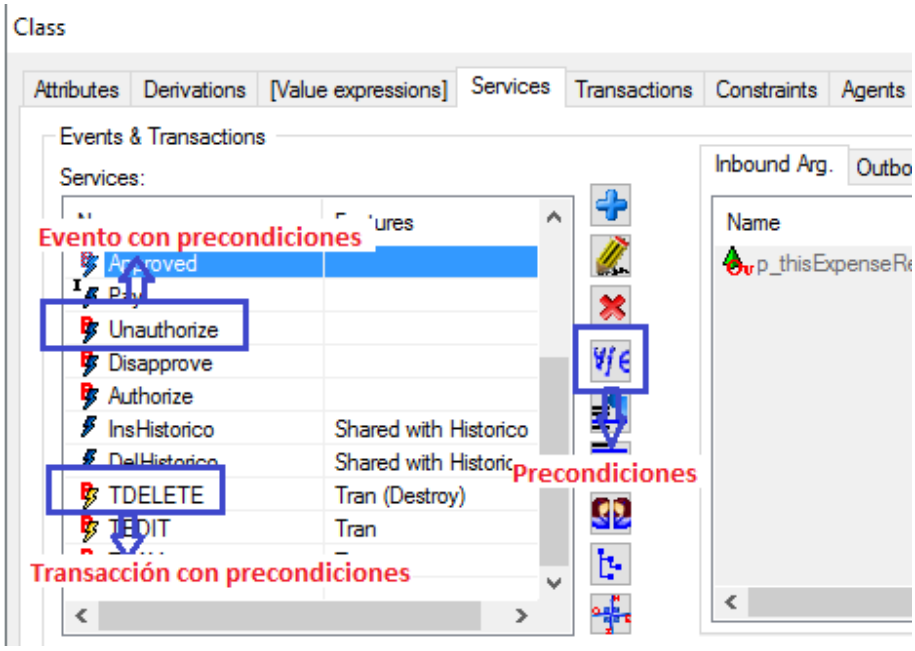


Figura 55. Ejemplo de precondiciones para servicios y transacciones en Integranova

Una vez que se haya seleccionado el evento, transacción u operación sobre el que deseamos ejecutar la condición de visibilidad vamos a tener el cuadro de diálogo de las precondiciones donde se puede definir la o las fórmulas requeridas para que se cumpla la condición que deseamos expresar. En el siguiente gráfico podemos apreciar cómo se visualizas las precondiciones en Integranova.

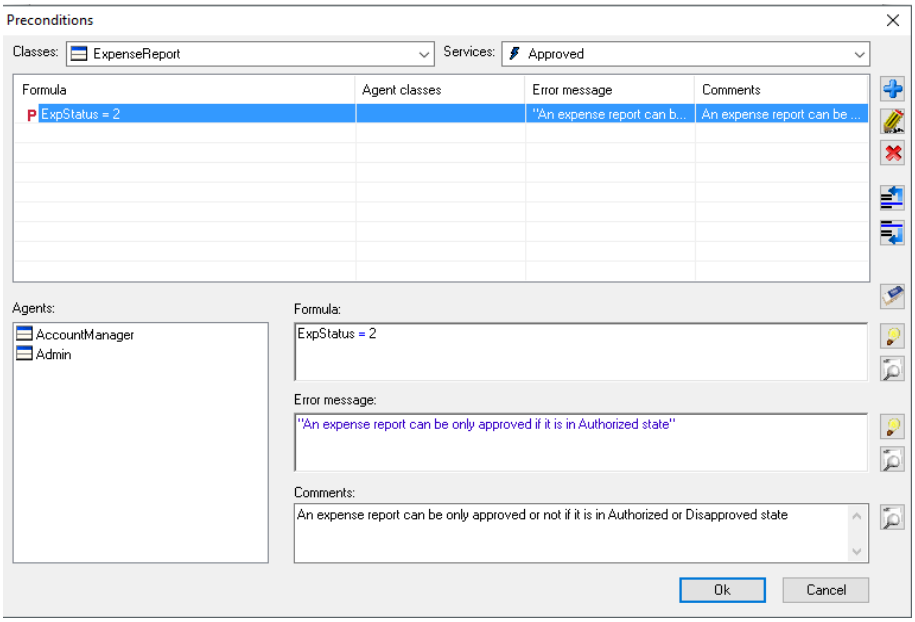


Figura 56. Ejemplo de cuadro de diálogo de precondiciones en Integranova

Las precondiciones en Integranova respetan la herencia de las entidades, esto quiere decir que en la clase hija se pueden agregar precondiciones a los servicios definidos en la clase padre. Cuando una instancia es especializada, todas las precondiciones son revisadas, tanto las que se encuentran en la clase padre como en la hija.

Restricciones de integridad. Son condiciones que deben ser verdaderas para todos los estados posibles de un objeto. Estas serán verificadas después de la ejecución de los servicios que modifican el estado de un objeto. El mensaje de error será mostrado al usuario cuando la restricción no haya sido satisfecha. Si las fórmulas de las restricciones de integridad son definidas sobre atributos constantes o roles estáticos, entonces la restricción será revisada después de la creación del objeto.

Para las restricciones se manejan criterios similares a las precondiciones en el sentido de que cuando un objeto es especializado, todas las restricciones, tanto del padre como del hijo, son revisadas.

En resumen y realizado una comparativa entre las precondiciones y las restricciones de integridad, podemos concluir que las primeras se definen en un servicio, mientras que las restricciones se definen en la clase-; además, las precondiciones deben ser ciertas para el estado actual de un objeto mientras que en la integridad las restricciones deben ser ciertas para todos los estados posibles del objeto.

Las condiciones previas se verifican antes de la ejecución del servicio y las restricciones después de la ejecución de cualquier servicio que modifique el estado de la clase es por esta razón que son más recomendadas en lugar a las restricciones de integridad. Esto se debe a que las restricciones se evalúan siempre que se ejecuta un servicio (que cambia el estado del objeto), para ello, si la condición es muy compleja, su cálculo puede ralentizar la ejecución.

4.9.2. WebRatio

WebRatio maneja las precondiciones de varias maneras, entre ellas tenemos las reglas de validación, condiciones y visibilidad y selectores. Estas se manejan a nivel de componentes en el modelado IFML y pueden ser configuradas directamente por el analista.

4.9.2.1. Reglas de validación.

Se utilizan para comprobar que los datos insertados por el usuario a través de la interfaz de la aplicación cumplan con los requisitos del modelado de datos. Como un ejemplo podemos decir que los datos que se insertan en un formulario (o en cualquier otro componente visual o, de entrada), se pueden aprobar mediante reglas de validación, que son especificaciones para las condiciones que deben cumplir un dato para ser considerado como válido. Existen varias reglas de validación dentro de WebRatio detalladas a continuación:

- **Captcha.** Nos permite tener una validación de captcha, en la que antes de ser validado el campo se debe comprobar que no es un robot el que está ingresando la información.
- **Colección.** Valida si el campo se encuentra o no se encuentra dentro de una Entidad.
- **Comparación.** Valida si el campo es o no es igual a otro campo del mismo formulario.
- **Tarjeta de crédito.** Validación que nos permite evaluar si el dato ingresado corresponde a un número de tarjeta válido.
- **Correo electrónico.** Valida que el dato ingresado tenga el formato de un correo electrónico válido.
- **Similar.** Evalúa si el campo ingresado es igual a un valor o expresión agregada directamente por el analista.
- **Campo obligatorio.** Verifica que el campo no se encuentre vacío antes de ser ingresado a la base de datos.
- **Expresión regular.** Es un campo más abierto que nos permite establecer, por medio de expresiones regulares, la validez de un campo. Esto puede ser empleado para verificar un número de identificación o matrícula de vehículo.
- **Validación de tipo.** Nos permite validar que el tipo de datos ingresados en un campo tenga el formato correcto.
- **Dimensión de valor.** Nos permite realizar una validación del tamaño de los valores ingresados.

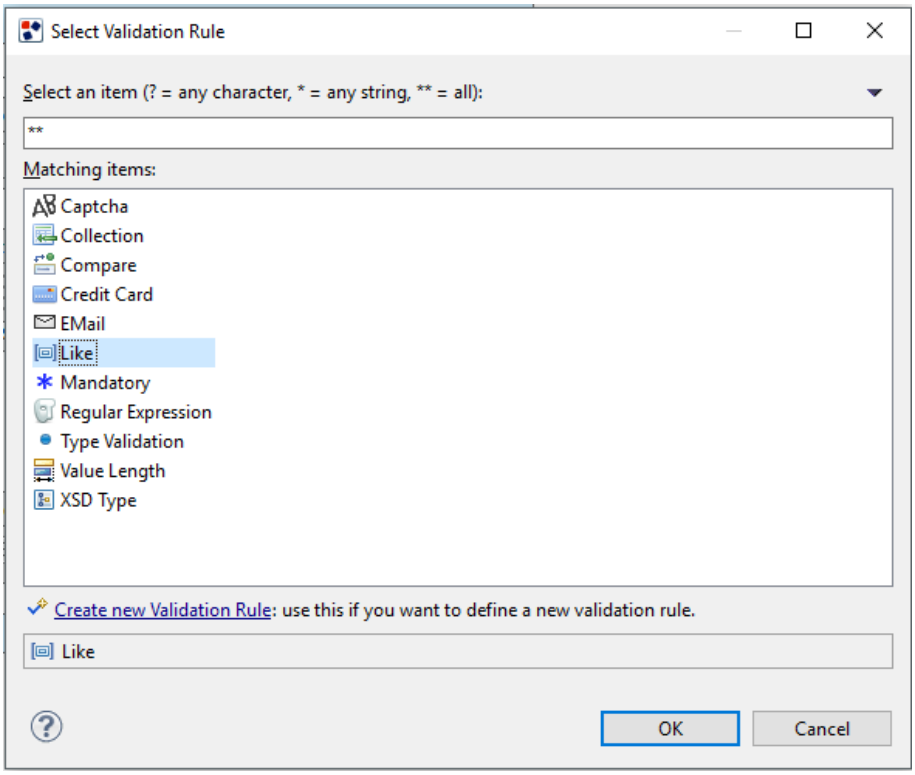


Figura 57. Ejemplo de reglas de validación en WebRatio

4.9.2.2. Condiciones de visibilidad.

Son características le permiten ocultar, deshabilitar o mostrar elementos en las páginas generadas de la aplicación web si se verifican las condiciones especificadas. Es posible integrar el perfilado horizontal y vertical de la aplicación Web, filtrando datos en filas y columnas según el perfil de usuario. Un ejemplo de perfilado horizontal filtra la lista de productos según el país del cliente actual. Esto implica que el modelo de datos debe contener información que conecte los productos y las entidades del país.

Las condiciones, junto con las funciones de Ajax, ayudan a realizar formas ricas y modulares. Es posible desactivar los campos según condiciones específicas y ocultar partes de un formulario según el número de campos rellenados. Las condiciones de visibilidad permiten tanto desactivar como ocultar elementos de diseño que se enumeran a continuación:

- Atributos
- Campos
- Enlaces
- Unidades
- Celdas
- Grillas

Al aplicar una Condición es posible elegir aplicar su negación usando un botón dedicado. Esto significa aplicar la condición usando una negación lógica que invierte el resultado esperado. Para comenzar con Condiciones, es necesario aprender dos conceptos básicos que son: variables y expresiones de activación [50].

Variable. El punto de partida para definir una condición es el concepto de variable. Se puede crear una variable como un subelemento de página. Los componentes que forman parte de la variable son:

- **Nombre.** Es el nombre utilizado para la variable, debe respetar las convenciones de nomenclatura de Java, ya que corresponden a una variable de secuencias de comandos de Groovy.
- **Tipo.** Hace referencia al tipo de datos asociados con la variable. Si se deja vacío, no se realiza ninguna conversión de datos.
- **Componente.** Es el componente de la página en la que se basa la variable. Si se rellena esta propiedad, se va a definir una variable basada en componentes.
- **Parámetro.** Es uno de los parámetros de salida del componente elegido en la propiedad anterior. Se puede elegir el parámetro de salida de una lista desplegable que muestre todos los valores disponibles para ese componente (también se puede elegir valores no utilizados como atributos de visualización).
- **Valor.** Es el valor predeterminado de la variable. Si se llena dejando la propiedad componente vacía, se podrá definir una variable libre.

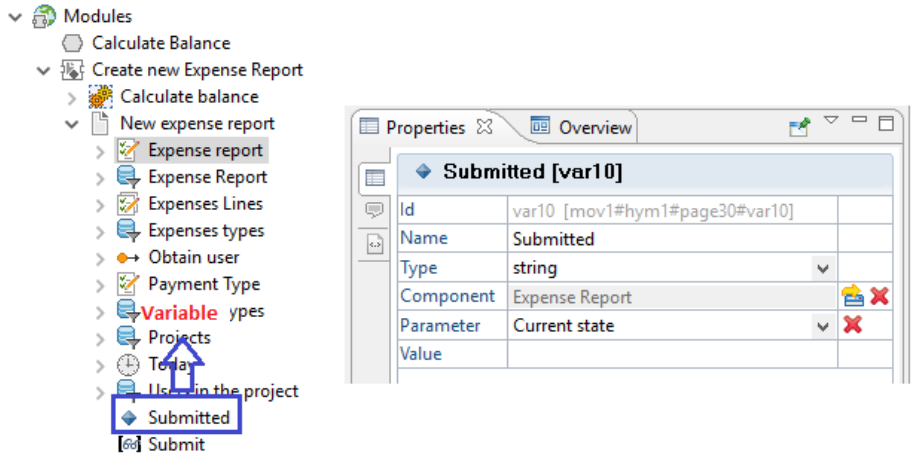


Figura 58. Ejemplo de variable con sus propiedades en WebRatio

Expresión de Activación. Una expresión de activación es una expresión Groovy que devuelve un valor booleano que especifica si la condición se cumple o no. Puede definir expresiones de activación basadas en variables que empiezan desde páginas o páginas maestras. Las propiedades con las que cuenta una expresión de activación son:

- **Nombre.** Aunque no es estrictamente necesario, es una buena práctica utilizar una convención de nomenclatura Java para la expresión.
- **Expresión.** Es la expresión Groovy que define la condición. Aquí puede escribir un script de Groovy que utilice las variables definidas en la página o en la página maestra.

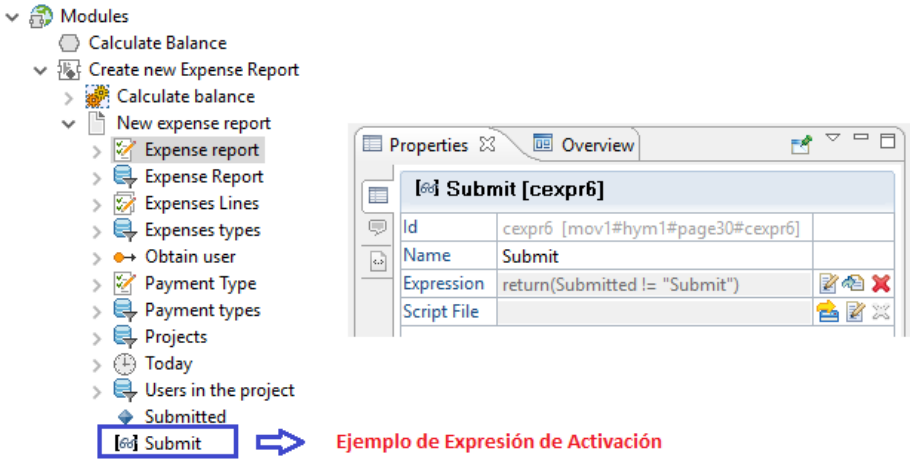


Figura 59. Ejemplo de expresión de Activación en WebRatio

4.9.2.3. Selectores

Son un conjunto de criterios de selección que recuperan las instancias de la entidad del componente. Los componentes de visualización definidos sobre entidades pueden requerir un selector para determinar las instancias de la entidad utilizada para construir su contenido. El selector contiene una o más condiciones para un componente o un campo precargado. Tales condiciones se utilizan para hacer la consulta sobre el origen de datos, que en realidad determina las instancias en las que se basa el contenido del componente. Existen tres tipos de condición:

- **Condición de Atributo.** Una condición de atributo selecciona sólo las instancias de entidad para las cuales el valor de algún atributo satisface un predicado. Por ejemplo, una condición de atributos para una lista simple puede ser: "Incluir en un índice de combinaciones sólo las coincidencias de empleados que se encuentre en estado de despedidos"
- **Condición de Clave.** Una condición de clave selecciona sólo las instancias de entidad para las cuales el valor de algún atributo clave satisface un predicado.
- **Condición de Relación.** Una condición de relación selecciona sólo las instancias de entidad que están relacionadas con alguna otra instancia de entidad por medio de una cadena específica de relación.

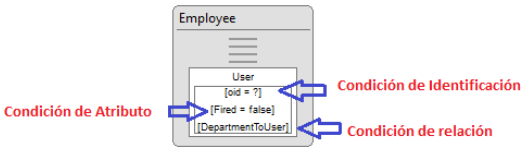


Figura 60. Ejemplo de selectores de una lista en WebRatio

4.10. Control de Agentes

Como parte fundamental de toda aplicación web, es el control de agentes o roles. Esto permitirá que diferentes tipos de usuarios puedan interactuar sobre la misma página, pero con accesos a diferentes secciones o funcionalidades de la misma.

4.10.1. Integranova

Integranova maneja el concepto de agentes, donde una relación de agente está definida entre dos clases, llamada interface, en la misma existe una relación con las clases del servidor y del agente como tal. Un agente en Integranova tiene una funcionalidad o se expresa como una clase. En la definición de un agente, el analista indica que atributos y atributos de otras clases (servidor) están disponibles para ser visualizadas y ejecutadas por un agente.

En Integranova se pueden visualizar las relaciones con los agentes por medio de una línea entrecortada como se ilustra en la siguiente imagen:

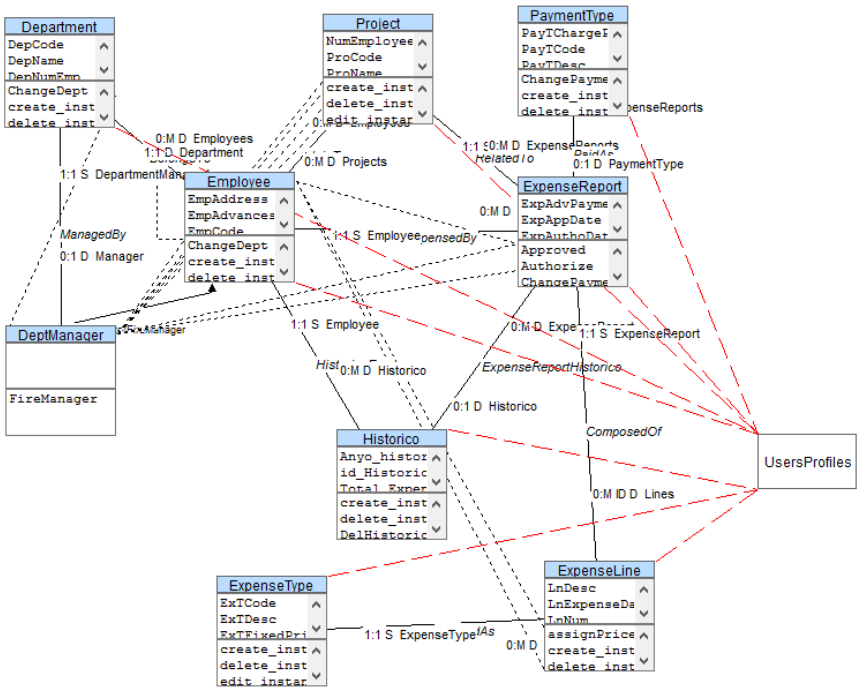


Figura 61. Ejemplo de visualización de actores en Integranova

Alcance de un agente. Un agente puede tener la característica de *All Classes* que tiene una visibilidad sobre todos los servicios, atributos, roles y clases del sistema completo, un ejemplo de esto es el administrador.

Los elementos de un agente son:

- **Clase servidor.** Es la clase en la que los elementos pueden ser vistos, ejecutados o navegados.
- **Agente.** Puede ver, activar o navegar para los atributos, servicios o roles de la clase de servidor.
- **Temporalidad.** Es una propiedad que indica si la ejecución de un servicio por el agente debe ser registrada.
- **Mensaje de ayuda.** Son los comentarios acerca de las que definiciones de un agente.
- **Fórmula.** Limita el acceso al agente seleccionado.

4.10.2. WebRatio

La modalidad de WebRatio para gestionar roles dentro de una aplicación parte de una serie de componentes que permiten otorgar como denegar permisos a puntos específicos de toda la especificación del dominio. Dado que el modelo web está compuesto por vistas de sitio, áreas y páginas, es posible otorgar el acceso a cada uno de ellos sólo a los usuarios deseados. Puede ser necesario que un grupo de usuarios pueda acceder a una vista de sitio en particular o que, en la misma vista de sitio, diferentes usuarios tengan acceso a diferentes áreas o páginas.

Para poder realizar esto, WebRatio emplea las entidades Usuario, Grupo y Módulo las cuáles serán descritas a continuación:

- **Usuario:** Representa a un usuario de la aplicación (en el que se incluye el nombre de usuario y contraseña dentro de los atributos de la entidad)
- **Grupo:** Representa un conjunto de usuarios con las mismas propiedades (derechos de acceso)
- **Módulo:** Representa una parte lógica de la aplicación, a la que queremos restringir el acceso (vistas de sitio, áreas y páginas).

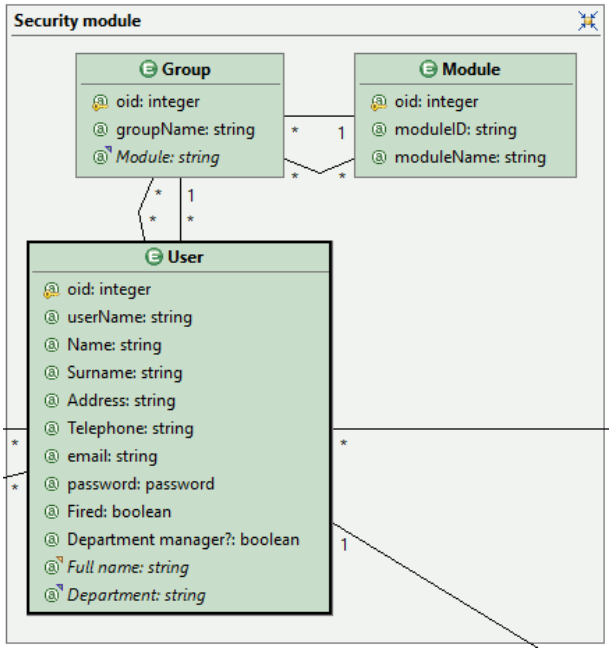


Figura 62. Ejemplo de las tres entidades que interactúan en el control de agentes de WebRatio

Las relaciones que mantienen las entidades son las siguientes:

- **Grupo a Usuario por Defecto:** Es una relación 1: N que vincula a un usuario con su grupo predeterminado
- **Usuario a Grupo:** Es una relación N: N que vincula un usuario a otros grupos (con los cuales comparte los derechos de acceso)
- **Grupo a Módulo por defecto:** Es una relación 1: N que vincula un grupo con un Módulo predeterminado
- **Grupo a Módulo:** Es una relación N: N

De acuerdo con este modelo podemos definir varios usuarios, cada uno conectado a un grupo por defecto o a otros grupos en caso de ser necesario. Cada grupo definido está

conectado a un módulo predeterminado y a otros módulos. Un módulo es la representación de una parte de la aplicación con acceso restringido. De este modo, un usuario está conectado a todas las partes de la aplicación protegida a las que puede tener acceso, a través de las relaciones con los grupos. El significado de las relaciones "Usuario a Grupo por Defecto" y "Usuario a Grupo" y del "Grupo a Módulo por Defecto" y las relaciones "Grupo a Módulo" es que cuando un usuario es identificado (a través del login), se carga automáticamente la información de su grupo predeterminado junto con la información de su módulo predeterminado. El resultado de esto es que el usuario, después del inicio de sesión, es redirigido a su módulo predeterminado. Después de la redirección, un usuario puede tener acceso a todos los módulos protegidos conectados a sus grupos.

Módulos protegidos y política de visibilidad de enlaces. Si se desea proteger un módulo, se debe configurarlo como protegido en el modelo web (marcando el indicador apropiado en la vista Propiedades), antes de agregarlo a la tabla de módulos por medio de un proceso que se explicará más adelante. En la siguiente gráfica se mostrará las propiedades de un *Site View* donde se puede seleccionar la opción de protegido.

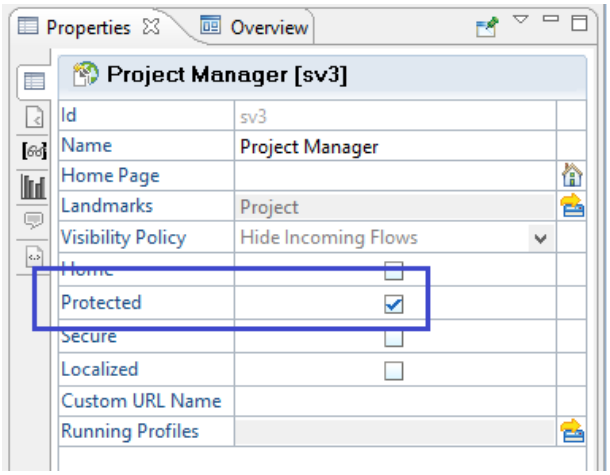


Figura 63. . Ejemplo de propiedades de un Site View con la propiedad de protegido en WebRatio

Además, al definir al módulo como protegido, es posible definir su política de visibilidad de vínculos. La directiva de visibilidad de vínculo define el comportamiento de la aplicación cuando un usuario intenta acceder a una página a la que no tiene acceso. La política de visibilidad de vínculos, tiene las siguientes características:

- Ocultar vínculos entrantes: los enlaces al módulo protegido sólo están visibles si el usuario puede acceder a él (la condición predeterminada).
- Mostrar solo los vínculos entrantes: los enlaces al módulo protegido siempre están visibles, pero si un usuario no autorizado intenta acceder a él, es redirigido a la página de inicio de sesión predeterminada.
- Deshabilitar enlaces entrantes: los enlaces al módulo protegido siempre están visibles, pero están deshabilitados si el usuario no puede acceder al módulo.

Componentes para manejo de roles. Las funciones de inicio de sesión y cierre de sesión son proporcionadas por las siguientes unidades:

- **Unidad de acceso**. Esta unidad recibe como entrada un nombre de usuario y una contraseña y verifica en la tabla de usuario si el usuario existe. Si existe el

usuario, la unidad realiza un re direccionamiento al módulo predeterminado del grupo predeterminado del usuario y establece los parámetros de contexto adecuados (UserCtxParam, GroupCtxParam). Es posible agregar un enlace OK a la unidad. Este enlace se considera cuando el usuario que intenta iniciar sesión pertenece a un grupo para el que no está definido un módulo protegido predeterminado. Por lo tanto, en este caso, puede redirigir el usuario explícitamente a una página, conectando la página con la unidad de inicio de sesión. Si se modela este enlace, pero el usuario actual tiene un módulo predeterminado asociado, simplemente se ignora y el usuario se redirecciona a su módulo predeterminado.

- **Unidad de cierre de sesión.** Esta unidad borra la información de usuario y grupo almacenada en los parámetros de sesión "UserCtxParam" y "GroupCtxParam" y redirige al usuario a la vista de sitio seleccionada.
- **Unidad de Cambio de Grupo**. Actúa como una unidad de acceso y de salida conjuntas, pero sin suprimir las informaciones de usuario y grupo (recibe como entrada un nombre de usuario y contraseña y redirige al usuario al módulo predeterminado conectado al grupo del usuario correspondiente).

Área de administración de WebRatio. La tabla Módulo almacena la lista de todos los módulos protegidos del Proyecto Web. Esto significa que cada vez que se cambia el modelo Web, añadiendo o eliminando módulos protegidos, es necesario agregar o quitar el registro correspondiente en la tabla del módulo. Estas operaciones pueden ser realizadas manualmente por el usuario o pueden realizarse a través del área de administración WebRatio. Para esto sirve la página o Área de administración de WebRatio, puesto que nos provee de los formularios que están protegidos y que deseamos agregar a la tabla de módulos.

Figura 64. Ejemplo de propiedades de proyecto encargadas de la configuración del área de administración de WebRatio

Es importante tener en cuenta que esta página no forma parte de su aplicación Web y, por lo tanto, ninguno de los usuarios de la aplicación puede acceder a ella. Es sólo un área de administración privada que le permite administrar módulos protegidos mientras está desarrollando la aplicación web. Esta área de gestión muestra:

- **El estado de implementación de la aplicación.** Aquí hay información sobre la aplicación Web, como la IP, el número de puerto, la ruta de la aplicación, el número de páginas y las unidades utilizadas.
- **La Configuración de Módulos Protegidos.** Aquí hay dos listas diferentes. La primera lista muestra los módulos protegidos presentes en el modelo Web pero aún no insertados en la tabla del módulo. La segunda lista muestra los

registros de la tabla de Módulos que no tienen el módulo protegido correspondiente en el Modelo Web, por lo que se pueden quitar.

5. Resultados

Luego de una evaluación exhaustiva de las dos herramientas y siguiendo un enfoque OO-Method como base de comparativa se obtuvieron los siguientes resultados:

	Integranova	**WebRatio**
Modelo de Datos	Permite conectar con cualquier base de datos. La conexión se configura en el momento de generar el código, fuera de los modelos conceptuales	Gran versatilidad de conexión, permitiendo conectar la herramienta con cualquier base de datos relacional disponible en el mercado.
Clases	Concepto claro de clase, maneja atributos de una forma clara con tipos de dato de los mismos que se adaptan a funcionalidades actuales.	Manejo de clases con la nomenclatura de Entidades, mayor versatilidad en el tipo de datos brindando mayor especificidad de los mismos. Las entidades tienen características de persistencias, pudiendo manejar entidades virtuales que no interactúan directamente con la base de datos.
Roles	Permite gama completa de relaciones, sin embargo, su interfaz es caduca y no se adapta a convencionalidades de usabilidad para el analista. No tiene una gestión de errores muy necesaria para conocer los posibles fallos que pudiera tener el modelo.	Interfaz intuitiva para generación de relaciones. Gama completa de cardinalidades. Gestión de errores amplia que permite identificar potenciales errores sintácticos.
Herencia	Soporta la herencia y permite a nivel de modelado definir una entidad relación. El padre comparte con sus hijos no solo sus atributos sino también sus eventos, transacciones y operaciones.	Soporta la herencia a nivel de modelado, tiene características de especialización de los hijos y comparte con ellos atributos, pero sus eventos deben ser modelados independientemente.
Atributos	Los atributos tienen nombres nemotécnicos, si se desea aplicar las etiquetas que serán utilizadas en la interfaz de usuario se las define en el sector de alias. Procesos realizados todos manualmente, creación de claves primarias y características propias de los atributos como permitir si los mismos soportan valores nulos.	Permite la utilización de caracteres especiales en la definición de nombres unificando el proceso de etiquetado para la interfaz de usuario. Automatización de procedimientos como la creación de claves primarias. Características como permitir valores nulos son embebidas directamente por el sistema y son modeladas, en caso de ser necesarias, directamente en el modelado de dominio con IFML.

Tipos de atributos	Los tipos de atributos son conocidos como constantes, variables y derivados.	Los tipos de atributos de una entidad se manejan automáticamente dando la característica de atributos constantes de una manera automática al crear la entidad y asignando directamente el primer atributo como constante que sería la clave primaria. En caso de necesitar más de un atributo constante se lo puede modelar con gran facilidad.
Tipos de datos de atributos	Los tipos de datos que soporta Integranova son genéricos, adaptados para ciertas bases de datos, en la modernas serían generalizados o embebidos en otro tipo de datos.	Tipos de datos más específicos y con gran capacidad de sincronización con cualquier tipo de base de datos relacional actual.
Atributos derivados	La capacidad de generación de atributos derivados es prácticamente manual. Pese a la existencia del asistente, es todavía un proceso complejo y difícil de comprensión.	Diferentes modalidades de atributos derivados correctamente agrupados y de una utilización sencilla e intuitiva. Permite una sincronización con la base de datos en caso de ser necesaria creando las vistas requeridas de una forma transparente al analista. Tiene la opción también de importar vistas generadas directamente en la base de datos permitiendo una transparencia entre el modelado y la capa de datos dando la oportunidad de reutilización de componentes previamente diseñados.
Servicios	Servicios de creación configurados automáticamente al momento de la definición del modelado de datos y con la posibilidad de configuración y modificación futuras. Tiene el problema de una interfaz poco intuitiva y con un asistente que generaconfusión. No reporta un sistema de mensajes de error a modo de guía y es una tarea ardua encontrar un posible error sintáctico.	Los servicios en WebRatio están categorizados en diversos componentes o “Unidades” en los que se encuentra incorporada la acción que desea ejecutar. Al manejarlos por separado y diseñarlos por medio de una nomenclatura validada por la OMG, permite una mayor versatilidad al momento de definir procesos más complejos partiendo de servicios más simples. Al igual que el modelado de datos, la parte de modelado IFML cuenta con un asistente y un buscador de errores que nos permite determinar de una

		mejor manera donde se encuentra el error de forma sintáctica. De igual manera tiene un debug para poder localizar errores semánticos que pudieran estar involucrados en el desarrollo de las diferentes operaciones.
Transacciones	Las transacciones en Integranova son complejas y tienen algunas directrices para poder ser comprendidas y ejecutadas correctamente. El orden es fundamental para el desarrollo adecuado de una transacción. Tiene características de todo o nada, esto quiere decir que, a nivel de transacciones, si falla algún componente todo el proceso cae y no realiza ningún cambio.	Las transacciones en WebRatio permiten una modularización de las mismas lo cual es una característica muy útil al momento de la reutilización de componentes y facilita la lectura de un modelado conceptual. Las transacciones en WebRatio siguen una secuencia, si alguno de sus componentes falla se ejecutarán los servicios hasta que haya existido un error. Es vital el manejo de excepciones al momento del desarrollo o pruebas constantes que, por medio de su generación incorporada de un servidor local, se lo puede realizar con gran facilidad.
Precondiciones	Las precondiciones pueden ser definidas a nivel de operación o a nivel de clase, estas tienen la opción de ser heredadas de una clase padre a una clase hijo. Integranova maneja dos tipos de condiciones, las precondiciones y las restricciones de integridad, las segundas no son muy recomendadas por el coste computacional que ralentiza al aplicativo web.	Las precondiciones tienen gran versatilidad para ser manejadas en WebRatio. Estas pueden ser configuradas con reglas de validación, condiciones de visibilidad o selectores. Esta versatilidad permite a WebRatio manipular las condiciones con las que se manejan las restricciones del aplicativo acorde sea necesario. Es decir, es mucho más específico que las precondiciones con las que interactúa Integranova.
Control de Agentes	El control de agentes y roles en Integranova es muy sencillo de configurar y emplear, es intuitivo a diferencia de gran parte de sus herramientas y permite establecer los permisos adecuados para el acceso a la información de la o las personas interesadas acorde al rol que posea.	El control de agentes en WebRatio es un tanto largo de configurar, sin embargo, una vez logrado, permite tener un control completo de los permisos de cada componente protegido en la definición del dominio. Este tipo de control permite al usuario dar mayor nivel de seguridad a su aplicativo en caso de que así lo requiriera, permitiéndole adoptar las políticas de seguridad que

		considere oportunas para su aplicativo e incorporarlas al modelo.

5.1. Respuestas a Preguntas de Investigación

Código	Pregunta de Investigación	Respuesta Obtenida
PI-1	¿Se puede considerar a la herramienta WebRatio como una integradora del paradigma orientado a modelos de manera que masifique los resultados y lo haga respetando las estructuras basadas en arquitecturas model driven?	Si, WebRatio es una herramienta que posee un enfoque MDD completo, el cuál ha sido complementado por medio de normativas aceptadas por la OMG y se ha encontrado, desde el momento de su creación, en constante crecimiento y evolución, enfrentando los nuevos retos que los mercados actuales presentan y adaptándose a las nuevas necesidades empresariales. Respeta de igual manera las estructuras basadas en arquitecturas Model-Driven asumiento así la categoría de herramienta capaz de satisfacer necesidades específicas por medio de la abstracción de sus contenidos.
SPI-1	¿Es posible considerar a WebRatio como una herramienta más potente y actual que Integranova para el desarrollo e implementación de proyectos basados en arquitecturas o paradigmas model driven?	Si, WebRatio ha presentado mejoras consistentes en aspectos de usabilidad y eficacia con respecto a la herramienta Integranova, denota una clara organización de manejo de contenidos y una mayor versatilidad de adaptación con las nuevas tecnologías existentes. De igual manera permite una gran facilidad al momento de realizar cambios en su código fuente y posee una amplia gama de herramientas que facilitan su conexión con aplicaciones existentes al igual que con motores de bases de datos establecidos.
SPI-2	¿Webratio es una herramienta que mantiene y respeta los lineamientos establecidos por la OMG para una implementación basada completamente en modelos?	Sí, al ser una herramienta enfocada en modelos ha buscado siempre su evolución para estar actualizada y sobretodo normatizada por organismos internacionales como la OMG, se denota su evolución desde WebML hasta llegar a nuevas tecnologías como iniciativas en BPM y desarrollo de aplicativos móviles, compitiendo con otras herramientas enfocadas a estos procesos como OOH.
SPI-3	¿WebRatio presenta mejoras o inconsistencias con respecto a	WebRatio presenta varias mejoras con respecto a otras herramientas basadas en modelos, tiene ciertas

	otras herramientas basadas en modelos para la gestión de implementaciones de software?	inconsistencias y hasta cierto punto sale de los parámetros establecidos de sujetarse a desarrollo de modelos establecidos en UML, sin embargo, estas actualizaciones han permitido obtener una herramienta más versátil que va de la mano con los nuevos cambios que se han presentado como retos en el enfoque Model-Driven.

6. Conclusiones y Trabajos Futuros

6.1. Conclusiones

Luego de la realización de la presente investigación se ha llegado a concluir que existen grandes vacíos en trabajos relacionados a las comparativas entre herramientas Model-Driven, si bien es cierto, existe una amplia bibliografía en lo referente a comparativas de las mencionadas herramientas con otros paradigmas de programación como el Orientado a Objetos, Funcional e Imperativa; sin embargo, existen grandes vacíos en la comparativa de las herramientas que el mercado posee actualmente y que se encuentran basados en enfoques orientados a modelos.

Se considera que la evolución de las herramientas y paradigmas permite ya a los investigadores realizar nuevos avances en esta materia, puesto que es una rama en la que existe gran capacidad de desarrollo e investigación, siendo considerada por muchos expertos como el futuro del desarrollo de aplicaciones.

En lo que respecta a la investigación se concluye también que los avances tecnológicos han permitido desarrollar nuevas herramientas de modelado de software y los retos a los que se enfrentaban anteriormente los investigadores, se han ido corrigiendo poco a poco, convirtiéndose así en nuevos entes de valor de las herramientas. Las problemáticas de usabilidad y modularización se han visto subsanadas en gran medida, gracias a la aparición de nuevos enfoques y técnicas que han permitido avanzar a una nueva era en los que refiere a MDD.

Integranova, como herramienta existente, cubre en gran parte las necesidades de un framework MDD, sin embargo, se ha visto desactualizada y el nivel de usabilidad de la misma con respecto a otras herramientas es bajo, sin embargo, mantiene su potencialidad de generar aplicativa web robustos. WebRatio, al ser una herramienta relativamente joven, mantiene esa mentalidad de reinventarse constantemente, no lo hace de una manera aleatoria sino basándose en estándares apoyados por la OMG y convirtiéndose hoy por hoy, en una herramienta de vanguardia en el diseño e implementación de aplicaciones industriales.

Las nuevas tendencias exigen a los desarrolladores de frameworks, facilidades e implementaciones que no solo se encuentren acorde a los nuevos dispositivos móviles sino también, que permitan una relación transparente con aplicaciones establecidas con anterioridad y sobre las cuales poseen un amplio universo de información y datos que no pueden perderse.

La migración de la información es una opción para este tipo de situaciones, sin embargo, hay estructuras de bases de datos tan grandes y complejas que resultaría un trabajo arduo la generación de un nuevo motor y diseño de base de datos para implementar la solución implementada en MDD. Es por este motivo que la factibilidad para relacionarse con nuevos modelos establecidos ya sea de una forma directa o por medio de servicios web es imprescindible para aplicativos diseñados para grandes corporaciones internacionales.

Finalmente, se observa también la necesidad de transparencia entre código fuente y modelo diseñado, esto permitirá establecer características cada vez más específicas, aunque el objetivo final sería no depender en absoluto del código. Esta transparencia facilitaría la migración de desarrolladores para convertirlos en analistas de modelos y permitir un crecimiento exponencial hacia este paradigma que ha evidenciado mejoras en lo referente al desarrollo industrial de aplicaciones web y actualmente móviles.

6.2. Trabajos Futuros

Como trabajos futuros se tiene una amplia gama en el área del enfoque Model-Driven, se ha incursionado en aspectos hacia donde se ha dirigido la tecnología como desarrollo móvil e Internet de las Cosas, sobre las cuales las herramientas han ofrecido una enorme gama de versatilidades enfocado a los desarrollos modernos.

Se pretendería realizar una documentación con miras una publicación acerca de la investigación realizada en este trabajo de fin de máster en la cual se denoten los resultados obtenidos, esto con miras a un mejoramiento en la calidad de ambas herramientas puesto que las mismas son una respuesta para un desarrollo ágil y con grandes respuestas a soluciones industriales y empresariales.

Se busca de igual manera incentivar a los investigadores a emprender nuevos esfuerzos en mayores estudios acerca de los avances de estas herramientas, promocionando su uso y evidenciando sus posibles falencias para así poder encontrar mejores soluciones y un cambio positivo y definitivo hacia este enfoque.

7. Referencias

[1] J. B. Quintero y R. Anaya, «MDA y el Papel de los Modelos en el Proceso de Desarrollo de Software,» *Revista EIA,* pp. 131-146, 2007.

[2] S. España Cubillo, *Methodological Integration of Communication Analysis into a Model-Driven Software Development Framework,* Valencia, Valencia: Universidad Politécnica de Valencia, 2012.

[3] C. Cuevas, L. Barros, P. Lopez Martínez y J. Drake, «MDE Technology as Support for Real-Time Systems Development Environments,» *Revista Iberoamericana de Automática e Informática Industrial RIAI,* pp. 216-227, 2013.

[4] A. Kepple, J. Warmer y W. Bast, «MDA explained: The Model Driven Architecture: Practice and Promise.,» 2003.

[5] R. France y B. Rumpe, «Model-Driven Development of Complex Software: A Research Roadmap. In Future of Software Engineering,» *ICSE '07,* p. 37–54, 2007.

[6] D. Wile, «Lessons Learned from Real DSL Experiments. In Proceedings of the 36th Annual Hawaii International Conference on System Sciences,» *HICSS,* p. 265–290, 2003.

[7] T. Greifenberg, K. Hölldobler, C. Kolassa, M. Look, P. Seyed Nazari, K. Müller, A. Navarro, D. Plotnikov, D. Reiss, A. Roth, B. Rumpe, M. Schindler y A. Wortmann, «A Comparison of Mechanisms for Integrating Handwritten and Generated Code for Object-Oriented Programming Languages,» *Model-Driven Engineering and Software Development (MODELSWARD),* 2015.

[8] B. Rumpe, Modellierung mit UML, Segunda Edición ed., Springer., 2011.

[9] B. Rumpe, Agile Modellierung mit UML : Codegenerierung Testfälle, Refactoring, Springer, 2012.

[10] B. Rumpe, M. Schindler, S. Völkel y I. Weisemöller, «Generative Software Development.,» *Proceedings of the 32nd International Conference on Software Engineering, ICSE '10,* p. 473–474, 2010.

[11] J. Ringert, B. Rumpe y A. Wortmann, «From Software Architecture Structure and Behavior Modeling to Implementations of Cyber-Physical Systems.,» *Software Engineering 2013 Workshopband,* p. 155–170, 2013.

[12] K. Krogmann y S. Becker, «A Case Study on Model-Driven and Conventional Software Development: The Palladio Editor,» *Software Engineering (Workshops),* pp. 169-176, 2007.

[13] J. Molina y O. Pastor, «MDA, OO-Method y la Tecnología OLIVANOVA Model Execution,» *Proceedings of I Taller sobre Desarrollos Dirigidos por Modelos, MDA y Aplicaciones,* pp. 1-9, 2004.

[14] M. Escalona y G. Aragón, «NDT. A Model-Driven Approach for Web Requirements,» *IEEE Transactions on Software Engineering,* pp. 377-390, 2008.

[15] J. Panach, S. España, O. Dieste, O. Pastor y N. Juristo, «In Search of Evidence for Model-Driven Development Claims: An Experiment on Quality, Effort, Productivity and Satisfaction,» *Information and Software Technology,* pp. 164-186, 2015.

[16] M. Brambilla, S. Ceri, E. Della Valle, F. Facca, C. Kubczak, T. Margaria, B. Steffen y C. Winkler, «Comparison: Mediation on WebML/WebRatio and jABC/jETI,» de *Semantic Web Services Challenge*, Boston, Springer, 2009, pp. 153-166.

[17] R. Rodríguez, Evaluación de la Usabilidad en Términos de Eficiencia, Eficacia y Satisfacción de la Herramienta INTEGRANOVA, Madrid: Universidad Autónoma de Madrid, 2016.

[18] D. Kolovos, R. Paige y F. Polack, «The Grand Challenge of Scalability for Model Driven Engineering,» *Models in Software Engineering,* pp. 48-53, 2008.

[19] D. Parnas, «On the criteria to be used in decomposing systems into modules,» *Communications of ACM 15(12),* p. 1053–1058, 1972.

[20] D. L. M. R. K. Hearnden, «Incremental model transformation for the evolution of model-driven systems,» *MoDELS 2006. LNCS,* pp. 321-335, 2006.

[21] D. P. R. P. F. Kolovos, «Detecting and Repairing Inconsistencies Across Heterogeneous Models,» *Proc. 1st IEEE International Conference on Software Testing, Verification and Validation,* p. 356–364, 2008.

[22] GWT Mail Application, «GWT Mail Application,» [En línea]. Available: http://code.google.com/webtoolkit/examples/mail/.

[23] R. Acerbis, A. Bongio, M. Brambilla, M. Tisi, S. Ceri y E. Tosetti, «Developing eBusiness Solutions with a Model Driven Approach: The Case of Acer EMEA.,» *7th International Conference on Web Engineering,* pp. 539-544, 2007.

[24] M. Brambilla y S. Butti, «Fifteen years of Industrial Model-Driven Development in software Front-Ends: From WebML to WebRatio and IFML,» *WebRatio,* 2014.

[25] S. Comai y P. Fraternali, «A Semantic Model for Specifying Data-Intensive Web Applications Using WebML,» *Semantic Web Workshop,* 2011.

[26] O. Pastor y J. Molina, Model-Driven Arquitecture in Practice. A Software Production Environmente Based on Conceptual Modeling, New York: Springer, 1998.

[27] O. Pastor y I. Ramos, «OASIS 2.1.1: A Class-Definition Language to Model Information Systems Using an Object-Oriented Approach,» *Servicio de Publicaciones UPV,* pp. 2,54,57,59,173,174,212,327, 1995.

[28] P. Letelier, I. Ramos, P. Sánchez y O. Pastor, «OASIS Versión 3.0: Un enfoque formal para el modelado conceptual orientado a objetos,» *Servicio de Publicaciones UPV,* pp. 54,57,59,174,212,301, 1998.

[29] P. Molina, Especificación de interfaz de usuario: De los requisitos a la generación automática, Valencia: Universidad Politécnica de Valencia, 2003.

[30] J. I. Panach, N. Juristo y Ó. Pastor, «Including Functional Usability Features in a Model-Driven Development Method,» *Computer Science and Information Systems,* pp. 999-1024, 2013.

[31] L. Bass y J. Bonnie, «Linking Usability to Software Architecture Patterns,» *The journal of systems and software,* pp. 187-197, 2003.

[32] J. Gómez, E. Insfrán, V. Pelechano y O. Pastor, «The Execution Model: a Component-Based Arquitecture to Generate Software Components From Conceptual Models,» *Proceedings of International Workshop on Component-Based Information Systems Engineering. 10th International Conference on Advanced Information Systems Engineering, CAiSE-98,* pp. 87-94, 1998.

[33] J. Gómez, O. Pastor, E. Insfrán y V. Pelechano, «From Object-Oriented Conceptual Modeling to Component Based Development,» *Database and Expert Systems Applications,* pp. 332-341, 1999.

[34] R. Acerbis, A. Bongio, S. Brambilla, S. Ceri y P. Fraternali, «Web Applications Design and Development with WebML and WebRatio 5.0,» *TOOLS,* pp. 392-411, 2008.

[35] I. Manolescu, M. Brambilla, S. Ceri, S. Comai y P. Fraternali, «Model-Driven Design and Deployment of Service-Enabled Web Applications.,» *ACM TOIT,* p. 5:3, 2005.

[36] M. Brambilla, S. Ceri, P. Fraternalli y I. Manolescu, «Process Modeling in Web Applications.,» *ACM TOSEM,* p. 15:4, 2006.

[37] S. Ceri, F. Daniel, M. Matera y F. Facca, « Model-driven Development of Context-Aware Web Applications,» *ACM TOIT,* p. 7:1, 2007.

[38] M. Brambilla, I. Celino, S. Ceri, D. Cerizza y E. Della Valle, «Model-Driven Design and Development of Semantic Web Service Applications.,» *ACM TOIT,* p. 8:1, 2008.

[39] S. Bozzon, S. Comai, P. Fraternali y G. Toffetti Carughi, «Conceptual Modeling and Code Generation for Rich Internet Applications,» *International Conference on Web Engineering,* pp. 353-360, 2006.

[40] M. Brambilla y P. Fraternali, «The Interaction Flow Modeling Language (IFML),» 03 2014. [En línea]. Available: www.ifml.org.

[41] M. Brambilla y P. Fraternali, Model-Driven UI Engineering of Web and Mobile Apps with IFML, Massachusetts, USA: Elsevier, 2015.

[42] WebRatio, «WebRatio - Entorno de desarrollo rápido para aplicaciones Móviles y Web,» 29 Noviembre 2013. [En línea]. Available: http://my.webratio.com/learn/learningobject/domain-model-overview?inu1k.current.att1u=21&link=ln231x&fllbck=.sv2&cbck=wrReq64933.

[43] WebRatio, «WebRatio - Entorno de desarrollo rápido para aplicaciones Móviles y Web,» 13 Marzo 2014. [En línea]. Available: http://my.webratio.com/learn/learningobject/extending-the-domain-model-derivation-v-72?link=oln72ae.redirect&pcp1x=extending-the-domain-model-derivation-v-72&nav=14&cbck=wrReq28229.

[44] OMG, Interaction Flow Modeling Language (IFML) Official OMG Specification, OMG, 2014.

[45] M. Frigerio, «WebRatio,» 14 05 2012. [En línea]. Available: http://my.webratio.com/learn/learningobject/connect-unit?link=oln72ae.redirect&pcp1x=connect-unit&nav=14&cbck=wrReq37784.

[46] A. Molinaroli, «WebRatio,» 14 05 2012. [En línea]. Available: http://my.webratio.com/learn/learningobject/disconnect-unit?link=oln72ae.redirect&pcp1x=disconnect-unit&nav=14&cbck=wrReq68564.

[47] M. Bruno, «WebRatio,» 14 05 2012. [En línea]. Available: http://my.webratio.com/learn/learningobject/reconnect-unit?cbck=wrReq4816&link=oln72ae.redirect&pcp1x=reconnect-unit&nav=14.

[48] WebRatio, «WebRatio - Entorno de desarrollo rápido para aplicaciones Móviles y Web,» 21 Marzo 2014. [En línea]. Available: http://my.webratio.com/learn/learningobject/reusable-models-modules-module-definitions-v-72?link=oln72ae.redirect&pcp1x=reusable-models-modules-module-definitions-v-72&nav=14&cbck=wrReq78223.

[49] WebRatio, «WebRatio - Entorno de desarrollo rápido para aplicaciones Móviles y Web,» 21 Marzo 2014. [En línea]. Available: http://my.webratio.com/learn/learningobject/reusable-models-modules-module-definitions-v-72?link=oln72ae.redirect&pcp1x=reusable-models-modules-module-definitions-v-72&nav=14&cbck=wrReq78223.

[50] L. Bordonaro, «WebRatio - Entorno de desarrollo rápido para aplicaciones Móviles y Web,» 24 Octubre 2011. [En línea]. Available: http://my.webratio.com/learn/learningobject/getting-started-with-visibility-conditions-v-72?link=oln72ae.redirect&pcp1x=getting-started-with-visibility-conditions-v-72&nav=14&cbck=wrReq30819.

[51] Eclipse, «Eclipse Modeling Project,» 2008. [En línea]. Available: http://www.eclipse.org/modeling/emf/.

[52] Eclipse, «Eclipse Modeling - EMFT,» 2008. [En línea]. Available: http://www.eclipse.org/modeling/emft/?project=cdo.

[53] Eclipse, «Eclipse Modeling - EMFT,» 2008. [En línea]. Available: http://www.eclipse.org/modeling/emft/?project=teneo.

[54] Eclipse, «Eclipse - Graphical Modeling Framework,» 2014. [En línea]. Available: http://www.eclipse.org/gmf-tooling/.

8. Anexos

8.1. Desarrollo Práctico de Ejercicio

8.1.1. Ejercicio planteado. Parte 1

8.1.1.1. Introducción

El sistema de gestión de informes de gastos es una herramienta que ayuda al proceso de informes de gastos a seguir. Utilizando esta herramienta, es fácil seguir el ciclo de vida de los informes de gastos, desde su creación hasta que se paga al empleado.

Los empleados dan en sus informes de gastos cuando todos los billetes relacionados con viajes de negocios o un almuerzo de negocios hayan sido ingresados.

Si el empleado recibió un pago por adelantado, éste debe ser ingresado cuando se crea el informe de gastos. Además, cada gasto se especificará en una línea. El pago anticipado debe ser restado del gasto total que se pagará al empleado.

Cuando se emite un Reporte de Gastos, éste debe ser revisado por el Gerente de Departamento. El mismo decidirá si el informe de gastos está autorizado y continúa su flujo. Pero puede denegar el informe y especificar el motivo por el cual no autorizó el informe de gastos.

Antes de pagar un informe de gastos, el pago debe ser aprobado por el Gerente de Cuentas. También puede negar el pago, pero en este caso, debe indicar la razón por la cual se le niega.

Si un Gerente de Departamento o Gerente de Cuentas niega el informe de gastos, entonces el empleado debe modificarlo, dependiendo de la razón dada por los gerentes.

Después de que el empleado modifique el informe de gastos, debe volver a emitirlo para iniciar el proceso.

Finalmente, cuando se aprueba un informe de gastos, un usuario de Cuentas o el Administrador de Cuentas puede pagarlo. A continuación, el estado del informe de gastos se "pagará" y se mantendrá la fecha de pago.

8.1.1.2. Definición de datos del problema

Reporte de gastos

- **Encabezado y pie de página:** información general sobre los gastos. Recoge todos los gastos relacionados para un proyecto y un empleado.
- **Identificador del informe.** Debe ser único.
- **Fecha de creación del informe de gastos.**

- **Empleado que entrega el informe**
- **Proyecto relacionado con estos gastos**
- **Breve descripción del motivo de los gastos.**
- **Pago por adelantado**
- **Total de gastos**: suma de todos los gastos
- **Saldo (Gastos totales - Anticipo).** Si el saldo es superior a 0 y el tipo de pago contiene un coste adicional, este valor se restará. En caso de que después de restar el coste adicional, el resultado de la operación sea negativo, entonces el balance será cero.
- **Estado actual del informe de gastos.** Puede ser:
  - 0-Abierto
  - 1-Enviar
  - 2-Autorizado
  - 3-Aprobado
  - 4-Pagado
- **Fecha de aprobación.**
- **Fecha de pago y tipo de pago.**
- **Razón por la cual el informe de gastos no está autorizado** (opcional).
- **Razón por la cual el informe de gastos no es aprobado** (opcional).
- **Comentarios relacionados con el pago** (opcional).

Línea de Gastos

Cada gasto se especifica en una línea.

Datos a almacenar:

- **Número de línea**: cada informe de gastos tendrá su propio número de lista
- **Fecha de gastos**
- **Tipo de gastos:** (Kilómetros, alquiler de coches, billete de avión, factura de almuerzo, etc.)
- **Unidades:** Número de elementos definidos como gasto. Por ejemplo, kilómetros, número de factura del almuerzo, etc. Su valor predeterminado es 1
- **Precio:** Representa el precio por unidad. Se asigna en función del tipo de gasto.
- **Descripción de gastos:** una breve descripción sobre el gasto. Por ejemplo:
  "Billete de avión de Barcelona a Munich" o
  "Ticket de almuerzo en el Restaurante Gourmet con clientes (CHG Company)
- **Gasto total:** precio * unidades

Tipo de Gasto

Un tipo de gasto contiene todo tipo de gastos posibles que se pueden agregar a un informe de gastos. Por ejemplo, alquilar un coche, billete de gasolina, etc.

- **Código de tipo de gasto.** Longitud máxima 4 caracteres.
- **Breve descripción sobre el tipo de gasto**
- **¿Es un precio fijo de la empresa?**
- **Precio:** fijo / recomendado.

Tipo de Pago

Contiene todas las formas posibles con las que la empresa pueda devolver el dinero gastado por la misma.

El tipo de pago se puede cambiar durante el proceso de informe de gastos.

No es obligatorio vincular un tipo de pago cuando se crea un informe de gastos. Se puede vincular durante el proceso de informe de gastos. Pero antes de pagar un informe de gastos, debe estar relacionado con un tipo de pago.

La información relacionada con un tipo de pago es:

- **Código de tipo de gasto:** Debe ser único. Longitud máxima 4 caracteres.
- **Breve descripción sobre el tipo de gasto**
- **¿Cargo adicional?**: Algunos tipos de pago tienen cargos adicionales que disminuyen la cantidad que se debe pagar al empleado. De forma predeterminada, la cantidad es cero.
- **Porcentaje de cargo adicional:** Este campo indica si la cantidad como cargo adicional es una cantidad simple o un porcentaje del monto total del informe de gastos

Departamento

Un empleado puede ser un miembro de un solo departamento. Cada departamento tiene un gerente que es un empleado de este departamento.

Código de departamento: Debe ser único. Longitud máxima 4 caracteres.

Nombre de Departamento

Gerente de departamento

Número de empleados activos: Que pertenecen a este departamento

Proyecto

Un proyecto puede ser desarrollado por un grupo de empleados. De la misma manera, un empleado puede trabajar en varios proyectos. Un proyecto es administrado por el Gerente del Departamento y puede agregar o remover empleados del grupo de trabajo.

- **Código del proyecto:** Debe ser único. Longitud máxima 4 caracteres.
- **Nombre del proyecto**
- **Número de empleados:** Que se encuentran trabajando en el proyecto
- **Total de informes de gastos:** Que estén relacionados con el proyecto

Empleado

Este es el conjunto de todos los empleados pertenecientes a la empresa.

- **Código del empleado:** Debe ser único. Longitud máxima 4 caracteres.
- **Nombre y apellido**
- **Dirección, teléfono y correo electrónico**
- **Departamento**
- **¿Es gerente de departamento?**
- **Despedido** (ahora, él no está trabajando para esta empresa)

Gerente de Departamento

Un empleado puede ser un Gerente de Departamento y tendrá una funcionalidad adicional. Para cada departamento, no es posible tener más de un gerente. Cuando un empleado que es Gerente de Departamento es trasladado a otro departamento, él será un empleado normal en el nuevo Departamento.

- **Código del Empleado:** Debe ser único. Longitud máxima 4 caracteres. Se recomienda usar las iniciales del empleado.
- **Departamento que es el Gerente.**

Usuarios

Los usuarios no pertenecen al grupo de empleados.

- **Código de usuario:** Debe ser único. Longitud máxima 4 caracteres.
- **Nombre y apellido**
- **Email** (opcional)

Dependiendo del nivel de acceso, un usuario puede ser especificado como:

- Usuario de Cuentas
- Gerente de Cuentas
- Administrador

8.1.1.3. Descripción del Ejercicio 1

Durante el primer ejercicio, se define la parte estática de su modelo. La parte estática de un modelo debe contener:

- Clases
- Atributos
- Relaciones estructurales (agregación y herencia)

Para los atributos derivados, será necesario definir las fórmulas de derivaciones correspondientes.

Además, si se definen las relaciones de herencia, entonces será necesario añadir, el evento transportista correspondiente y el evento liberador (opcional).

8.1.2. Ejercicio Planteado. Parte 2

8.1.2.1. Introducción

La funcionalidad de gestión de informes de gastos se incluirá en esta práctica. Esta funcionalidad se caracteriza por:

- Proceso de informe de gastos (proceso básico y principal de la aplicación)
- Ciclo de vida de un empleado en la empresa
- Gestión de proyectos
- Gestión del departamento

- Modo de pago y tipo de gestión de gastos
- Gestión de usuarios

8.1.2.2. Proceso del Reporte de Gastos

El sistema se utiliza una vez que el empleado reúne toda la documentación (billetes, boletos, etc.) relacionada con sus gastos que se han realizado durante una actividad laboral (viajes, comida, compras...) y quiere que se les reembolse.

El empleado puede haber pedido algo de dinero por adelantado antes de hacer cualquier pago. Esta cantidad de dinero dada por adelantado se reflejará en el informe de gastos en el momento de la creación.

Una vez que se ha creado el informe de gastos, se agregará una línea de gastos por cada gasto que el empleado haya realizado. También será necesario un comentario explicando cada gasto.

Cuando se agrega una línea de gastos, su precio dependerá del tipo de gasto relacionado. Algún tipo de gastos tendrá un precio fijo y su importe se calculará sobre la cantidad de elementos (por ejemplo: € * km, dieta * días, etc.). El resto de los gastos no tendrá un precio fijo por lo que su cantidad dependerá de otros factores. En estos casos, un precio recomendado puede ser dado de manera predeterminada, aunque puede ser modificado por el empleado, lo que le permite introducir el precio exacto del gasto realizado (por ejemplo: un billete de tren).

Una vez introducida la información, el sistema asignará al informe de gastos el estado de "Abrir". Al estar en este estado, el informe de gastos puede ser modificado, puede tener nuevas líneas añadidas o eliminadas, incluso se puede eliminar por completo. Sin embargo, cuando el empleado emite el informe de gastos, el estado de los gastos se convertirá en "Emitido" y no podrá ser modificado en absoluto. En ese momento el informe de gastos está pendiente de ser autorizado.

Un Gerente del Departamento tiene que verificar que los Informes de Gastos son consistentes con el trabajo que un Empleado ha sido asignado. Por lo tanto, cuando se expide un Informe de Gastos, el Gerente de Departamento tendrá que autorizarlo para que el proceso pueda ser seguido. Después de esta autorización, los boletos y las facturas proporcionados por el empleado deberán ser verificados por el Gerente de Cuenta para que el Informe de Gastos pueda ser aprobado y listo para recibir el pago.

Si el informe de gastos está autorizado, su estado se convierte en "Autorizado", guardando la fecha del sistema en este momento. Una vez alcanzado este estado el informe de gastos estaría pendiente de ser aprobado. Si el informe de gastos no está autorizado, debe darse una razón de denegación y el estado de los gastos se convertirá en "No Autorizado". Con este estado el informe de gastos se puede modificar como lo hace cuando su estado es "Abierto". Cuando el empleado hace los cambios apropiados para hacer que el informe de los gastos sea autorizado, él / ella tendrá que emitir el reporte de gastos de nuevo.

Una vez aprobado el pago, el informe de gastos se establecerá como "Aprobado". En caso de que algo esté mal, se convertirá en "Desaprobado". En este estado el informe de gastos

se puede modificar como lo hace cuando su estado es "Abierto". Cuando se hacen los cambios apropiados, el informe de gastos debe ser emitido de nuevo para seguir el proceso.

Cuando el informe de gastos es aprobado, puede ser pagado. Una vez realizado el pago, el estado de los gastos se convierte en "Pago", ahorrando la fecha de pago y forma de pago (efectivo, transacción bancaria ...).

Cuando se paga un informe de gastos y el dinero avanzado excede los gastos totales, se agregará una sentencia notificando al empleado que tiene 15 días para devolver la diferencia entre el dinero avanzado y los gastos totales en los comentarios del Informe de gastos.

8.1.2.3. Ciclo de vida del empleado en la empresa

Posibilidad de crear nuevos empleados ingresando toda la información requerida.

Todos los datos de los empleados pueden ser modificados por el mismo servicio.

Si un empleado deja la empresa, se marcará como "Despedido". Él / ella será removido del proyecto / s que él / ella perteneció a, sin embargo el empleado no será borrado. Si el empleado era un departamento responsable, él / ella no será más el departamento responsable.

Un empleado puede ser cambiado de un departamento. Sin embargo, si el empleado era responsable de un departamento, no podrá ser cambiado a menos que él / ella fue despedido previamente.

Debe estar disponible un servicio para eliminar empleados. Si se elimina un empleado, solo si todos sus informes de gastos tienen un año de antigüedad, podrían eliminarse con su Empleado, de lo contrario, el Empleado no podría ser eliminado.

Un empleado puede ser promovido como responsable de un departamento. Si el mismo departamento tuviera ya un responsable, él / ella tendría que ser automáticamente despedido y entonces el otro empleado sería promovido.

8.1.2.4. Director de Proyectos

Posibilidad de crear nuevos proyectos.

Posibilidad de asignar empleados a proyectos existentes.

8.1.2.5. Director de Departamento

Pueden ser creados, modificados o eliminados por el administrador del sistema

Un Administrador puede cambiar el Gerente de Departamento.

8.1.2.6. Modo de pago y tipo de gestión de gastos

Ambos modos de pago y tipo de gastos pueden ser creados, modificados o borrados en el sistema.

8.1.2.7. Manejo de Usuarios

Posibilidad de agregar nuevos usuarios de cuentas, responsables de cuentas y administradores de sistemas.

Posibilidad de modificar o eliminar datos del sistema.

8.1.2.8. Descripción del Ejercicio 2

En la práctica 2, sólo se definirán los servicios necesarios (eventos y transacciones locales) para cubrir el proceso de informe de gastos arriba explicado y la gestión de los diferentes elementos.

Además, para cubrir toda la funcionalidad, todas las valuaciones de eventos y fórmulas de transacción tendrán que ser definidas.

8.1.3. Ejercicio Planteado. Parte 3

8.1.3.1. Introducción

Durante este ejercicio se van a definir las interfaces para cada perfil de agente existente en el sistema.

Además, se definen las restricciones del sistema para poder seguir correctamente el ciclo de vida del informe de gastos y mantener la integridad de la información de la base de datos.

Por último, se definirá el ciclo de vida del informe de gastos. Será necesario recordar la funcionalidad de los últimos ejercicios.

8.1.3.2. Definición de Perfiles

Auditor

Se debe permitir que un Auditor de sesión inicie sesión en la aplicación sin tener definido un usuario ni una contraseña.

Un auditor debe tener visibilidad sobre toda la aplicación para acceder a toda la información necesaria sin modificar ningún dato.

Líneas de Gastos y Manejo de Reportes

Cuando un empleado hace un informe de gastos, este informe se asignará automáticamente al empleado y no se le pedirá al empleado como argumento entrante.

El Administrador para que el Departamento de Gerentes pueda crear Informes de Gastos, pero, en este caso, tendrán que agregar el Empleado a quien se le ha asignado, como argumento de entrada.

Un empleado puede crear, modificar y eliminar sus informes siempre y cuando estén en estado "abierto".

El Gerente del Departamento será la persona encargada de autorizar o rechazar los informes de gastos.

El Gerente de Cuentas será quien aprueba o rechaza el pago del reporte de gastos.

Los Informes de gastos pueden ser pagados por los Usuarios y los Administradores de Cuentas (aunque hay casos especiales para los usuarios: consulte la sección de restricciones).

Manejo de Empleados

La gestión de los empleados sólo puede ser llevada a cabo por el Gerente del Departamento o el Administrador del Sistema.

Si un Gerente de Departamento crea un empleado, será asignado automáticamente al departamento donde pertenece el Gerente de Departamento. Pero en el caso de que el Administrador cree al empleado, el departamento podrá ser elegido.

Sólo el Administrador podrá:

- Cambiar a un empleado del departamento.
- Promover o despedir a un empleado de / hacia el Gerente del Departamento.
- Elimine un empleado.

Manejo de Proyectos

Sólo pueden ser administrados por un Administrador o un Gerente de Departamento.

Los empleados sólo pueden ver los proyectos en los que participa.

Manejo de Departamentos

La administración del departamento es llevada a cabo por los Administradores.

La información de un departamento puede ser visualizada por sus empleados, su Gerente de Departamento, todos los usuarios y todos los Gerentes de Cuenta.

Tipos de Gasto y formas de pago

El Administrador los crea, modifica o elimina, pero son visibles para todos los usuarios del sistema.

Manejo de Usuarios

La gestión de usuarios de Cuentas es llevada a cabo por los Gestores de Cuentas o el Administrador.

La información sobre los usuarios de Cuentas sólo puede mostrarse a los Administradores de Cuentas, a los Administradores y al Usuario de Cuentas.

La información sobre los administradores de cuentas sólo se puede mostrar al administrador de cuentas mismo o los administradores.

Los Administradores de Cuentas y la Administración de Administradores sólo pueden ser realizados por un Administrador. Pero un administrador de cuentas podrá modificar sus propios datos.

Restricciones de la Aplicación

Un empleado no puede tener más de 2000 € en pagos anticipados en todos sus Informes de Gastos que no se pagan, los Informes de Gastos que exceden esta cantidad de dinero en pagos anticipados no pueden ser creados. (Tal vez un nuevo atributo es necesario para cumplir este requisito).

Una Línea de Gastos nunca tendrá un valor negativo.

Si un informe de gastos no está en estado "abierto", no puede modificarse o eliminarse (pero si el agente es el administrador).

Un informe de gastos puede ser "Emitido" si está en estado "Abierto".

Un informe de gastos sólo puede ser autorizado o no si se encuentra en estado "Emitido".

Un informe de gastos sólo puede aprobarse o no si se encuentra en estado "Autorizado".

Un usuario de cuentas puede pagar el informe, siempre y cuando el informe sea aprobado y el saldo no exceda de 3000 €.

Si un Empleado es Gerente de Departamento, el Empleado no puede cambiar de Departamento a menos que deje de ser Gerente de Departamento.

Al promocionar un Empleado al Gerente de Departamento, si hay un Gerente de Departamento anterior para este Departamento, es necesario despedir a este Gerente de Departamento antes de promover el nuevo.

Si un empleado ha sido despedido, no será posible asignar nuevos informes de gastos a este empleado.

Si un empleado ha sido despedido, no puede ser promovido a Gerente ni cambiado de departamento.

Ciclo de Vida del Reporte de Gastos

A continuación, se explica el proceso que sigue el informe de gastos:

Cuando se introducen datos, el sistema asigna al informe de gastos el estado "Abierto". Mientras que el estado es "Abierto", el informe de gastos puede ser modificado, y las nuevas líneas se pueden agregar o eliminar el informe de gastos con todas sus líneas. Sin embargo, cuando el empleado emite el informe de gastos, no puede ser modificado; Cambia al estado "Emitido". Ahora el informe de gastos está esperando la autorización.

El Gerente del Departamento se encarga de autorizar o no los informes de gastos. Si se autoriza un informe de gastos, se convierte en el estado "Autorizado" y estará a la espera de ser aprobado. Si no es así, el informe de gastos se convierte en "Abrir". En este estado, el informe de gastos puede modificarse de nuevo. Cuando un empleado ha realizado los

cambios necesarios para obtener el informe de gastos autorizado, vuelve a "Emitir" el informe de gastos, por lo que comienza de nuevo su proceso desde el estado "Emitido".

El Gerente de Cuentas será el que autorice o no los pagos de los informes de gastos. Cuando se aprueba el pago, el informe de gastos se verifica como "Aprobado". Si hay algo que no está siguiendo la normativa, el gerente de Cuentas no aprobará el informe de gastos y el estado del informe de gastos será "Abierto". En este nuevo estado, el informe de gastos puede modificarse de nuevo. Cuando los cambios necesarios para ser aprobados son realizados por el empleado o por el Gerente del Departamento, emitir el informe de gastos de nuevo por lo que comienza su proceso desde el estado "Emitido".

Cuando el informe de gastos ha sido aprobado por el Gerente de Cuentas, cualquier miembro del departamento de Cuentas puede pagar el informe de gastos, pero sólo si el saldo de gastos es inferior a 3000 €. En este caso, sólo el administrador de cuentas puede pagarlo. Cuando se ha pagado el pago, el informe de gastos se marca como "Pago".

Cuando se ha pagado un informe de gastos, sólo el administrador puede eliminar y si y sólo si ha pasado un año después de la fecha de pago.

8.1.3.3. Descripción del Ejercicio 3

En el ejercicio 3 se va a definir:

- Los diferentes perfiles que se pueden introducir en la aplicación y sus permisos de visibilidad, ejecución y navegación (interfaces de agentes).
- Las limitaciones en la funcionalidad de la aplicación.
- Defina el diagrama de transición de estado sólo para la clase de informe de gastos. En este caso, redefinir o eliminar las condiciones previas creadas en el último ejercicio
- Preste atención a la definición de perfiles, ya que, en algunos casos, algunas funcionalidades deben estar encapsuladas, dependiendo del agente conectado.