UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

Validación del uso de métodos de producción de software de requisitos a código dirigidos por modelos.

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: José Vicente Rico Bellot

Tutor: Óscar Pastor López

Curso 2019/2020

Resumen

Esta tesis pretende estudiar y analizar las herramientas Great (Ruiz, 2018) e IntegraNova (IntegraNova, 2020) y la viabilidad del uso de ambas como forma de producir software. El método de producción se basa en modelos conceptuales de requisitos usando el método de Análisis Comunicacional, y en su transformación a un esquema conceptual del sistema siguiendo el OO-METHOD, para finalmente generar el código del software en forma automática. Para ello se realizará un estudio del funcionamiento de las tecnologías en las que se basan Análisis Comunicacional (GREAT) y OO-Method (IntegraNova) con su documentación plasmada en esta tesis. Tras aportar la documentación de las tecnologías y las herramientas se procederá a realiaar un caso de uso práctico que constará de un ejemplo guiado en el que se partirá de un sistema de información real, el cual será modelado con Great en análisis comunicacional y se realizará el proceso de transformaciones pertinente hasta llegar a convertirlo en el código equivalente que sirva para poder manejar ese sistema de información. Todo este proceso será documentado en detalle, ayudando a la comunidad de modelamiento a tener esta documentación disponible para cuando quieran investigar el funcionamiento de las herramientas, debido a que son herramientas experimentales y no constan de demasiada información sobre ellas. Al final de la tesis se evaluarán los resultados obtenidos en las diferentes fases del proyecto y se realizarán unas valoraciones para reflejar cuales son los problemas y las oportunidades que nos ofrece esta forma de producir código ante las formas de programar tradicionales.

Palabras clave: Análisis comunicacional, OO-Method, Software dirigido por modelos, Great, IntegraNova

Abstract

This thesis aims to study and analyze the Great (Ruiz, 2018) and IntegraNova (IntegraNova, 2020) tools and the feasibility of using both as a way of producing software. The production method is based on conceptual models of requirements using the Communicational Analysis method, and on its transformation to a conceptual scheme of the system following the OO-METHOD, to finally generate the software code automatically. To this end, a study of the operation of the technologies on which Communicational Analysis (GREAT) and OO-Method (IntegraNova) are based will be carried out with its documentation reflected in this thesis. After providing the documentation of the technologies and tools, a practical use case will be carried out that will consist of a guided example that will start from a real information system, which will be modeled with Great in communication analysis and the process of pertinent transformations until it is converted into the equivalent code that is used to be able to manage that information system. This entire process will be documented in detail, helping the modeling community to have this documentation available for when they want to investigate the operation of the tools, since they are experimental tools and do not contain much information about them. At the end of the thesis, the results obtained in the different phases of the project will be evaluated and some evaluations will be made to reflect what are the problems and opportunities that this way of producing code offers us compared to the traditional ways of programming.

Keywords: Communicational Analysis, OO-Method, Model Driven Software, Great, IntegraNova

Tabla de contenidos

1. Introducción

1.1. Motivación

En la actualidad el mundo está repleto de aparatos electrónicos que facilitan las tareas del día a día. Cuando utilizamos nuestro ordenador o el móvil, hay una infinidad de aplicaciones, programas y webs que pretenden satisfacer las necesidades que tenemos en ese momento. Para que estas utilidades puedan ser construidas y garantizar que funcionen correctamente hay un equipo que se encarga del desarrollo y de la actualización del producto.

En campos como el diseño, la edición de video o la producción musical existen herramientas que de forma sencilla e intuitiva permiten al usuario la construcción de aquello que deseen, ya sea componer una canción o diseñar una pieza. Estas herramientas parten desde el nivel más básico, donde prácticamente cualquier usuario puede realizar la mínima creación hasta niveles más altos de dificultad, donde un usuario formado y experto puede realizar piezas más complejas.

En el campo de la programación no existen herramientas capaces de proporcionar estas facilidades. Dentro de la programación existe una gran variedad de lenguajes, en su mayoría basados en unas reglas comunes, que depende de qué se pretenda crear tienen unas ventajas y unos inconvenientes. Para el uso de estos lenguajes por regla general no existe un programa concreto donde se utilizan, si no que existen diferentes IDE (Entornos de desarrollo integrado) donde podemos crear proyectos de diferentes lenguajes. Algunos IDE relevantes pueden ser Eclipse, Netbeans o Visual Studio.

Algo que tienen en común los lenguajes de programación es que como indica su nombre, son lenguajes, y por tanto la forma de construcción de un programa se generará a partir de líneas de código que el creador ha escrito. Al contrario de las herramientas de otras áreas nombradas previamente, para poder empezar a programar, aunque sea de la forma más básica, se debe saber en qué lenguaje se quiere programar y los puntos básicos de ese lenguaje.

Asimismo, podemos apreciar una problemática con la barrera de conocimientos continua que existe en la programación. Usualmente, cuando te enfrentas a un nuevo problema, aparecen estrategias para resolverlo que no conoces, y que no siempre disponen de abundante documentación donde aprender a resolverlo.

Se puede percibir que hay un problema con la complejidad que oponen los lenguajes de programación actuales a poder plasmar las ideas en el código. Por lo tanto, hay un amplio margen de avance tanto para abrir el mundo de la programación a usuarios menos cualificados, como para facilitar su trabajo a aquellos que decidan dedicarse a la programación.

1.2. Objetivos

En este trabajo de fin de grado se pretende evaluar en qué punto se encuentra la generación de código de forma automática, esta evaluación se realizará mediante la combinación del uso de una herramienta de análisis comunicacional, como es Great, y una herramienta de OO-Method, que es IntegraNova.

Esta tesis se compone de dos partes principales:

- Por una parte, se realizará un desarrollo de un caso práctico documentado, que, partiendo de un ejemplo, avance los diferentes niveles de abstracción de la arquitectura de *software* dirigida por modelos hasta llegar a una implementación a código mediante el uso de las herramientas nombradas anteriormente. Este caso práctico dedicará la mayoría de sus esfuerzos en garantizar que se abarcan la mayoría de las opciones de generación que Great nos ofrece.

Al contrario que IntegraNova, que es un programa popular y completo, Great es un programa de uso para investigación y en desarrollo. Este es el motivo por el que la evaluación se centra en Great, porque hay un menor conocimiento de que es capaz de hacer. Por consiguiente, es importante ofrecer nueva información que permita seguir progresando en la investigación, de esta forma, todo el proceso estará documentado dentro de esta tesis facilitando futuros proyectos y tesis que pretendan utilizar el programa.

- Por otra parte, a partir de los resultados obtenidos en el punto anterior se realizará una evaluación para observar las opciones que Great nos ofrece respecto a todas las posibilidades que existen en el análisis comunicacional. Como Great es un programa prácticamente pionero, la valoración de los resultados se realizará en función de dos objetivos principales.

En primer lugar, se realizará una valoración de la adaptación del análisis comunicacional a la herramienta. De esta forma, la evaluación se realizará mediante reglas establecidas en investigaciones previas de análisis comunicacional y se adaptará a las utilidades que nos ofrece la herramienta Great. Así se ofrecerá una visión de los puntos fuertes de la herramienta y de aquello que tiene margen de mejora o de revisión.

1.3. Preguntas de investigación

Para poder obtener las mejores preguntas de investigación del objeto de estudio de esta tesis hay que analizar los objetivos planteados en apartados anteriores. Por consiguiente, al igual que es interesante dividir los objetivos del trabajo en dos partes, lo más adecuado es analizar ambos objetivos por separado y obtener las incógnitas que pretendemos resolver al final del proyecto.

Como aparece en apartados anteriores, uno de los objetivos de la tesis es aportar documentación del proceso de derivación desde un caso práctico planteado en texto hasta la implementación a código, con las transformaciones realizadas por Great e IntegraNova. Así pues, estas son las cuestiones planteadas:

- ¿Es posible automatizar el proceso de programación para poder generar código a partir del diagrama de análisis comunicacional diseñado en Great, también llamado diagrama de eventos comunicativos?
- ¿Un usuario sin conocimientos de programación, pero con algunos conocimientos de análisis comunicacional sería capaz de generar código?
- ¿La documentación que aporta esta tesis es suficiente para que alguien sin conocimientos sea capaz de realizar el proceso?
- ¿Esta tesis proporciona conocimientos nuevos a las materias de análisis comunicacional y OO-Method?

La otra parte de la tesis y objetivo de esta es el análisis de los resultados obtenidos en la tesis, por tanto, se plantean las siguientes preguntas:

- ¿Cuáles son los beneficios de utilizar este proceso?
- ¿Qué brechas funcionales hay actualmente realizando este proceso?
- ¿Vale la pena dejar usar este proceso a la implementación de código tradicional?

1.4. Impacto esperado

Este trabajo de final de grado está enfocado a la investigación, de esta forma, tanto su composición como las valoraciones finales que se consigan extraer serán beneficiosas para la comunidad de Modelamiento Conceptual de Sistemas de Información. Al igual que este trabajo de final de grado tiene dos objetivos principales, los principales beneficiados de la realización de esta tesis se pueden dividir en dos grupos principales.

Primeramente, toda la documentación que aparezca en esta tesis, podrá ayudar a todo aquel investigador, estudiante o usuario que pretenda realizar un proceso similar al que aparecerá en próximos apartados. Ya sea exclusivamente la instalación o el uso de Great, con las diferentes posibilidades que ofrece, como crear un diagrama de eventos comunicacional, o transformarlo a un diagrama de clases UML; o sea realizar un ciclo completo como el de esta tesis, partiendo del diagrama de eventos comunicacional de Great, realizando la integración con IntegraNova, hasta la generación de código de IntegraNova.

En segundo lugar, las valoraciones extraídas tras todo el transcurso de este proyecto podrán ayudar al equipo de desarrollo de Great a realizar mejoras en los aspectos que encuentre problemas. Dentro del grupo PROS (PROS, 2020), equipo de investigación en el que soy participe aportando esta tesis, hay un contacto directo con Marcela Ruiz, creadora de Great. De esta forma, todo bug o brecha funcional que encuentre y sea comunicado será un gran *feedback* tanto para ella como para los compañeros que tienen intención de continuar el desarrollo de la herramienta.

1.5. Metodología

El desarrollo de este trabajo de fin de grado sigue las directrices establecidas en la metodología "Ciencia del diseño". (Wieringa, 2014) El ciclo de la ingeniería narra el ciclo de vida de un artefacto, como puede ser un producto software, separándolo en diferentes fases e iteraciones. Este ciclo divide el proceso en cuatro fases, las cuales son investigación del problema, diseño del proceso, validación del proceso y tratamiento de la implementación. Asimismo, cada fase tiene unas características principales, facilitando la identificación de la fase o las fases en la que ubicar el proyecto, como se puede apreciar en la figura 1.

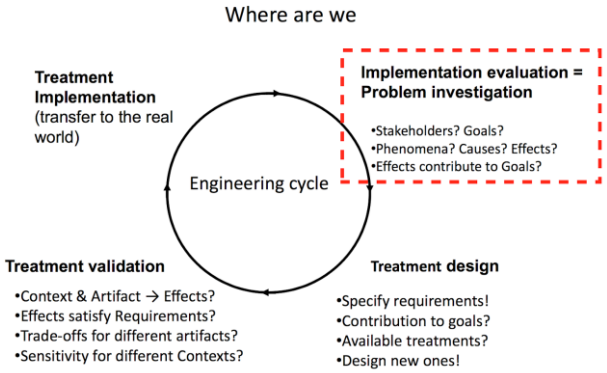


Figura 1. Ciclo de vida de un artefacto

Por lo tanto, analizando las diferentes opciones con respecto a esta tesis la opción más adecuada para obtener una visión completa del artefacto con el que trabajamos es etiquetar este trabajo dentro de la investigación del problema. Esto es debido a que esta tesis tiene como intención principal aportar conocimiento, no intentar realizar una mejora. Asimismo, aunque este trabajo contiene diseños e implementaciones propias, no tiene como intención exhibir estas como soluciones para la creación de un producto o como una metodología; muy por el contrario, pretende proporcionar doctrina para ayudar a otros proyectos posteriores.

Cada fase del ciclo de la ingeniería consta de un proceso interno para resolver las preguntas de conocimiento dentro del proceso de desarrollo ingenieril, llamada ciclo empírico y dividido en diferentes operaciones. En la figura 2 podemos apreciar el ciclo empírico que constituye la investigación del problema. Este ciclo parte del análisis del problema, continua el diseño de la investigación, la validación de esta investigación, la ejecución de la investigación y finalmente se analizan los resultados. Este es un ciclo iterativo, de esta forma a partir de los resultados obtenidos aparecen nuevas incógnitas que investigar.

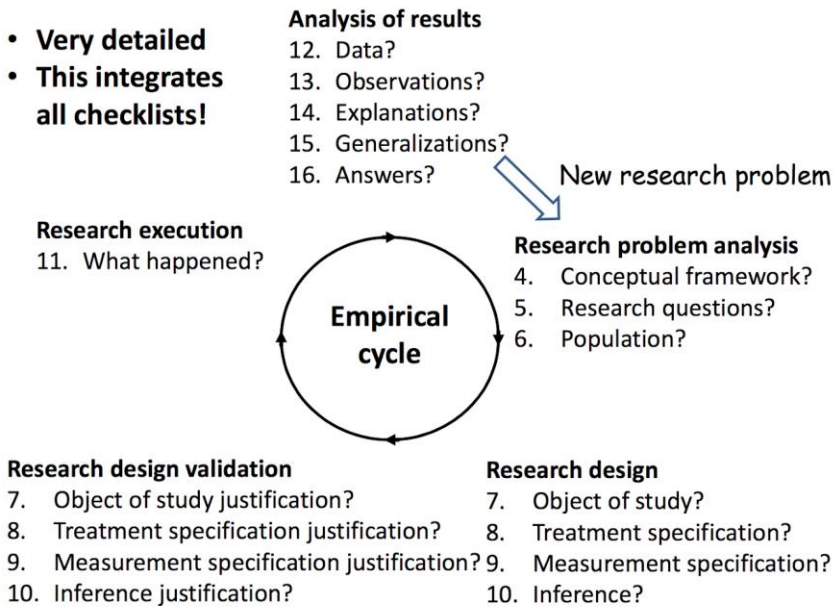


Figura 2. Ciclo empírico de la investigación del problema

1.6. Estructura

Los contenidos de la tesis están estructurados en función a la combinación de los aspectos básicos de una tesis de grado junto a la estructura del ciclo empírico comentado en el apartado anterior. Los principales puntos del trabajo son los siguientes:
1. Introducción: Se muestran las bases de los contenidos que se trabajarán a lo largo de la tesis y se muestra una visión amplia del problema planteado.
2. Estado del arte: Estudio del contexto tecnológico en el que se encuentran las tecnologías y espacio de conocimiento que se pretende rellenar.

3. Análisis del problema: Estudio extendido del problema que se va a trabajar y todo lo que le rodea. Así pues, se plantearán los puntos que se incidirá y se trabajará en los puntos posteriores.
4. Diseño de la investigación: Análisis de la viabilidad de solución planteada y especificación de los conceptos o aspectos importantes para el desarrollo de esta solución.
5. Desarrollo de la solución propuesta: Explicación del proceso que ha seguido la solución propuesta para convertirse en la solución final. También se incluye una exposición de los problemas que han surgido y las decisiones importantes que se han tomado.
6. Implantación: Evolución desde la idea planteada de la solución hasta la obtención del producto final, junto a la documentación de todo el proceso.
7. Validación: Evaluación del cumplimiento del método por parte de la herramienta mediante las reglas de las que el método está basado.
8. Conclusiones: Reflexiones extraídas del trabajo realizado, presentación de los problemas y ventajas del trabajo con estas herramientas.
9. Trabajos futuros: Lista de posibles mejoras o trabajos que se podrían trabajar en un futuro.
10. Bibliografía

1.7. Colaboradores

Como se ha comentado en apartados anteriores, esta tesis forma parte de un proyecto de mayor tamaño desarrollado por diferentes integrantes del grupo de investigación PROS, dentro de la UPV. Este trabajo de fin de grado está escrito y trabajado por José Vicente Rico Bellot, no obstante, hay diferentes colaboradores que tienen un papel importante dentro del desarrollo de la tesis.

En primer lugar, René Noel López, profesor de la universidad de Valparaíso de Chile. René forma parte del grupo PROS, donde realiza su tesis doctoral dentro del mismo proyecto en el que se encuentra esta tesis. De este modo esta tesis se desarrolla con su supervisión, donde el aporta una visión de cuál debe ser el camino a seguir en las decisiones del día a día.

En segundo lugar, Oscar Pastor López, profesor de la UPV y tutor de este trabajo de fin de grado. Oscar es el director del grupo de investigación PROS. Oscar ha sido el encargado de seleccionar el tema de la tesis y colabora enfocando el trabajo por el camino que realmente debe tener este, además de ayudar a tomar las decisiones importantes.

A continuación, destacar la colaboración de equipo de trabajo de la herramienta Great, encabezado por Marcela Ruiz, por ceder la herramienta para que pueda trabajar con ella. Marcela es una profesora de la ZHAW (*Zúrich University of Applied Sciences*) y colabora con el grupo PROS.

Por último, destacar la colaboración del grupo de investigación PROS por ofrecer un espacio donde realizar una aportación y cederme las licencias necesarias para realizar la tesis.

1.8. Convenciones

Las reglas de escritura que se siguen durante el desarrollo de esta tesis son las siguientes:

- Las palabras extranjeras se remarcarán en cursiva.
- Se entrecomillarán las citas textuales externas a la obra.

2. Contexto tecnológico

2.1. Ingeniería dirigida por modelos

El desarrollo software dirigido por modelos surge tras la creación de la *Model Driven Architecture* (MDA) en el 2000 por Object Management Group y tenía como intención mejorar la asentada arquitectura dirigida por objetos. Entre los creadores de programas siempre ha existido la necesidad de elevar los niveles de abstracción necesarios para la creación de nuevos programas; así pues, gracias a la mejora en los equipos de computación y compiladores existentes surgió la MDA.

La ingeniería dirigida por modelos (MDE) es un nuevo paradigma dentro de la ingeniería del software basada en la MDA que se ocupa de la utilización de modelos del software para mejorar la productividad y algunos otros aspectos de la calidad del software, como el mantenimiento y la interoperabilidad entre sistemas. Este paradigma ha mostrado su potencial para dominar la complejidad arbitraria del software al proporcionar un mayor nivel de abstracción y elevar también el nivel de automatización. (Vicente-Chicote, 2012)

Un modelo es una simplificación de un entorno, como resultado de un proceso de abstracción, y ayuda a comprender y razonar sobre ese entorno. Un modelo oculta ciertos detalles para mostrar aquellos relevantes para cierto propósito. Los modelos son expresados mediante alguna notación que depende de su propósito y destinatarios. En el caso del software, un modelo es una descripción de un aspecto de un sistema software escrita en un lenguaje bien definido. De esta forma, los modelos software permiten especificar aspectos tales como los requisitos, la estructura, el comportamiento o el despliegue de un sistema. Como es lógico, si un modelo debe ser interpretado por una máquina para generar código o controlar un sistema, deberá estar expresado con una notación descrita formalmente, por ejemplo, mediante un metamodelo. (Catalunya, 2020)

La MDA se puede estructurar en cuatro niveles de abstracción, donde podemos observar las diferentes fases por las que pasa un producto software en este paradigma. Estos niveles en MDA se estructuran de M3 a M0, siendo M3 el nivel más abstracto y M0 el nivel más concreto.

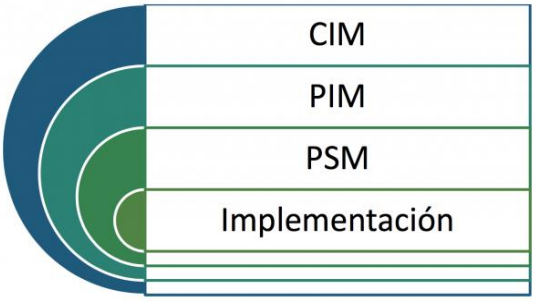


Figura 3. Fases de la MDA

En primer lugar, encontramos el M3 o *Computation Independent Model* (CIM). El CIM se corresponde con el metametamodelado y se le conoce como el modelo del dominio o del negocio, ya que está modelado en términos familiares a los expertos del negocio.

Representa lo que se espera del sistema, sin abordar temas referentes a las tecnologías del mismo.

En segundo lugar, está el M2 o *Platform Independent Model* (PIM). Este nivel corresponde al metamodelado y permite diseñar el sistema en diferentes plataformas, enfocándonos en una tecnología concreta.

El siguiente nivel es el M1 o *Platform Specific Model* (PSM). En el PSM los diseños que se realicen corresponderán a una plataforma concreta y combinan el modelo PIM con los requerimientos concretos de esta plataforma.

Por último, encontramos el M0, también llamado implementación a código del sistema. Este nivel corresponde a la implementación en algún lenguaje de programación tradicional del sistema que hemos diseñado. En MDE este nivel se obtiene a partir de la transformación del M1 o previos, eliminando la necesidad de conocer las técnicas de programación para construir un producto software.

Por regla general, en MDA los entornos de desarrollo no se ubican en un nivel concreto, en realidad su objetivo es convertir el esquema creado en un nivel a otro de un nivel diferente, generalmente inferior. En esta tesis se trabaja con dos programas, Great, el cual parte de un esquema de nivel M2 y lo transforma a M1, también llamada transformación *Model-to-Model* (M2M) debido a que a partir de un esquema realiza las transformaciones necesarias para convertirlo en otro modelo. También está IntegraNova que obtiene el esquema M1 generado previamente y lo transforma a la implementación a código o nivel M0, también llamado transformación *Model-to-text* (M2T). Actualmente en el mercado existen diferentes opciones para realizar tanto transformaciones M2M como M2T. Algunas opciones destacables son las siguientes. (MDETOOLS, 2020)

- ATL (ATL Transformation Lenguaje) es un lenguaje de transformación de modelos M2M y una herramienta para la edición de estos modelos. En el campo de la MDE, ATL proporciona formas de producir un conjunto de destino a partir de un conjunto de modelos fuente.

Figura 4. Logo ATL

```
rule clase2tabla {
    from
        ca: UML!Clase
    to
        ta: ER!Tabla (
            Nombre <- ca.nombre,
            Padre <- ca.diagrama,
            columnas <- col
        ),
        col: ER!Columna (
            Nombre <- 'id_' + ca.nombre
        )
}
```

Figura 5. Ejemplo ATL

- Desarrollado sobre la plataforma Eclipse, el entorno integrado ATL proporciona una serie de herramientas de desarrollo estándar. ATL proporciona una forma de producir modelos de destino a partir de un conjunto de modelos de origen. mediante reglas que definen cómo se combinan y navegan los elementos del modelo de origen para crear e inicializar los elementos de los modelos de destino. (ATL, 2020) (MDETOOLS, 2020)

- MetaEdit+ es un entorno de desarrollo enfocado en la creación y uso de lenguajes de modelado específicos de dominio y la transformación a otros modelos (M2M). El conjunto de herramientas maduras proporciona las herramientas y el proceso para definir sus lenguajes y generadores de modelado, sin tener que escribir código. (MetaCase, 2020) (MDETOOLS, 2020)

Figura 6. Logo MetaEdit+

Entre las diferentes ventajas que proporciona esta herramienta podemos destacar los siguientes puntos:

1. Acceso programático a los datos del modelo y a diferentes funciones a través de API.
2. Importar y exportar modelos con otras herramientas.
3. Parámetros de línea de comando para automatizar operaciones.
4. Ejecutar comandos externos y analizar archivos a través de generadores.

- Acceleo es una tecnología basada en plantillas que hace uso de transformaciones para crear generadores de código personalizados. Le permite producir automáticamente cualquier tipo de código fuente a partir de cualquier fuente de datos disponible en formato EMF. Acceleo está integrado en Eclipse, facilitando así la instalación y uso de los componentes integrados. (MDETOOLS, 2020)

  Entre las características de Acceleo podemos destacar estos aspectos:

  1. Fácil transformación M2T mediante reglas
  2. Integración con Eclipse, con todas las facilidades que no aporta Eclipse IDE añadidas
  3. Generación de código sencilla e incremental

- MagicDraw es una herramienta de modelado de sistemas con soporte de trabajo en equipo la cual tiene como principal característica que es capaz de realizar transformaciones M2M y M2T. Esta herramienta de desarrollo facilita el análisis y el diseño de sistemas y bases de datos orientados a objetos (OO). MagicDraw proporciona independencia de cualquier proceso de desarrollo de software, de esta forma permite la centralización de negocios y modelado de procesos, captura de requisitos y diseño. Dentro de su usabilidad destaca la su interfaz sencilla y enfocada a aumentar la productividad. (Team, 2020) (MDETOOLS, 2020)
  Algunos de los puntos destacables de la herramienta son:

  1. Generación de código y otros diagramas a partir de diagramas.
  2. Posibilidad de trabajar en colaboración dentro del mismo diagrama.
  3. Generación de informes relacionados.
  4. Personalización de las herramientas de navegación de acceso rápido.

2.2. Análisis comunicacional

Antes de centrarnos en el análisis comunicacional hay varios términos relacionados que es importante tenerlos en consideración. En primer lugar, un sistema de información es un conjunto de datos que interactúan entre ellos con un fin común. En lo que respecta a la tesis un sistema de información refleja una organización, de esta forma el término abarca todos los datos necesarios para realizar procesos internos o procesos externos que necesitan información de la organización. En segundo lugar, un requisito software es una necesidad de implementación en el sistema que tiene un *stakeholder* o usuario del sistema. Así pues, la ingeniería de requisitos es una rama de la ingeniería del software que pretende traducir esos requerimientos en funciones o cambios implementados en el sistema. (España, 2009)

De esta forma, podemos definir el análisis comunicacional como un método para el desarrollo y computación de sistemas de información, este método es una aproximación de la ingeniería de requisitos que se centra en las interacciones que existen entre el sistema y el entorno. Esto quiere decir que al contrario que otros métodos que tienen como punto central los objetos que realizan las acciones, en el análisis comunicacional se destacan los usuarios o agentes internos o externos y las acciones que realizan en el sistema para cumplir con procesos.

Dentro del diseño de soluciones software es recurrente separar el proceso en dos partes, el diseño del problema y el diseño de la solución. El espacio del problema hace referencia a todo aquello que ocurre antes de empezar a diseñar y que conlleva explorar la problemática que ha motivado el proceso de diseño y, por lo tanto, comprender qué es lo que nuestro producto o servicio debe resolver. Por otro lado, el espacio de la solución engloba todas las actividades orientadas a analizar las posibles soluciones a un problema y encontrar la que mejor se ajusta al problema definido. En análisis comunicacional esta idea se adapta mediante un sistema de niveles, donde se parte de nivel inicial en el que se describe a grandes rasgos de la organización, y que mediante transformaciones avanza niveles hasta llegar a una implementación real de la organización. (España, 2011) (González, 2011)

El espacio del problema abarca tres niveles, en el primer nivel de requisitos, también llamado nivel de sistema y subsistemas, el analista describe el sistema operativo desde el punto de vista estratégico. Por un lado, cuando la organización es compleja, es aconsejable descomponer el problema en subsistemas o áreas organizativas. Por otro lado, el analista obtiene requisitos relacionados con los indicadores comerciales de nivel estratégico.

En el segundo nivel o nivel de procesos el analista describe los procesos comerciales desde una perspectiva comunicacional. El objetivo es descubrir intercambios de información entre usuarios, también llamadas interacciones comunicativas, internamente y con su entorno, y describirlas teniendo en cuenta sus aspectos dinámicos y estáticos; es decir, crear el Diagrama de eventos comunicativos y el Glosario de objetos de negocio, respectivamente. El diagrama de eventos comunicativos es el tipo de diagrama que se utiliza en análisis comunicacional, en los siguientes apartados se explicará extensamente y se observarán ejemplos. (Moreno, 2020)

En el nivel de interacciones comunicativas los eventos comunicativos que aparecen en el Diagrama de eventos comunicativos deben describirse en detalle. Los requisitos asociados a un evento se estructuran mediante una Plantilla de especificación de eventos. La plantilla está compuesta por un encabezado y tres categorías de requisitos: contacto, contenido comunicacional y requisitos de reacción.

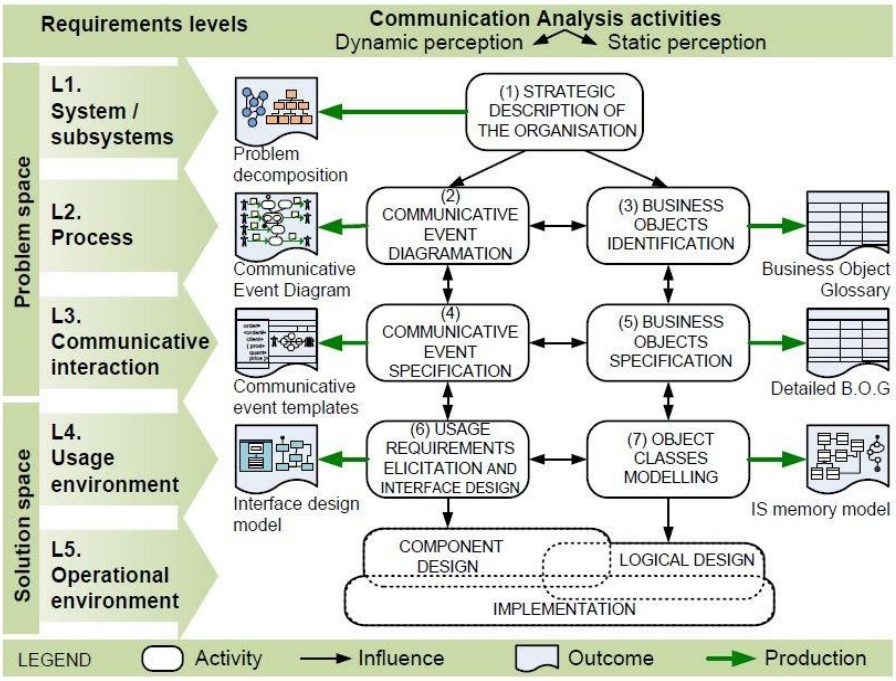


Figura 7. Niveles de requerimientos en análisis comunicacional

Respecto al espacio de la solución encontramos los dos niveles finales. El cuarto nivel o nivel del uso del entorno es el utilizado para organizar la interfaz del Diagrama de eventos comunicativos y realizar la elicitación del uso de los requerimientos detallados previamente, de esta forma se pueden construir los objetos de negocio que serán modelados en la parte estática. En este nivel el analista generará un diagrama estructurado correctamente junto a todos los objetos de negocio definidos. (Gómez, 2010)

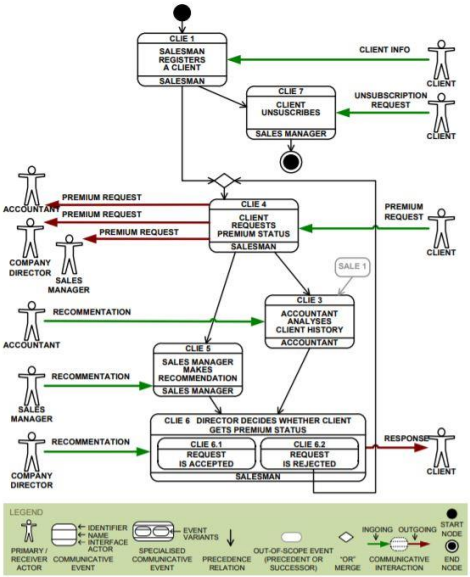


Figura 8. Ejemplo diagrama de eventos comunicativos

Por último, en el nivel cinco se combinará el diagrama de eventos comunicativos junto a la capa de negocio que se ha modelado para generar una implementación del entorno que se ha modelado a lo largo del proceso. Esta implementación no necesariamente es una implementación a código, como aparece en apartados anteriores existen implementaciones mediante modelos de otra metodología. (Gómez, 2010)

Respecto a la situación tecnológica en la que se encuentra este método, el análisis comunicacional se está aplicando actualmente en grandes proyectos en entornos industriales; como es la integración de Anecoop S. Coop (un importante distribuidor español de frutas y hortalizas) con sus cooperativas asociadas. (informativos.net, 2020)

2.3. OO-Method

OO-Method es un método de desarrollo de software orientado a objetos orientado a modelos que se enfoca en modelar sistemas de información empresarial a nivel conceptual de forma automática. OO-Method está basado en OASIS (López, 1997) que es un lenguaje formal para la especificación conceptual de sistemas de información en base al paradigma Orientado a Objetos, este método sigue una propuesta estructural similar a la del análisis comunicacional separando la parte de espacio del problema y espacio de la solución. En este caso OO-Method tiene la particularidad de que genera automáticamente el espacio de la solución, permitiendo así al usuario centrar sus esfuerzos en afinar los automatismos para implementar correctamente el espacio del problema (Sergio España, 2010) (Pastor, 2007)

El modelo conceptual de OO-Method se genera automáticamente a partir de un modelo de requerimientos. Este se obtiene a partir de la combinación de cuatro vistas complementarias, como son un modelo de objetos, un modelo dinámico, un modelo funcional y un modelo de presentación. (Pastor, 2007)

Partiendo del modelo de objetos permite especificar el aspecto estático del sistema en forma de un diagrama de clase compatible con UML (*Unified Modeling Language*). Este modelo permite describir los siguientes elementos: los objetos de negocio en términos de clases de objetos, relaciones estructurales entre clases, agentes del sistema y relaciones entre agentes y servicios de clase. Las primitivas de modelado del modelo de objetos son las habituales de un diagrama de clases UML, es decir las clases tienen atributos y servicios (los tradicionalmente llamados métodos en UML), y están interrelacionadas por medio de relaciones estructurales.

Figura 9. Ejemplo modelo de objetos y modelo dinámico

El modelo dinámico especifica las posibles secuencias de invocaciones de métodos que pueden ocurrir en la vida de un objeto mediante un diagrama de transición de estado. Este diagrama de estado impone restricciones en la activación de servicios de clase mediante las siguientes primitivas de modelado.

Respecto al modelo funcional, este modelo especifica el efecto que tienen las invocaciones de métodos en el estado de los objetos mediante reglas de evaluación. Estas reglas de evaluación están escritas en pseudocódigo independiente del lenguaje de programación. El modelo funcional contiene las evaluaciones para un atributo variable y un evento de una determinada clase. Es decir que las evaluaciones que se realicen sólo se podrán definir en eventos y variables, nunca para transacciones y operaciones, puesto que los servicios moleculares producen el cambio de estado en los objetos al enlazar la ejecución de diversos eventos. (Sergio España, 2010)

Service name: C_bar	
Precondition 1	
Formula	date_of_qux <> NULL
Error msg	"This foo has not been quxed yet"
Precondition 2	
Formula	bar_type = NULL
Error msg	"This foo has already been barred"

Figura 10. Ejemplo modelo de funcional

Por último, el modelo de presentación proporciona al analista un medio para la captura de los requisitos de interfaz de usuario. De esta manera, permite especificar interfaces abstractas y homogéneas, independientes de consideraciones de diseño. El modelo se basa en distintos patrones, como pueden ser la presentación, la facilidad de encontrar la información o la seguridad de acceso para recoger los requisitos.

OO-Method es una idea de Oscar Pastor director del grupo PROS, que junto al equipo que ha ido formando parte de este grupo de investigación han ido avanzando en la idea de este método. Respecto al contexto tecnológico actual del método, en 2006 Oscar Pastor en colaboración con IntegraNova sacó al mercado la herramienta OlivaNova, que actualmente forma parte del modelador de IntegraNova. Esta herramienta será la que se utilice en esta tesis, y como se ha comentado previamente, permite generar una implementación a código a partir de un diagrama. (informativos.net, 2020)

2.4. Discusión

La crítica a los trabajos previos en las metodologías comentadas previamente se puede separar la en dos partes. Esto se debe a que la primera parte, donde se encuentra la ingeniería dirigida por modelos no hay intención de tener un gran impacto y aportación de conocimiento, al contrario que el análisis comunicacional y OO-Method, áreas donde se pretende ampliar el conocimiento existente.

Por un lado, está la MDE, esta parte de la ingeniería es mundialmente utilizada y hay una gran cantidad de trabajos en la UPV que trabajan con este concepto. De hecho, dentro de la rama de ingeniería del software en el grado de ingeniería informática en la UPV hay una asignatura de esta metodología. Respecto a estos proyectos y tesis, la principal diferencia es la iniciativa, ya que en lo métodos revisados se modela el sistema para generar su código, y en este proyecto se modela el problema y sus requerimientos, para luego pasar al modelo del sistema.

Por otra parte, están los métodos con los que trabajan las herramientas que se van a utilizar en esta tesis, estos son el análisis comunicacional y OO-Method. Ambas metodologías son muy de nicho, por tanto, no hay una gran cantidad de proyectos dedicados a su investigación. En la UPV las diferentes investigaciones que se han realizado en las que se aporta conocimiento a estas materias pertenecen al grupo PROS, en concreto he analizado tres de ellas que aportan información relevante. Las tesis analizadas son "Communication Analysis: a Requirements Engineering Method for Information Systems" (España, 2009) y "Methodological integration of Communication Analysis into a model-driven software development framework" (España, 2011) ambas escritas por Sergio España, y "Aplicación práctica de MDA con OlivaNova para una aplicación de evaluación de recursos humanos" escrita por Diego Solá. En estas tesis se trabaja con el concepto de análisis comunicacional y OO-Method, pero en las tesis de Sergio España únicamente se trabaja con conceptos teóricos no aplicados a una herramienta, y en la tesis de Diego Solá parte de un diagrama de OO-Method para realizar el proceso de Olivanova.

2.5. Propuesta

La propuesta de este trabajo de fin de grado es realizar el desarrollo completo de un caso de uso utilizando las herramientas Great para realizar las transformaciones de análisis comunicacional a OO-Method y IntegraNova modeller para transformar de OO-Method a una implementación a código. De esta forma complementaria las tesis de Sergio España (España, 2009) (España, 2011), donde falta el uso de herramientas para llevar a la práctica la teoría que plantea, y la tesis de Diego Solá, donde falta desarrollar previamente un ejemplo y transformarlo hasta llegar al diagrama de OO-Method que él plantea.

3. Análisis del problema

A partir de la propuesta de esta tesis es interesante analizar el problema que se plantea e investigar las diferentes oportunidades y opciones con las que se puede trabajar. De esta forma, se proporcionará la mayor información posible sobre todo lo que rodea al trabajo, facilitando la implementación que se realizará posteriormente.

3.1. Análisis del entorno

3.1.1. Análisis de la propiedad intelectual

Actualmente todos los programas informáticos obtienen sus beneficios a partir de licencias, que son usadas por los usuarios que pretender usar el software en cuestión. La siguiente lista está compuesta por los programas que se van a usar a lo largo del proyecto y el método de obtención de la licencia para trabajar con ellos.

Programa	Funcionalidad	Obtención
Great	Modelado de análisis comunicacional y transformación a diagrama de OO-Method	Clave proporcionada por equipo de desarrollo de Great
IntegraNova Modeller	Modelado de OO-Method	Clave proporcionada por grupo PROS
IntegraNova Star	Transformación de OO-Method a implementación	Clave solicitada a IntegraNova, a la espera de respuesta
Microsoft Teams	Comunicación con el grupo PROS (René, Oscar…)	Clave proporcionada por la UPV
Microsoft Office	Redactar tesis	Clave proporcionada por la UPV
Visual Studio	Evaluar código *frontend*	Versión de estudiante
Visual Studio Code	Evaluar código *backend*	Versión de estudiante

3.1.2. Análisis ético

Respecto a los debates éticos que puede plantear esta tesis son limitados, ya que este trabajo no va a crear ningún producto nuevo. Sin embargo, es un paso más para un avance muy grande que sería la automatización del desarrollo software, facilitando a todo el que quisiera crear su programa que pudiera hacerlo de una forma relativamente sencilla.

Esta problemática plantea un debate ético similar al de la implantación de robots industriales que sustituyen a las personas como peones en fábricas. Analizando la historia reciente del ser humano comprobamos que todo avance ha hecho progresar más rápido al ser humano y al igual que ha eliminado muchos trabajos, ha creado otros nuevos, más sencillos y cómodos para el que los realiza.

Comparándolo con otras áreas, sería equiparable al avance que ha tenido la industria del cine, cuando se ha pasado de editar las escenas cortando los fotogramas uno a uno a ahora que con un programa de edición pulsando el ratón en segundos puedes cortar una escena fácilmente. Otro ejemplo más cercano aún es el de diseñador de piezas industriales, antes del gran avance en la informática las piezas industriales se diseñaban a mano sobre un papel y estos diseños se le proporcionaban al empleado que cortaba la placa de hierro para que aplicara los cortes necesarios sobre la placa, con todas las imperfecciones que esto supone. Actualmente existen diseñadores 3D que desde su ordenador diseñan la pieza visualizándola en todo momento y envía la orden para que una máquina realice los cortes exactamente como debe, reduciendo el riesgo de que haya errores humanos.

En definitiva, a lo largo de la historia han ido apareciendo inventos que han cambiado sectores y eliminado puestos de trabajo, pero con esa eliminación de trabajo han aparecido oportunidades y nuevos trabajos que no existían; por tanto, que esta tesis colabore en el cambio del sector de la programación no se podría considerar algo negativo.

3.1.3. Análisis de riesgos

Hay muchos factores que pueden alterar el correcto desarrollo de esta tesis, incluso impidiendo que se finalizara, estos son algunos de los más relevantes:

- Problema personal: El proyecto se desarrolla durante un tiempo prolongado, durante ese tiempo podrían surgir inconvenientes que no me permitieran finalizarlo, como problemas familiares, un accidente, problemas de salud…
- Desastre natural: Podría aparecer un desastre natural que me afectara en primera persona, como podría ser un terremoto, inundaciones… La mayoría de los casos no me imposibilitarían finalizar la tesis, pero si la retrasarían.
- *Pandemia mundial: En diciembre apareció en China un virus llamado SARS-CoV2 el cual los expertos dijeron que podría afectar a toda la población mundial. Esta tesis está escrita en parte durante el proceso de confinamiento en el que ha estado España por culpa de este virus. Esto no ha imposibilitado continuar la tesis, pero ha afectado suavemente a la comunicación.

También hay factores enfocados a situaciones más reales respecto a las tecnologías:

- Problemas con la tecnología: El desarrollo dirigido por modelos solo se estudia en una asignatura a lo largo de la carrera y la documentación para el uso de las

herramientas es limitada. El alumno podría ser incapaz de finalizar el proceso por falta de documentación al respecto.
- Problemas de claves: En el proyecto se trabaja con diferentes programas y en la mayoría se necesitan claves para su activación y uso. Podría darse el caso de que no se le ceda inmediatamente la clave de algún programa al alumno, atrasando o imposibilitando la correcta finalización de la tesis.

3.2. Plan de trabajo

Para el correcto desempeño del proyecto dentro del tiempo establecido se va a seguir un plan de trabajo que va a servir de hoja de ruta. Esta tesis se va a estructurar en diferentes etapas diferenciadas, dentro de cada etapa habrá diferentes reuniones con René y con Óscar para comentar los avances, los problemas y las dudas que vayan apareciendo. Al igual que las reuniones se intercalan con las diferentes etapas, las escrituras del documento y la toma de apuntes se realizará en todas las etapas; así pues, no se incluirá una etapa propia para la escritura porque en todas se realizarán aportaciones.

La primera etapa del proyecto se basará en investigación y colecta de toda la información necesaria, ya sea en documentos o sean las licencias necesarias para realizar el producto final. En esta fase debe haber un intercambio continuo de información entre los supervisores y el alumno, donde le compartirán todas las tesis, libros o documentos que están relacionados para facilitarle el trabajo. La mayor parte del tiempo de esta etapa el alumno debe dedicarse a la recopilación y análisis de la información que obtiene, tanto de las fuentes y documentos que le proporcionan desde el equipo, como de la información que recoge por su cuenta.

En segundo lugar, la siguiente etapa estará enfocada en el modelado del proceso de negocio. Esta fase partirá del desarrollo de varias ideas del sistema que se pretende plantear hasta conseguir plasmar el diseño final en la herramienta, esta se divide en tres partes marcadas. En primer lugar, el alumno plantea varias ideas de sistemas a los supervisores, y entre ambos elegirán cuál es el que se adapta más a las necesidades que se pretenden rellenar, cuando se elija el sistema concreto, refinarán las características que tiene hasta conseguir el ejemplo de sistema más adecuado posible. Cuando ya tengan el ejemplo de sistema el alumno debe utilizar la información de análisis comunicacional y de la herramienta Great que dispone para diseñar el diagrama correspondiente al sistema que han elegido. Durante el modelado en la herramienta se realizarán cambios en el sistema, ya sea porque la herramienta no permite plasmar alguna opción, o porque hay oportunidades de mostrar nuevas características que no se habían planteado.

Cuando ya se disponga de una versión final del diagrama de análisis comunicacional, se pasará a la tercera etapa. En esta etapa se realizará la transformación, con su respectivo análisis, del diagrama que se dispone a su correspondiente en OO-Method en la herramienta Great. Great ofrece la posibilidad de realizar esta transformación, el alumno debe observar el diagrama obtenido y realizar los ajustes necesarios en el diagrama previo para que la transformación se realice correctamente y sin fallos evitables.

La cuarta se centrará en la evaluación del modelo de sistema del sistema generado, donde a partir del diagrama en OO-Method obtenido anteriormente y la herramienta IntegraNova realizaremos una transformación de ese diagrama a código. En primer lugar, el alumno modelará el sistema obtenido en IntegraNova, adaptando los resultados a las posibilidades que ofrezca la herramienta para que el sistema sea mínimamente viable. En segundo lugar, se generará una mínima versión ejecutable del sistema con la herramienta IntegraNova Star, perteneciente a IntegraNova.

Por último, se realizará un análisis completo del proceso realizado centrado en los resultados obtenidos y se sacarán conclusiones. Por un lado, se analizará la versión ejecutable del sistema y el código perteneciente a esta, y por otro lado se expondrá la viabilidad del proceso que se ha realizado en la tesis como proceso real mediante un análisis de las oportunidades y de las brechas funcionales que nos ofrece utilizar la unión de estas herramientas.

3.3. Presupuesto

Para calcular el presupuesto del proyecto hay que tener en cuenta varios factores. Por un lado, hay que tener en cuenta el coste humano, es decir el coste en horas de las personas que forman parte. Por otro lado, habría que valorar el coste no humano, ya sea el coste de las licencias o de los equipos necesarios para el correcto desarrollo de la tesis.

El tiempo de desarrollo aproximado de esta tesis es de quince semanas, durante estas semanas la aportación semanal en horas de los participantes será:

- 22 horas José Vicente Rico - Realizar todo el desarrollo del proyecto, desde la recolecta de material hasta el diseño y transformación de los esquemas
- 4 horas René Noel - Encargado de la supervisión diaria del proyecto, ayuda con las dudas básicas que surgen a lo largo de la tesis
- 1 hora Oscar Pastor - Encargado de la tutoría y la supervisión general del proyecto, dirigiendo el proyecto y resolviendo dudas de gran impacto dentro de la tesis

Además, también se podría tener en cuenta el coste de los equipos de trabajo y de las licencias, pero como ya se contaba con los equipos y las licencias son de uso académico y gratuitas no será necesario.

Con estas previsiones observaríamos que el alumno ha realizado 330 horas. El salario de un alumno en prácticas oscila entre 4€/hora y 8€/hora, por tanto, suponiendo que la supervisión de los tutores la proporciona la universidad el presupuesto aproximado del proyecto es de:

15 * 22 * 4 = 1320
15 * 22 * 8 = 2640

El coste oscila entre los 1320€ y los 2640€. A este presupuesto habría que incluirle el coste de las horas de colaboración de los tutores y el precio del equipo e instalaciones.

4. Diseño de la validación

Esta tesis tiene como objetivo principal validar la viabilidad del uso de las herramientas Great e IntegraNova para la generación del código fuente de un sistema de información, a partir del modelamiento conceptual de los requerimientos y del sistema. De esta forma, antes de mostrar el proceso de la validación es interesante mostrar algunos aspectos en los que se va a incidir posteriormente; ya sea especificar el proceso que se va a seguir o la medición para verificar el cumplimiento de los objetivos. Además, también se mostrará el diseño de los diferentes esquemas que se van a diseñar o generar mediante las herramientas.

4.1. Objeto de estudio

El objeto de estudio de la validación es un caso práctico de desarrollo de software dirigido por modelos. En el caso, se realizarán modelos conceptuales de procesos de negocio y se transformarán a un modelo conceptual del sistema, para el que posteriormente se generará automáticamente el código fuente del sistema. Tanto en los modelos como las transformaciones se utilizarán las herramientas GREAT e IntegraNova, para poder observar el comportamiento y la funcionalidad de todas las posibilidades que nos ofrecen estas herramientas.

El caso práctico consiste en el proceso que realiza una empresa de disfraces para la venta de sus productos. Esta empresa vende directamente al consumidor sus productos, ya sean disfraces o accesorios. El proceso empieza cuando un cliente acude a la empresa porque quiere realizar un pedido de disfraces, al llegar el dependiente le muestra un catálogo donde están todos los disfraces y accesorios que la empresa fabrica. De las diferentes posibilidades que hay el cliente elige un disfraz; como en muchas ocasiones los disfraces se piden para una fecha concreta es necesario que el cliente los obtenga antes de esa fecha. Así pues, el dependiente comprueba el stock que tienen del disfraz, si no tienen suficientes piezas de ese disfraz disponibles, el cliente evalúa si prefiere que le cosan los disfraces que faltan o elige otros disfraces de los que hay en la tienda. Si quiere que les cosan los disfraces que faltan el dependiente le envía la orden de coser esos disfraces. Además, en muchas ocasiones los disfraces van acompañados de accesorios que hay en la tienda, por consiguiente, el cliente elige el accesorio que quiere que acompañe al disfraz. Por último, cuando el cliente ya tiene los disfraces y accesorios preparados, el dependiente le entrega la factura con el coste total del pedido. Al pagar esta factura se le efectúa la entrega del pedido.

4.2. Especificación del proceso

El proceso ha sido en detalle descrito en la sección plan de trabajo, y se puede resumir en los siguientes pasos:

- Diseño del caso
- Preparación del entorno
- Modelado de proceso de negocio con método de Análisis Comunicacional en Great
- Transformación y análisis de modelo CA a OOM
- Traspaso de modelo OOM de Great a IntegraNova
- Generación de código del sistema
- Evaluación y análisis de cobertura.

4.3. Especificación de la medición

Con el objetivo de tener una correcta aplicación de las tecnologías y las herramientas a lo largo de la tesis es importante tener conocimiento de estas y seguir los estándares impuestos. De esta forma, es interesante imponer unas pautas en la medición de los diferentes aspectos que se están trabajando, así como asignar una forma de medir que se cumplan.

4.3.1. Trazabilidad conceptual

El primer aspecto a valorar es la trazabilidad conceptual. La trazabilidad conceptual es la unión de las funcionalidades de Great con los principios de análisis comunicacional de una forma unificada y completa. Es decir, realizar un análisis de todas las funcionalidades que nos ofrece Great mediante el objeto de estudio para evaluar cuanto se adapta a todas las posibilidades que nos brinda el análisis comunicacional.

Para medir la correcta aplicación de los aspectos comentados previamente, nos apoyaremos tanto en los conocimientos de la tecnología que tenemos el alumno y los supervisores, como en la herramienta Great y en las posibilidades que nos ofrece.

Por un lado, el alumno ha estado documentando y observando diferentes aplicaciones de la tecnología, y los supervisores son experimentados en el campo. De esta forma, al finalizar las primeras versiones del esquema se hará una puesta en común donde se dará el visto bueno de todos los integrantes del proyecto y se verificará que el esquema planteado es coherente con las directrices marcadas por la tecnología y la herramienta.

Por otra parte, la propia herramienta Great nos ayuda a identificar si la idea de esquema que tenemos es aceptable dentro de los parámetros comentados previamente. Al diseñar un modelo dentro de la herramienta se puede generar el diagrama de OO-Method equivalente. Así pues, si queremos comprobar que la idea de esquema de análisis comunicacional es coherente solo tenemos que generar el diagrama OO-Method, y sin entrar en más detalle de si el diagrama es correcto o no, si el diagrama OO-Method está completo la propuesta de diagrama de análisis comunicacional será coherente.

Esta medición se evaluará con la métrica:

- número de eventos comunicativos trazables desde CA a funcionalidades del sistema /número de eventos comunicativos totales

4.3.2. Evaluación de brechas funcionales de la herramienta de transformación

Uno de los aspectos más importantes en el proyecto son las brechas funcionales que tiene Great y el impacto que estas pueden tener para la correcta implementación de los esquemas diseñados, por tanto, la forma de evaluarlas es trascendental. Una brecha funcional es un fallo o ausencia de aplicación en la implementación de una funcionalidad necesaria para que funcione de forma completa un sistema. En este caso se debe evaluar si Great aplica correctamente todas las posibles funciones, características y transformaciones que brinda el análisis comunicacional.

Para llevar a cabo esta verificación con la mayor objetividad posible se seguirán unas reglas escritas por Sergio España en la tesis "Methodological integration of Communication Analysis into a model-driven software development framework" (España 2011). Sergio España las nombra como Object Model Rules y en ellas especifica los diferentes atributos que debe tener un esquema de análisis comunicacional para poderse transformar correctamente a OO-Method, desde los más básico, como que debe tener inicio o final hasta aspectos más detallados.

Las reglas están ordenadas como OMn (siendo n el número) acompañadas de un título que trata de resumir aquello que evalúan.

- Regla OM1. Determinación del alcance de la derivación.
- Regla OM2. Identificación de los *reference field*
- Regla OM3. Ordenación de eventos
- Regla OM4. Derivación de una nueva clase a partir de una agregación
- Regla OM5. Derivación de un atributo a partir de un *data field*
- Regla OM6. Selección del identificador de clases
- Regla OM7. Especificación del tipo de un atributo
- Regla OM8. Especificación del tipo de un atributo
- Regla OM9. Decisión sobre si se solicita un atributo en el momento de la creación
- Regla OM10. Decisión sobre si se permite valores nulos
- Regla OM11. Derivación de una relación estructural a partir de subestructuras anidadas
- Regla OM12. Derivación de una relación estructural a partir de un reference field
- Regla OM13. Inclusión de un servicio de creación a una clase recién creada
- Regla OM14. Inclusión de un servicio de fin o edición para mensajes complejos
- Regla OM15. Selección de una clase extendida
- Regla OM16. Inclusión de un servicio de edición de una clase extendida
- Regla OM17. Definición de una transacción
- Regla OM18. Determinación de los contactos y eventos de reacción
- Regla OM19. Introducción de una clase agente a partir de un rol organizacional
- Regla OM20. Derivación de relaciones de agentes para roles de interfaz
- Regla OM21. Creación de un agente para todo el sistema

Esta medición se evaluará con la métrica:

- Número de reglas que se cumplen / Número de reglas totales*
- Separación por categoría (Creación de clases, atributos, servicios…)
- Número de funcionalidades de OOM del sistema trazables a Eventos comunicativos/total de eventos comunicativos.

*Solo se contarán las reglas que se puede verificar su cumplimiento

Cuando se verifique el cumplimiento de estas reglas en Great se indicará si se cumple y es una funcionalidad correcta de Great o no se cumple y se trata de una brecha funcional, también cabe la posibilidad de que sea un aspecto que se trabaja de forma externa a la plataforma o que es un proceso que es interno y que no se muestra al usuario el resultado.

4.3.3. Evaluación del modelo generado

Al igual que es conveniente que el modelo conceptual sea coherente y completo, cuando ya tengamos una propuesta real del diseño del modelo y lo transformemos a OO-Method es importante valorar si el diagrama generado es como debería ser el diagrama si lo diseñaremos directamente en la tecnología, ya que nos indicaría que la herramienta cumple su propósito.

Para la evaluación del modelo OO-Method generado por Great utilizaremos la misma funcionalidad que para evaluar la trazabilidad conceptual, ya que obviamente para valorar el modelo generado habrá que transformar el modelo de análisis comunicacional. Para ello seguiremos los mismos pasos que realizamos en el primer apartado, pero en este caso, al pretender evaluar la completitud y la veracidad del modelo no simplemente nos valdrá con que parezca que es un modelo funcional. En este caso se evaluará de diferentes formas, tratando de asegurar de la mejor forma posible que se ha generado un modelo válido.

En primer lugar, se recurrirá tanto a la documentación de OO-Method recopilada como a un informe proporcionado por el grupo de investigación PROS, escrito por Sergio España y Marcela Ruiz donde documentan los diferentes elementos que debe tener un diagrama de esta tecnología. Este documento es un caso de uso práctico, acompañado de la documentación pertinente necesaria para la comprensión de los conceptos, tanto de OO-Method como tecnología, como de su integración a IntegraNova. En este caso nos apoyaremos de las definiciones teóricas de la tecnología para poder verificar si el diagrama que nos proporciona Great es válido.

En segundo lugar, al igual que en la evaluación de la trazabilidad conceptual, el alumno se reunirá con los supervisores, expertos OO-Method para que valoren la validez del diagrama, ya sea por errores del alumno, que se solucionarían con ajustes en el diagrama de análisis comunicacional, como por falta de completitud de Great. Esta información ayudará en gran parte a mejorar la herramienta y a comprobar si es viable el uso profesional de este proceso.

Para evaluar esta parte también se utilizará el *Gold Standard*, es importante mencionar que el *Gold Standard* corresponde a un modelo de Casos de Uso elaborado por personal

del Centro de Investigación PROS que representa toda la funcionalidad que idealmente se debería haber generado para el sistema. Esta se pondrá frente al modelo generado y se valorarán los resultados.

Esta medición se evaluará con la métrica:

- número de conceptos del modelo generado / nro total de conceptos del modelo de IntegraNova
- casos de uso implementados / casos de uso totales

4.3.4. Evaluación del código generado

Por último, otro de los aspectos que es relevante valorar su cumplimiento es el código generado tras la transformación de IntegraNova. Este será el último punto del desarrollo del proyecto antes de su finalización y presentación, por lo que nos permitirá apreciar en perspectiva si ha merecido la pena todo el proceso realizado para el resultado que obtengamos.

El código se evaluará de forma similar a la forma de evaluar el modelo en el punto anterior, ya que el código al igual que el diagrama se pueden realizar de diferentes formas y no hay una forma completamente objetiva de evaluarlas, de esta forma estas serán las formas de evaluar el código generado por IntegraNova.

Por una parte, se deberá evaluar si el código se adapta correctamente a los principios del lenguaje en el que se transforme. IntegraNova ofrece varios lenguajes para la transformación, tanto en el frontend como en el backend, por lo tanto, el alumno elegirá el que vea más viable, ya sea por los conocimientos que tiene del lenguaje o por la facilidad para documentarse y evaluar el código generado.

Por otra parte, para comprobar si un código está bien generado, tanto en el propio lenguaje, como en la fusión entre frontend y backend, se puede ejecutar y comprobar si funciona. De esta forma, analizaremos el proceso iniciado en la creación del modelo conceptual y verificaremos si el código generado nos permite ejecutarlo directamente sin problemas o cuantos problemas puede crear para conseguir la ejecución.

Con la unión de ambos puntos se puede obtener una visión de la calidad del código que IntegraNova es capaz de generar y nos ofrecerá una perspectiva de la viabilidad del uso de este método como método continuado de producción de software.

4.4. Diseño detallado de la validación

En la tesis podemos encontrar dos esquemas que se deben construir manualmente, los cuales nos ofrecen una visión en perspectiva del proyecto, ambos tienen una relevancia suficiente para incidir en ellos y explicarlos detalladamente. Así pues, en los siguientes apartados se mostrarán en detalle los esquemas diseñados en el proyecto con en detalle.

4.4.1. Diseño del Modelo de procesos de negocio

El modelo de procesos de negocio utiliza como tecnología análisis comunicacional y contiene las instrucciones indicadas en el objeto de estudio. Este modelo ha sido diseñado específicamente para mostrar todas las opciones que nos brinda Great, para poder analizar el comportamiento que tienen al generar posteriormente el diagrama de clases a partir de este. De esta forma, encontraremos un nodo inicial y uno final, eventos comunicacionales, con sus identificadores, roles organizacionales asignados a estos eventos y que se comunican por medio de estructuras de mensaje que contienen agregaciones, iteraciones… También podemos encontrar nodos lógicos de tipo or y and, y variedades de precedencia, como puede ser precedencia simple, eventos que contienen varios eventos según la decisión que se tome y diferentes precedencias o eventos con diferentes precedentes.

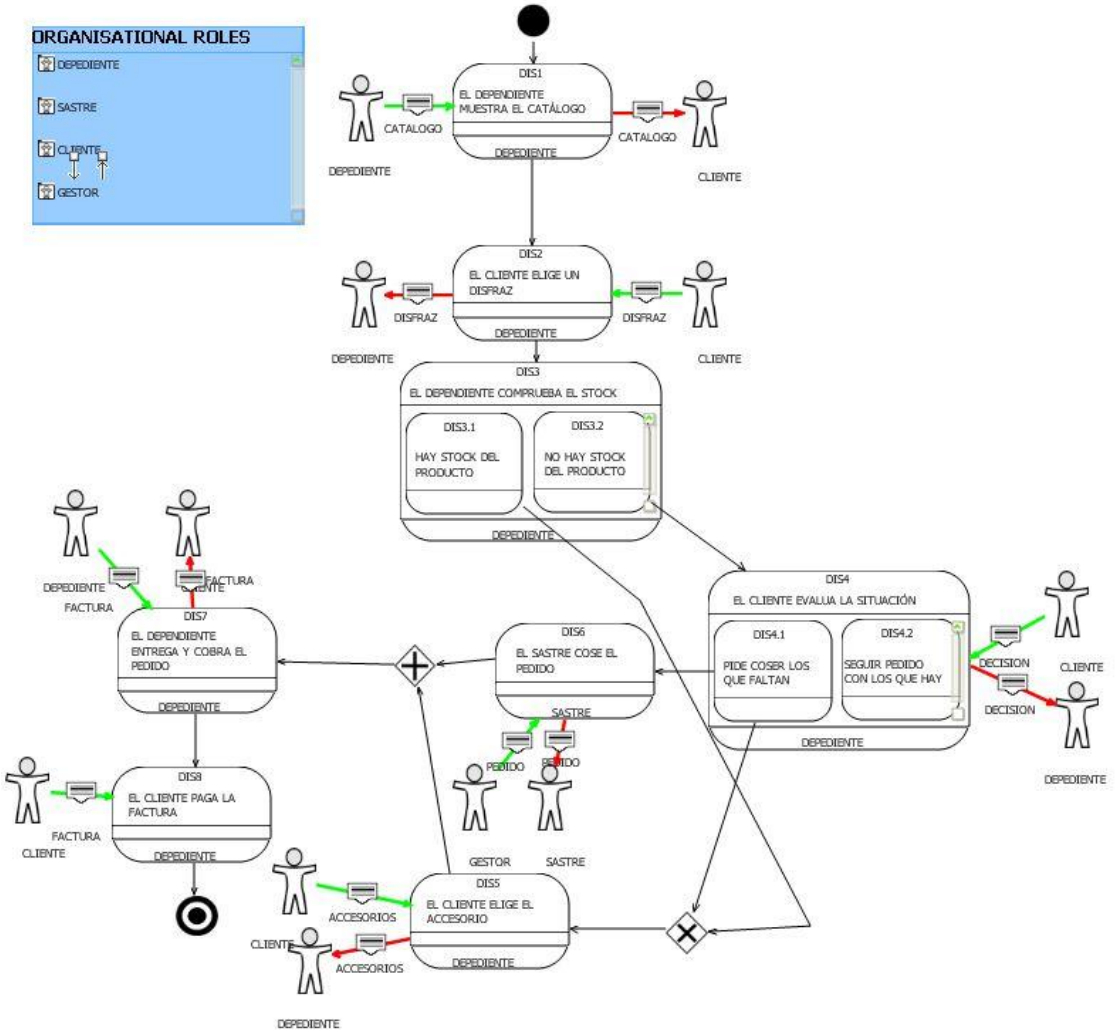


Figura 11. Diagrama de eventos comunicativos

Este es el esquema de análisis comunicacional completo, para una mejor comprensión de todos los conceptos en los que se incide, se explicará el esquema siguiendo el orden establecido por los nodos haciendo énfasis en la comunicación entre los roles organizacionales y el contenido de las estructuras de mensaje. Para una mejor comprensión de estas explicaciones es interesante leer el objeto de estudio previamente.

Para diseñar el esquema se siguió el siguiente proceso, con intención de sacar el máximo provecho de los mínimos eventos, para que estuviera todo esquematizado y simple. En primer lugar, se seleccionó el caso en base a un problema real, y con usuarios con disponibilidad para poder resolver dudas (tienda de disfraces). A continuación, se modeló el flujo de eventos comunicativos asociado de la compra de disfraces, identificando actores y puntos en los que interactúan. Se verificó que todos los elementos de AC se incluyeran en el diagrama. Por último, se especificaron las estructuras de mensaje para describir la información intercambiada entre los actores en los eventos comunicativos.

En primer lugar, un cliente aparece en la tienda con intención de comprar varios disfraces a la empresa. Así pues, el dependiente lo primero que hace es mostrarle el catálogo al cliente para que elija su favorito. Esta comunicación aparece en la estructura de mensaje CATALOGO.

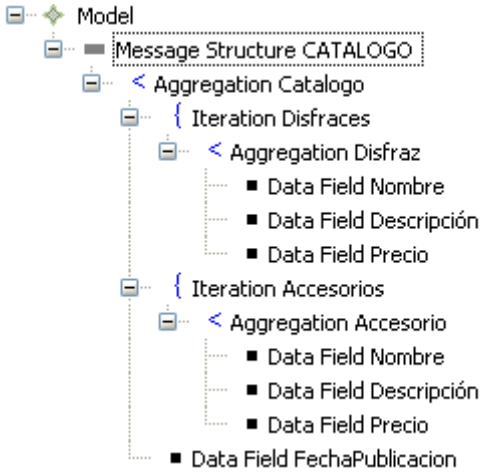


Figura 12. Estructura de mensaje CATALOGO

Como se aprecia en la imagen la estructura de mensaje CATÁLOGO contiene una agregación llamada catálogo. Esta contiene una iteración con la lista de disfraces, donde cada disfraz tiene un nombre, el cual es su identificador, una descripción y un precio. Al igual, también contiene una iteración con la lista de disfraces con un nombre como id, una descripción y un precio. Además, el catálogo contiene una fecha de publicación para tener un control de versiones de los catálogos que se van publicando. Estos *data field* que encontramos tienen un tipo propio, en ambos casos Nombre y Descripción son de tipo *text* y Precio es de tipo *Money*.

Cuando el cliente revisa el catálogo y tiene claro el disfraz que quiere elegir se lo comunica al dependiente mediante la estructura de mensaje DISFRAZ.

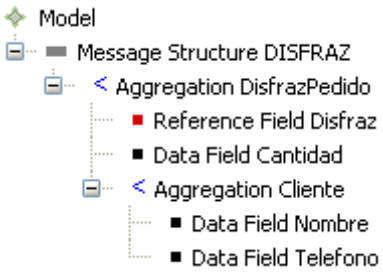


Figura 13. Estructura de mensaje DISFRAZ

En este caso en la estructura de mensaje DISFRAZ encontramos la agregación DisfrazPedido, que contiene una referencia al disfraz del catálogo que el cliente ha elegido. También contiene un *data field* con la cantidad de disfraces que quiere comprar.

Tras elegir el disfraz el dependiente comprueba si tienen stock del producto con la cantidad que el cliente ha indicado, aquí encontramos la primera decisión. Si tienen stock del disfraz irá directamente a elegir el accesorio, pero en caso de que haya stock suficiente del disfraz pasarán a los siguientes pasos.

Al comunicar el dependiente al cliente que no tienen stock del producto el cliente debe tomar la decisión de elegir si cosen los disfraces que faltan o seguir solo con los que haya en stock. Esta decisión se observa en la estructura de mensaje DECISIÓN.
En esta estructura de mensaje podemos encontrar una agregación llamada Decisión y un *data field* llamada Respuesta de tipo *boolean*.

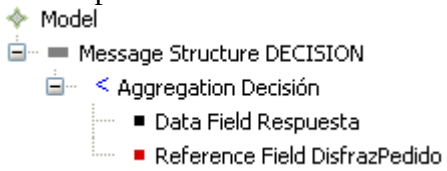


Figura 14. Estructura de mensaje DECISION

Si el cliente elige coser los disfraces que faltan la decisión se bifurca en dos direcciones. En primer lugar, se le indica al sastre que cosa los disfraces restantes. Esta información se comunica mediante la estructura de mensaje PEDIDO.

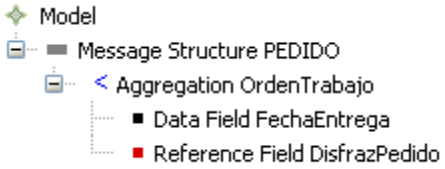


Figura 15. Estructura de mensaje PEDIDO

En esta estructura de mensaje observamos que contiene una agregación llamada Pedido con una referencia al pedido de disfraces que había realizado el cliente y una fecha de entrega máxima que tienen que tener en cuenta.

Por otro lado, la decisión de elegir coser los disfraces también deriva en la elección del accesorio, donde se encuentra con la posibilidad anterior de que hubiera stock del producto mediante un nodo lógico OR. Este nodo tiene como objetivo elegir el primer precedente que venga, es decir que o se cumple la opción de que hay stock o la de que no hay stock y cosen los que faltan. En este caso, al igual que antes el cliente elegía el disfraz ahora elige el accesorio que complementa este disfraz.

La estructura de mensaje que contiene la elección del accesorio llamada ACCESORIOS contiene una agregación llamada AccesorioPedido. Esta agregación contiene una referencia al accesorio que el cliente ha elegido de todos los que hay en el catálogo, también hay un *data field* con la cantidad de accesorios que el cliente quiere.

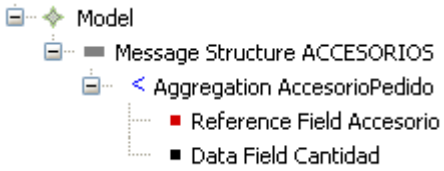


Figura 16. Estructura de mensaje ACCESORIOS

Tras elegir el disfraz se unirán de nuevo las opciones y encontraremos un nodo lógico tipo AND donde solo continuará el proceso si el sastre ha cosido los disfraces y el cliente ha elegido accesorios. Al cumplirse estas precondiciones se procederá a entregar la factura al cliente. Esta comunicación se producirá mediante la estructura de mensaje FACTURA.

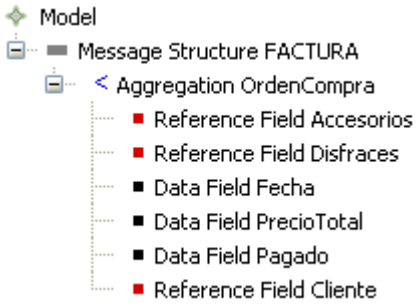


Figura 17. Estructura de mensaje FACTURA

Esta estructura de mensaje contiene los accesorios elegido y los disfraces elegidos, junto con la fecha que se efectúa la factura y el precio total a pagar.

Por último, el cliente paga la factura y se le entrega el pedido. Este evento comunicacional está creado para comprobar que sucede al escalonar las referencias a otras agregaciones y evaluar el comportamiento de Great.
Esta estructura de mensaje llamada también FACTURA es muy parecida a la otra llamada de la misma forma, solo que en esta se incluye una referencia a la agregación de la otra factura, la fecha de pago y la cantidad pagada.

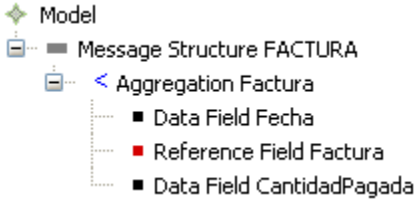


Figura 18. Estructura de mensaje FACTURA2

Finalmente llegamos al nodo final, que nos indica que se ha finalizado correctamente la compra de los disfraces y el proceso de venta se ha realizado.

4.4.2. Diseño de clases

El diseño del diagrama de clases se genera automáticamente en la herramienta Great, el único paso que se realiza de forma manual es la adaptación de este esquema a la herramienta IntegraNova. De esta forma, los esquemas que encontramos son los siguientes.

Este esquema contiene la transformación de todos los elementos que se han mostrado en la creación del esquema de análisis comunicacional. Como adaptación a este esquema, encontraremos el siguiente esquema en IntegraNova.

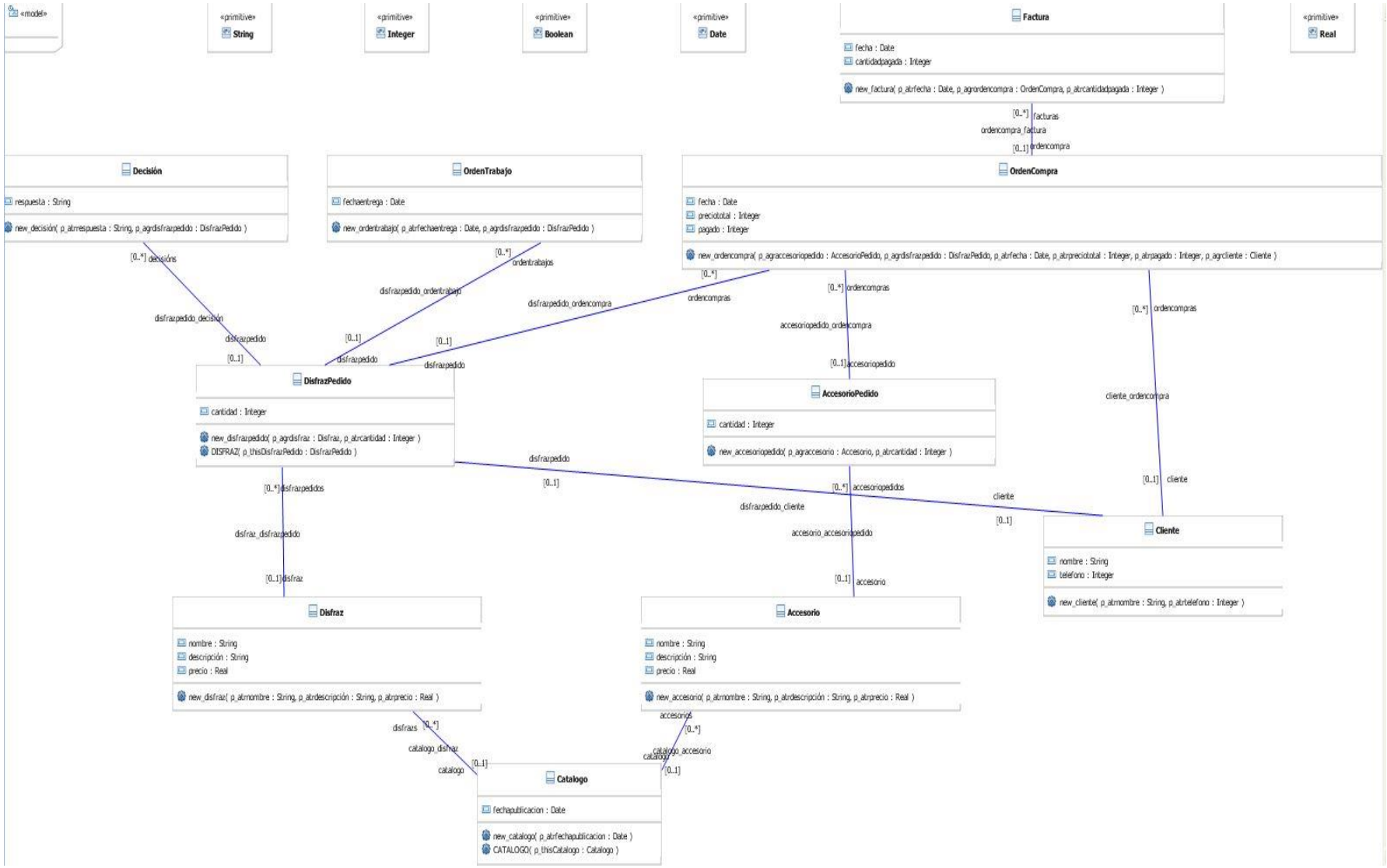


Figura 19. Diagrama OO-Method generado en Great

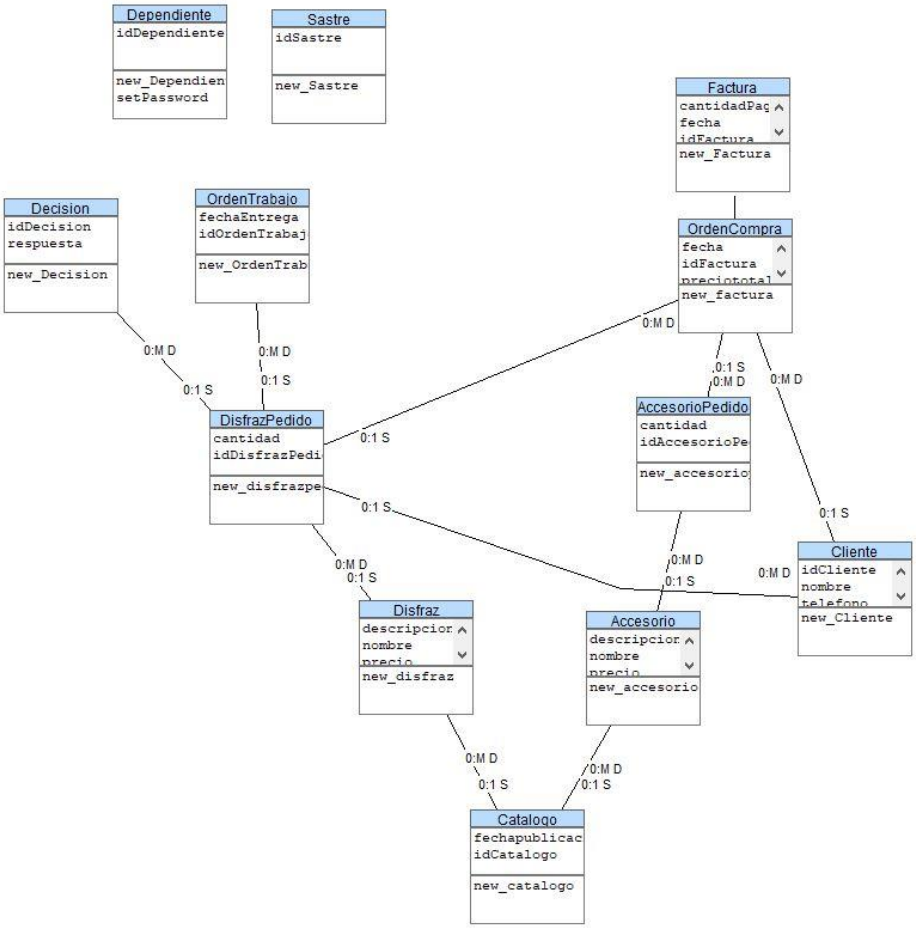


Figura 20. Diagrama OO-Method diseñado en IntegraNova

En este caso no vamos a incidir en la información que contiene cada clase, porque en el esquema generado en Great se puede observar toda esta información de forma clara. El único inciso que hay que incluir en el esquema en IntegraNova son las clases agente, que se encuentran arriba a la izquierda.

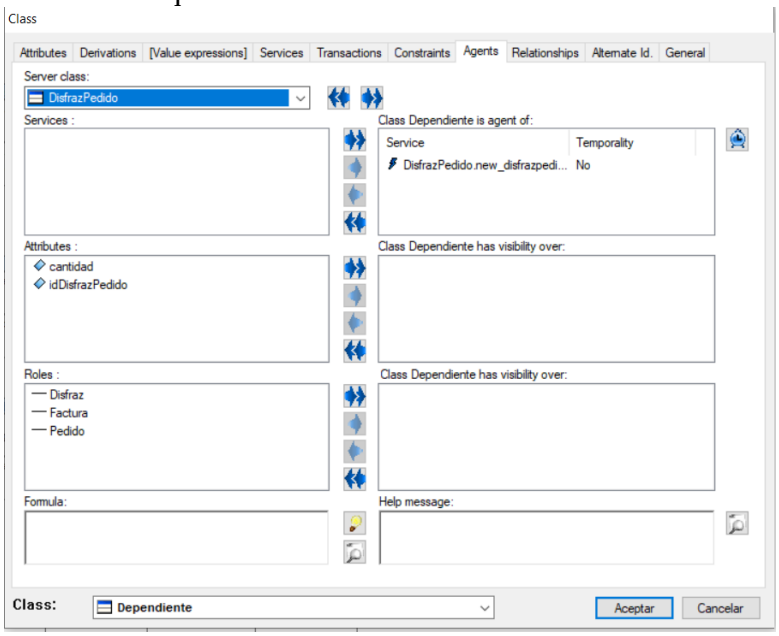


Figura 21. Clases agente en IntegraNova

Los agentes se asocian siguiendo la técnica de transformación, y que permiten definir los roles de los usuarios. En este caso podemos observar la clase Dependiente, donde en el apartado *Agents* tiene que indicar todas las clases que pertenecen a este agente, como por ejemplo DisfrazPedido. Este es un requisito indispensable para el correcto funcionamiento de la aplicación.

De esta forma, obtendremos el diagrama OO-Method completo diseñado en IntegraNova, con todas las relaciones entre clases y con las clases Agente existentes.

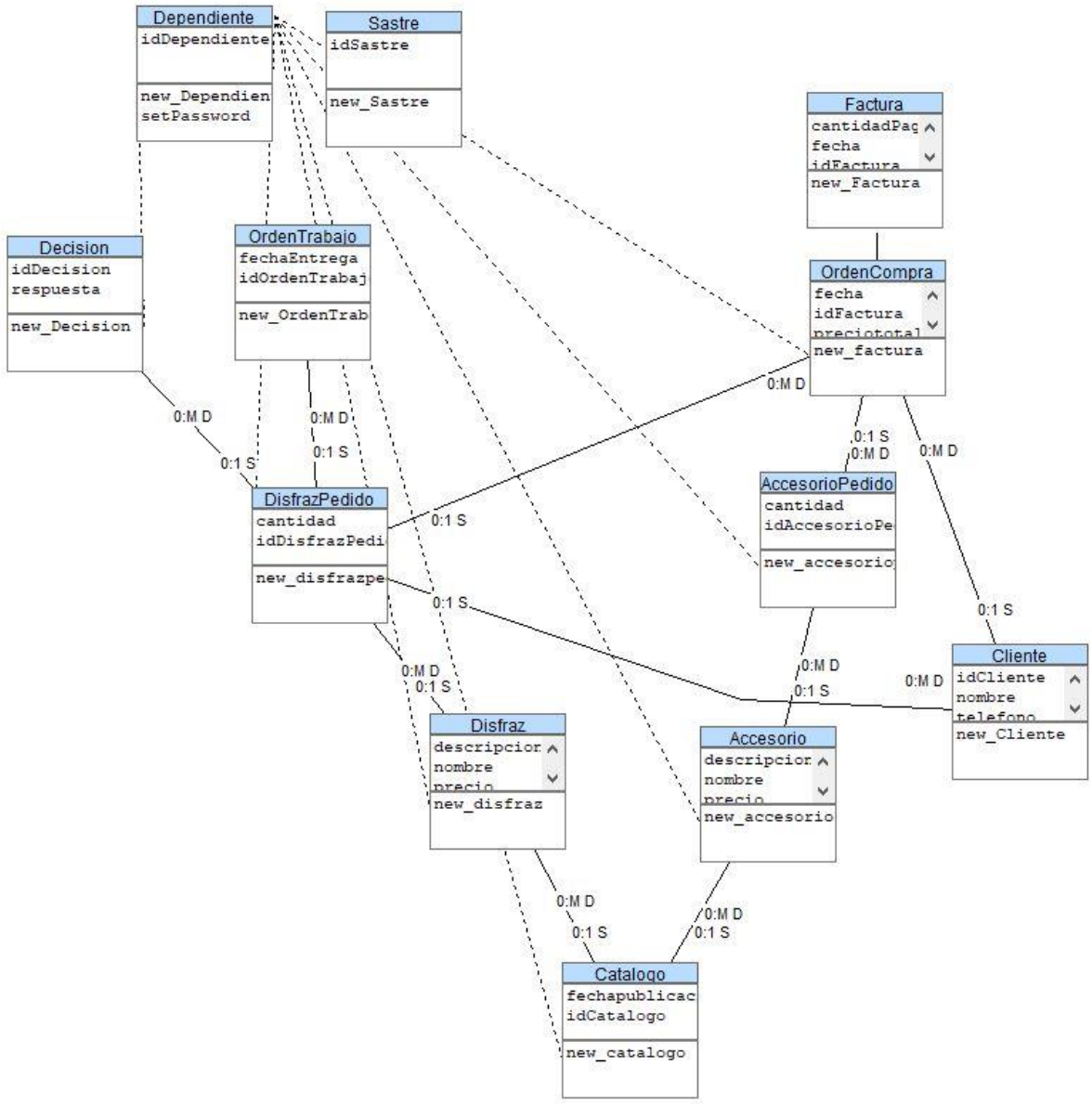


Figura 22. Diagrama OO-Method con clases agente en IntegraNova

5. Ejecución de la validación

En esta sección se mostrará de forma documentada la ejecución del objeto de estudio aplicado al caso práctico especificado a lo largo de los apartados anteriores mediante las tecnologías comentadas previamente. Asimismo, el desarrollo del caso práctico va a seguir un orden cronológico donde se explicará el proceso acompañado de material visual para facilitar la comprensión del material.

5.1. Puesta en marcha

Antes de empezar a utilizar los programas para el diseño del caso práctico hay que tener preparados una serie de componentes que serán necesarios para el correcto desarrollo del proyecto.

En primer lugar, es necesario un equipo con el que trabajar, en este caso concreto se ha utilizado el equipo del alumno con Windows 10, es posible que con otros sistemas operativos funcione de la misma forma, pero algunos programas son específicos para Windows.

Para la instalación de las herramientas se ha realizado una instalación hibrida. Para usar Great no se ha podido realizar una instalación común en Windows 10, ya que al ser un portable de un programa diseñado hace tiempo, no tenía compatibilidad con el sistema operativo y la versión concreta de Java. Para que el programa funcionara se ha instalado la herramienta **VirtualBox** y se ha utilizado una versión de **Windows XP** con **Java JDK 6**. Tanto la licencia de Great como el instalador de Windows XP han sido proporcionados por el grupo PROS.

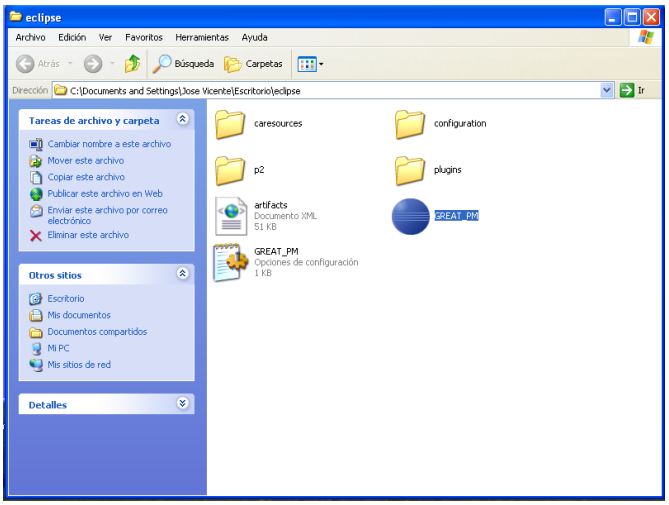


Figura 23. Instalación de Great

Al contrario que con Great, la instalación de IntegraNova ha sido una instalación típica para cualquier programa actual, el cual se instala mediante un ejecutable en Windows 10 sin problema. Para el uso de IntegraNova Modeller y IntegraNova Star son necesarias claves de instalación y uso, las cuales han sido proporcionadas por el grupo PROS.

Por último, para la verificación de la viabilidad del código es necesario un editor de código, desde IntegraNova recomiendan Visual Studio y es el IDE que se ha utilizado, su

instalación está adaptada a cualquier sistema operativo, en este caso se ha instalado en Windows 10.

5.2. Uso de Great

El primer escalón en la transformación de un diagrama a código es la herramienta Great, esta es una adaptación del IDE Eclipse Modelling Tools, por lo tanto, tendrá muchas similitudes con otras herramientas creadas de la misma forma.

El primer paso es crear un proyecto, así que pulsamos en **File -> New -> Project...** y en la ventana de New Project **General -> Project**

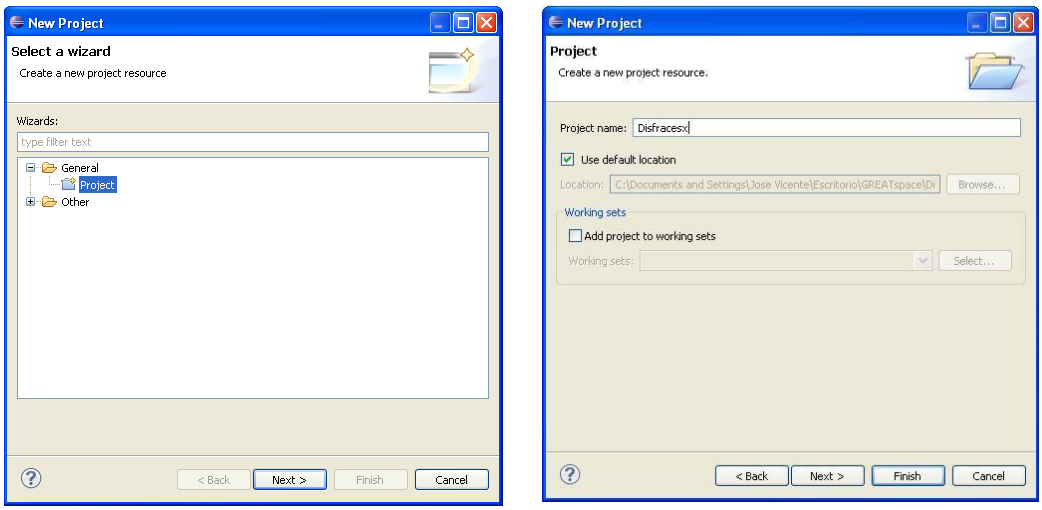


Figura 24. Creación de un proyecto en Great

Cuando ya tenemos el proyecto creado, pulsamos *click* derecho en este y **New -> Example -> Cametamodel Diagram**. El nombre del diagrama puede ser default.

Cuando ya tenemos el diagrama lo primero será añadir los roles organizacionales del sistema que pretendemos crear, por lo tanto, hacemos doble *click* en el menú de herramientas de la derecha donde pone **Organisational Role**. En caso de error en la escritura de algún nombre, para editarlo simplemente hay que pulsar el nombre y escribir en el teclado, de esta forma el nombre se borrará y se escribirá de nuevo.

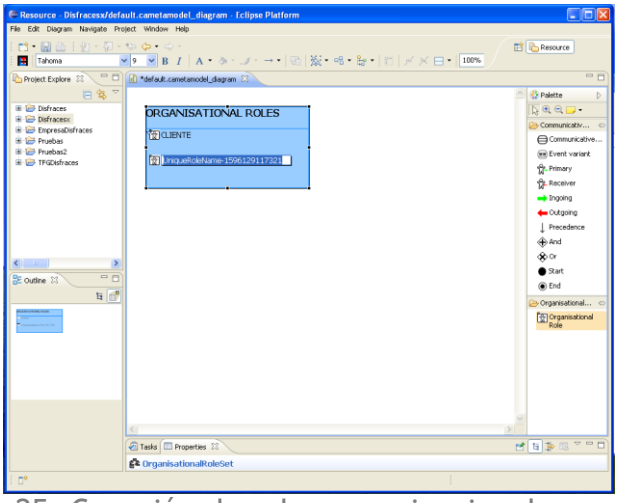


Figura 25. Creación de roles organizacionales en Great

Cuando ya tenemos los roles organizacionales procederemos a diseñar el proceso, y por lo tanto el primer paso será introducir un nodo inicial. Para ello pulsaremos en el icono de **Start** en la barra de herramientas de la derecha y pulsaremos en la parte del diagrama que queremos introducirlo. En el editor se puede mover la posición de todos los nodos a posteriori.

En segundo lugar, procederemos a introducir los eventos comunicativos, que al igual que el nodo *Start* haremos *click* sobre ellos y después pulsaremos en la parte del diagrama que queremos colocarlo. En estos eventos tendremos que introducir el **ID**, la **descripción**, y el rol organizacional al que están asignados. Para editar estos campos en el id y la descripción haremos un *click* y escribiremos la información, y para el rol haremos doble *click* y aparecerá un desplegable donde elegir uno de los roles organizacionales que hemos indicado previamente.

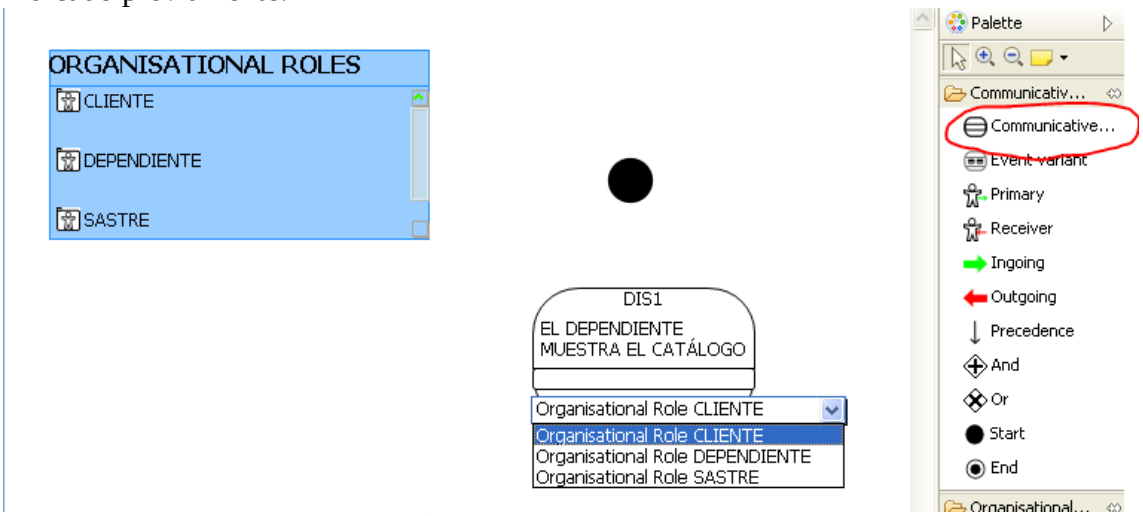


Figura 26. Creación de eventos comunicativos en Great

Cuando ya tenemos creado y documentado el evento procedemos a especificar la comunicación que se produce durante este evento. Para ello introduciremos un **Rol primario** y un **Rol recibidor**. Donde el rol primario es el que envía la información y el recibidor el que la recibe. Estos roles van acompañados del flujo de información, es decir, pulsaremos la flecha de **Ingoing** y la arrastraremos desde el rol primario hasta el evento, especificando el nombre de la estructura de mensaje que pretende comunicar. De la

misma forma, haremos lo mismo con la flecha de **Outgoing** desde el evento hasta el rol recibidor.

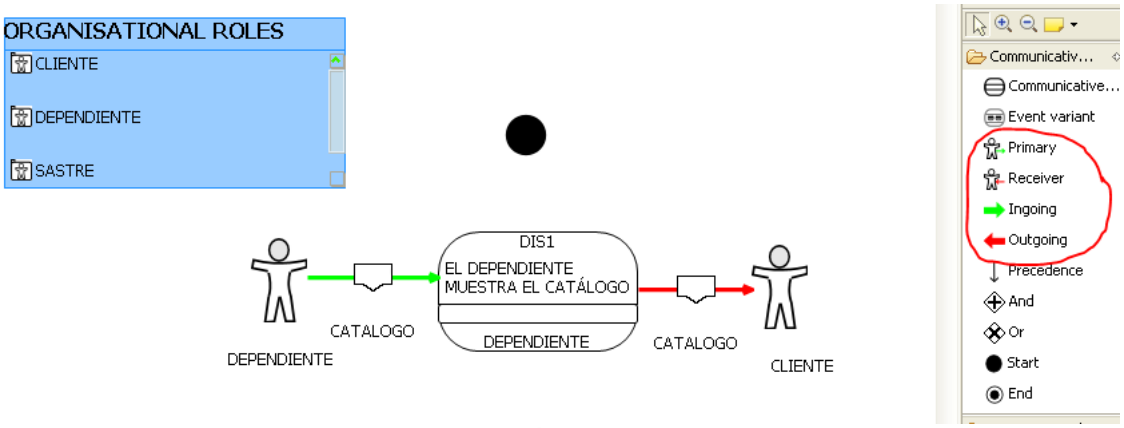


Figura 27. Inserción de roles en Great

Cuando ya tenemos unidos los roles con el evento crearemos la estructura de mensaje que contiene el mensaje que se pretende comunicar. Así pues, pulsaremos en el icono del mensaje en mitad de la flecha y elegiremos **CREATE_NEW_MS** si no hay ninguna estructura que se adapte a lo que hay que transmitir o elegiremos del desplegable la que haya ya creada. Solo puede haber un *Ingoing* y un *Outgoing* utilizando la misma estructura de mensaje.

Cuando estemos en la ventana de estructura de mensaje, para crear estructuras, pulsaremos *click* derecho ***New child -> Aggregation***. Cuando lo hayamos creado podremos modificar sus propiedades en la parte inferior en la ventana ***Properties***.

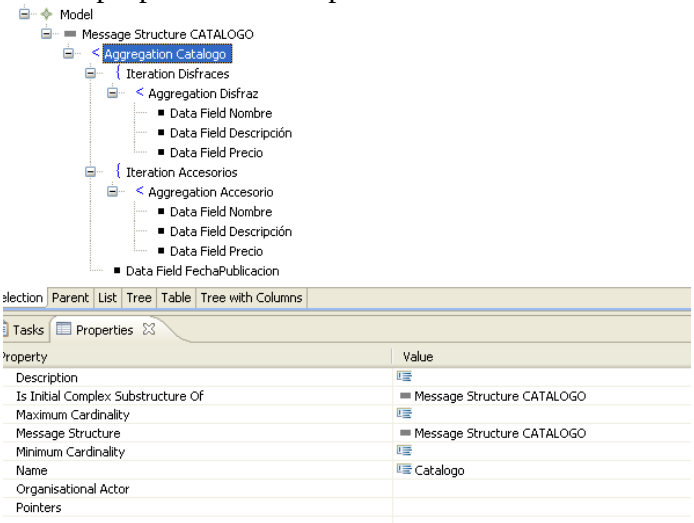


Figura 28. Creación de estructuras de mensaje en Great

Los contenidos de todas las estructuras de mensaje de este proyecto están en el apartado Diseño de procesos de negocio, por tanto, no se va a indagar tanto en qué se pone sino en cómo. El funcionamiento de algunas de las funcionalidades de esta parte, como el uso de *reference fields* se explicarán posteriormente en la parte de verificación del cumplimiento de brechas funcionales.

Cuando ya tengamos varios nodos podemos especificar el orden, para ello pulsaremos en la flecha de la parte derecha llamada ***Precedence*** y arrastraremos de un nodo al siguiente. Otras funcionales que Great proporciona para la creación de diagramas son la inclusión

de nodos OR y AND, los cuales se encuentran en la parte de la derecha y al igual que el nodo inicial se pulsan en la parte del diagrama que se quiera introducir. Su funcionalidad es continuar si alguno de los dos precedentes llega en caso de la OR y continuar si llegan los dos precedentes en caso de la AND.

También tenemos la opción de indicar si queremos incluir varias variantes en un mismo evento comunicacional para indicar que hay diferentes opciones que variarán en función de la elección que se produzca con el mensaje que se envía en ese evento. Para ello pulsaremos en ***Event variant*** en la barra de herramientas de la derecha y pulsaremos en el evento donde la queramos incluir. Su modificación es igual que la de un evento, pero sin asignarle un rol, ya que usa el del evento al que pertenece.

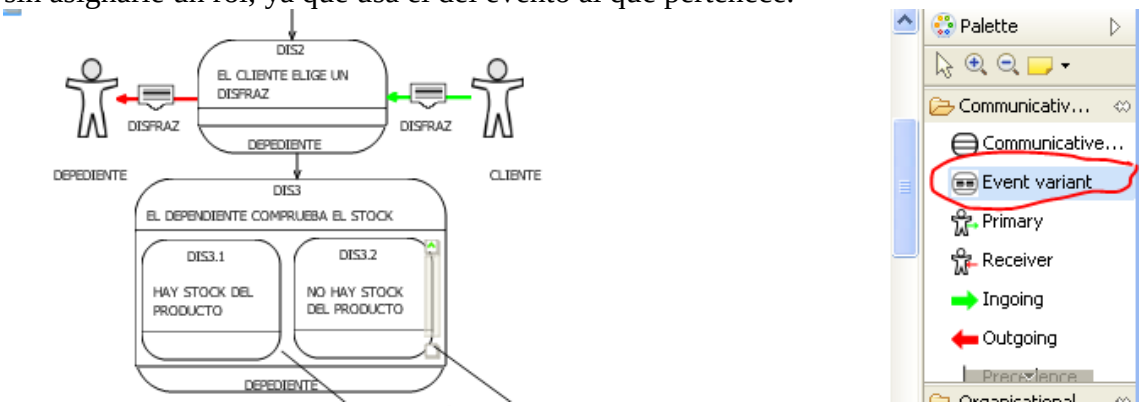


Figura 29. Creación de variantes de evento en Great

El último paso en la creación del esquema es el final. De esta forma, cuando hayamos finalizado el proceso incluiremos un nodo **End** indicando así el fin del diagrama. Para incluirlo hacemos *click* derecho y pulsamos en la parte del diagrama que queramos incluirlo.

Cuando el diagrama esté completo y listo, procederemos a realizar la transformación al diagrama de OO-Method, para hacerlo pulsaremos en el icono **CA to UML transformation** situado en la barra de herramientas superior.

Figura 30. Generación del UML en Great

Si la transformación se ha realizado correctamente debe aparecer un mensaje similar a el siguiente:

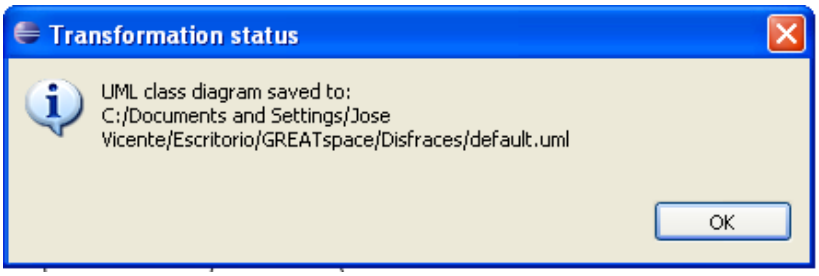


Figura 31. Mensaje de confirmación UML en Great

Cuando tengamos este mensaje se generará dentro del proyecto el archivo **"nombre".uml**, este archivo es similar a un XMI que contiene un listado de los elementos que se han generado. Este archivo nos sirve para generar el diagrama de clases, para ello haremos *click* derecho sobre ese archivo y elegiremos la opción ***Initialize Class* Diagram**.

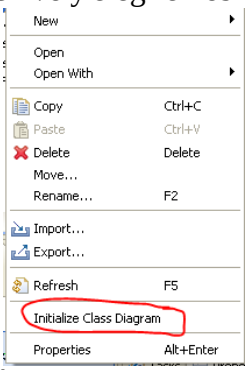


Figura 32. Creación del diagrama de clases en Great

Si el esquema diseñado está completo y es correcto debe ser similar al que hay en la sección anterior llamada Diseño de clases, teniendo en cuenta que en esa imagen estan las clases ordenadas para una mayor estética. Si alguna relación no está correcta seguramente será algún error en el diseño del diagrama de análisis comunicacional. Si hay intención de **modificar el diagrama** de análisis comunicacional, para generar de nuevo el diagrama de clases hay que borrar el archivo .uml, el archivo .umlclass no es necesario eliminarlo, ya que cuando inicializas el .uml se sobreescribe, solo hay que cerrar la ventana.

5.3. Uso de Integranova

Cuando ya tenemos el diagrama de clases en OO-Method listo tendremos que traspasar esa información al segundo programa que utilizaremos, es decir tendremos que diseñar el esquema en IntegraNova. Como se informa en secciones anteriores hay partes en el proceso de transformación que no son automáticas. Por el momento no hay forma de exportar el diagrama de Great a IntegraNova, por lo tanto, para continuar con el proceso hay que copiar toda la información que teníamos en el diagrama de Great a IntegraNova. Este proyecto tiene como enfoque observar y valorar el resultado que obtenemos con la mínima modificación manual desde el diagrama en análisis comunicacional hasta código, así pues, en esta documentación aparecerán los pasos mínimos para representar el diagrama obtenido en Great, para un diagrama de OO-Method completo se necesitan instrucciones más avanzadas que en este proyecto no se busca obtener.

En primer lugar, cuando ya tengamos el programa instalado y serializado procederemos a abrir un nuevo documento, al ser un programa actual no se diferencia de otros para este paso, pulsaremos en la barra superior ***File -> new***.

Cuando ya tengamos el diagrama en blanco procederemos a crear todas las clases que hay en el diagrama, para crear una clase nueva pulsaremos en el siguiente icono situado en la barra de herramientas superior.

Figura 33. Creación de una clase en IntegraNova

Cuando pulsemos en esta y pulsemos en la parte del diagrama que queremos introducirlo nos aparecerá la siguiente ventana, en esta pondremos el nombre y deseleccionaremos el cuadrado de ***Extended Creation***, ya que nosotros queremos indicarle que servicios y que atributos queremos añadir.

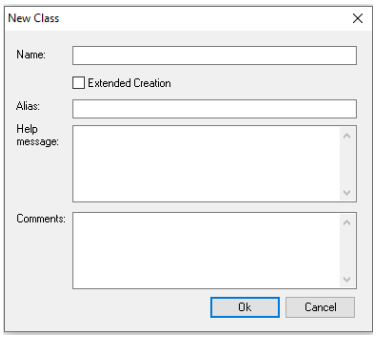


Figura 34. Información de una clase en IntegraNova

En el momento que tengamos las clases listas, procederemos a documentarlas, es decir incluir los atributos y servicios que aparecen en el diagrama de Great. Solo hemos conseguido obtener servicios de creación y los atributos serán los que hemos incluido en las estructuras de mensaje. Todas las clases deben tener un id, de esta forma si alguna clase no tiene asignado uno se crea uno predeterminado.

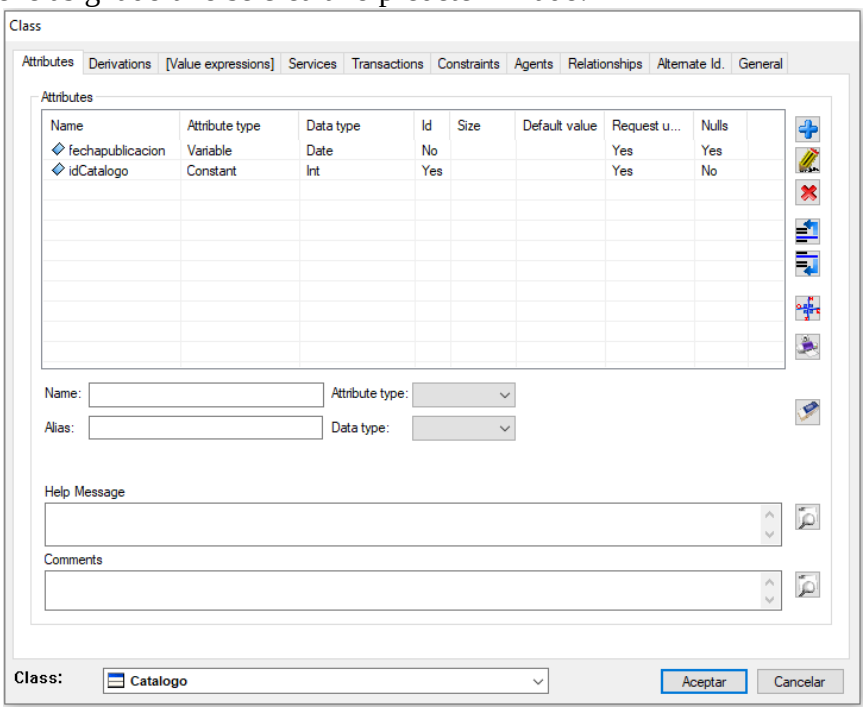


Figura 35. Atributos de una clase en IntegraNova

Como observamos en la imagen toda la ventana es bastante visual, para crear un nuevo atributo pondremos su nombre, tipo y tipo de datos y pulsaremos en el icono de más "+". Para la edición pulsaremos en el atributo y en el icono del lápiz, y para borrarlo pulsaremos en el atributo y en el icono de eliminar "x".

De la misma forma, podremos crear, modificar y eliminar los servicios de la clase, que es la forma de llamar a los métodos que pertenecen a esta. En este caso Great solo ha

generado servicios de creación. IntegraNova nos ofrece una opción para facilitarnos la creación de servicios habituales como son el *new* o el *distroy*, donde incluirá automáticamente al método los atributos necesarios para la creación del servicio.

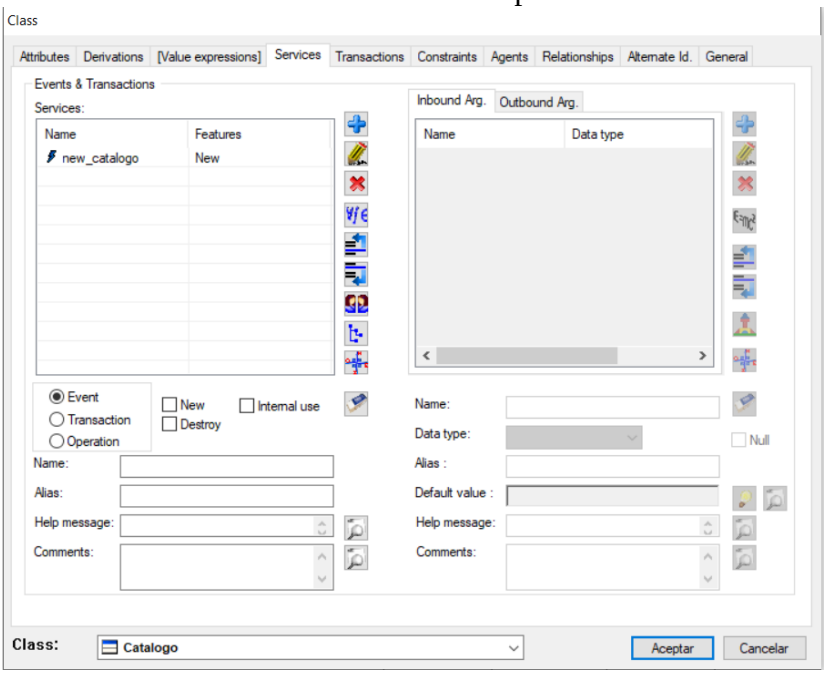


Figura 36. Servicios de una clase en IntegraNova

Para crear los servicios simplemente rellenamos los datos de la parte inferior izquierda tras haber añadido los atributos y seleccionamos la casilla de *new*. De esta forma creará el servicio.

Cuando ya tengamos todas las clases pobladas con los atributos y los servicios que le pertenecen procederemos a establecer las relaciones entre clases existentes. Para ello seleccionaremos a una de las clases, pulsaremos en el icono de la barra de herramientas superior ***new relationship*** y seleccionaremos a la otra clase.

Figura 37. Relaciones entre clases en IntegraNova

De esta forma crearemos la asociación, para establecer las cardinalidades de la relación haremos doble *click* sobre esta e introduciremos la información del diagrama proporcionado en Great.

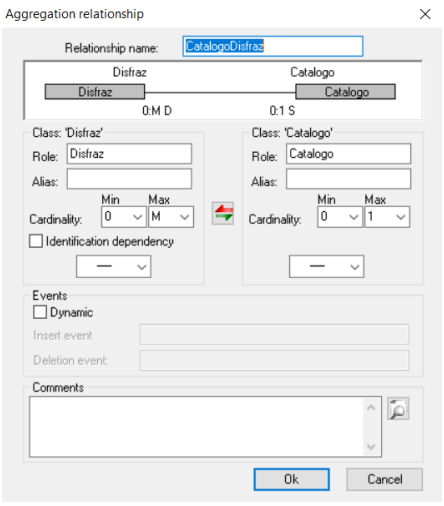


Figura 38. Cardinalidades entre clases en IntegraNova

Por último, tendremos que asignar las clases agente para cada clase del diagrama. Esta es una información que no aparece en el diagrama de Great, pero es indispensable para que el diagrama de IntegraNova esté completo y nos permita generar el código a partir de este.

Para saber el agente asignado a una clase tendremos que observar los roles organizacionales encargados de enviar la estructura de mensaje que contiene la creación de la clase. Es decir, analizaremos en que estructura de mensaje se crea la agregación referente a la clase que estamos observando y el rol organizacional que la envía será el agente asignado a la clase.

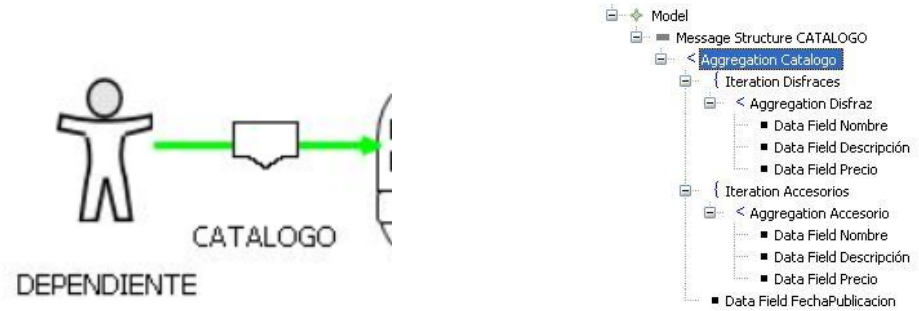


Figura 39. Detección de clases agente

Para asignarle la clase agente correspondiente a la clase pulsaremos en la clase agente, iremos a la pestaña de ***Agents*** y seleccionaremos todos los servicios correspondientes a la clase que queremos asignar ese agente, en nuestro caso como solo hay servicios de creación, pues moveremos el servicio *new*.

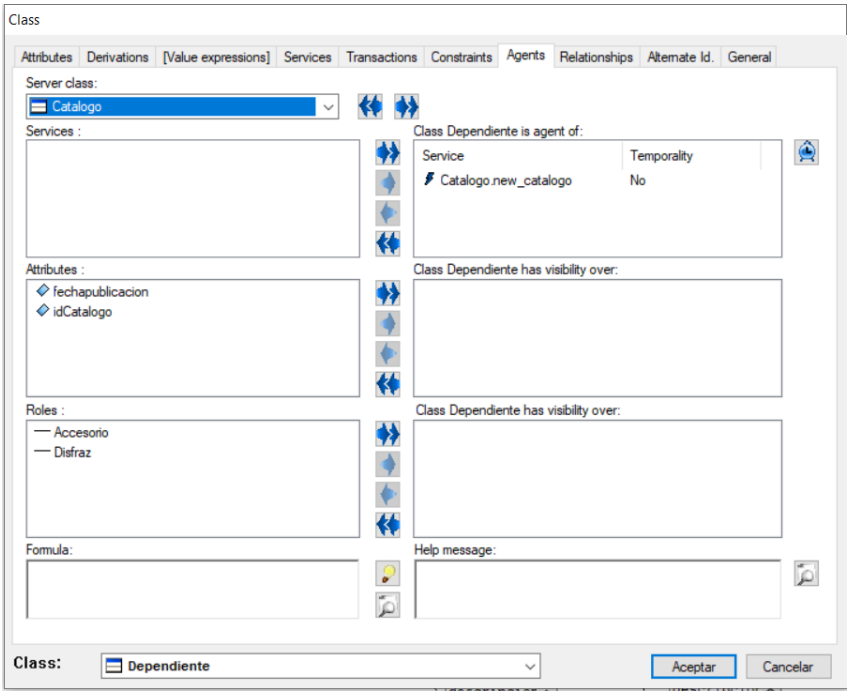


Figura 40. Clases agente en IntegraNova

Cuando ya tengamos el diagrama listo procederemos a la generación de código, para ello seleccionaremos en la parte superior ***Project -> Model Validation*** o pulsaremos Ctrl F5.

5.4. Generación de código

Al validar el modelo empezaremos una serie de pasos que derivaran en el código del programa que se ha ido diseñando durante todo el proceso. En primer lugar, verificaremos que nuestro modelo no tenga ningún error, de esta forma al aparecer la pantalla de validación del modelo tienen que aparecer 0 errores, el número de *warnings* en este caso no es importante ya que estamos diseñando un modelo con lo mínimo para que función, por tanto para que sea completo necesitará más servicios que no hemos incluido.

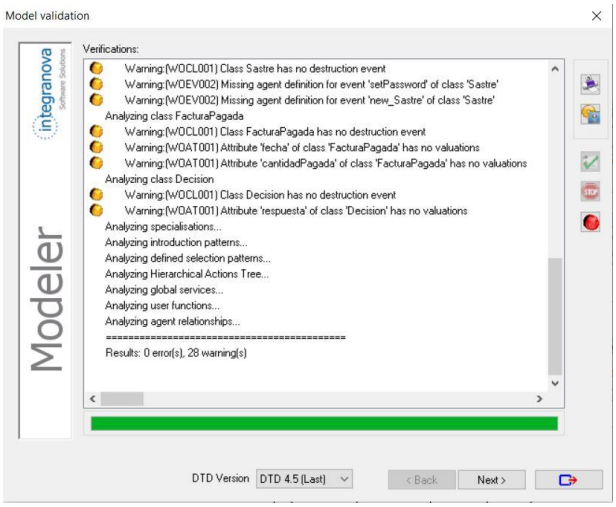


Figura 41. Verificación del modelo

Cuanto no tengamos ningún error en la validación pasaremos al siguiente paso, pulsamos *next*, guardamos el modelo en la carpeta en la que tengamos la información de nuestro sistema y se verificarán y generarán diferentes procesos. Si no hay ningún error aparecerá un mensaje "XML *was correcly generated*".

Cuando pasemos esa ventana pulsando *next* se abrirá una extensión de IntegraNova llamada IntegraNova Star. La primera pantalla que aparecerá mostrará recuadros para introducir información de usuario, contraseña… Para obtener esa información es necesario solicitarle permisos a IntegraNova enviándole el modelo y comentando el proyecto en el que estás trabajando y que ellos validen que puedes usarlo.

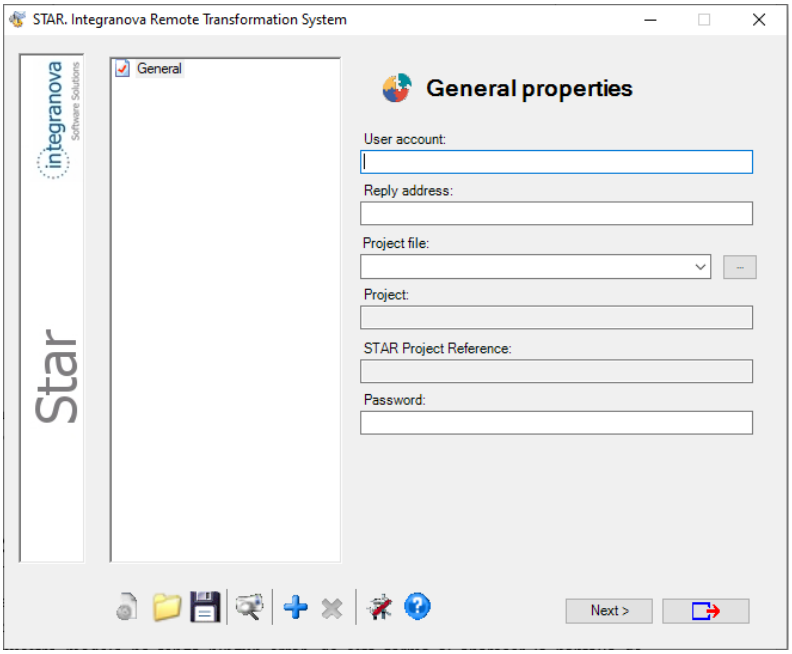


Figura 42. Pantalla de datos IntegraNova Star

Aparte de la información de usuario también habrá que introducir el modelo, el cual seleccionaras desde *Project file* y por supuesto, seleccionar los lenguajes que quieres obtener en forma de código. Para ello al pulsar el icono de "+" en la parte inferior de la pantalla aparecerá una ventana como esta.

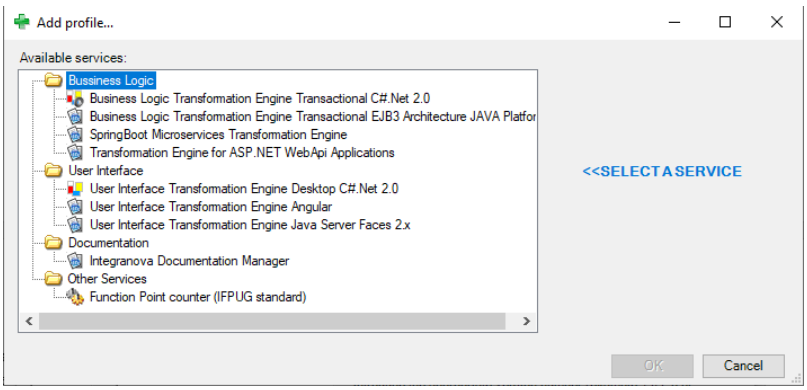


Figura 43. Perfiles de lenguajes en IntegraNova Star

Tras seguir estos pasos conseguimos obtener el código generado en el lenguaje que indiquemos en los perfiles ofertados en la figura 43. La generación de código que ofrece es muy extensa y genera la estructura completa de un proyecto del lenguaje especificado como se explicará de forma más amplia cuando se evalúen los resultados en secciones posteriores.

6. Análisis de resultados

En las secciones anteriores se establecieron unas pautas para la evaluación de la calidad de los resultados obtenidos durante el proceso que ha envuelto a este proyecto. Tras realizar este proceso es momento de aplicar estos controles para realizar la evaluación.

6.1. Resultados de trazabilidad conceptual

En primer lugar, evaluaremos los resultados obtenidos respecto a la trazabilidad conceptual, es decir, evaluaremos la cantidad de conceptos teóricos que la herramienta ha sido capaz de plasmar de una forma concreta y acertada. Para ello en la especificación de la medición se plantearon diferentes formas de evaluar la cantidad de opciones que nos ofrece Great para diseñar diagramas de análisis comunicacional intentando hacerlo de la forma más objetiva posible.

Por un lado, podemos afirmar que con la herramienta Great se pueden diseñar diagramas tan completos como los diagramas que existen en las documentaciones que circulan por la red y han sido proporcionadas al alumno por el grupo PROS a excepción de una funcionalidad concreta que no se ha podido diseñar de la misma forma que observamos en los diagramas de análisis comunicacional proporcionados.

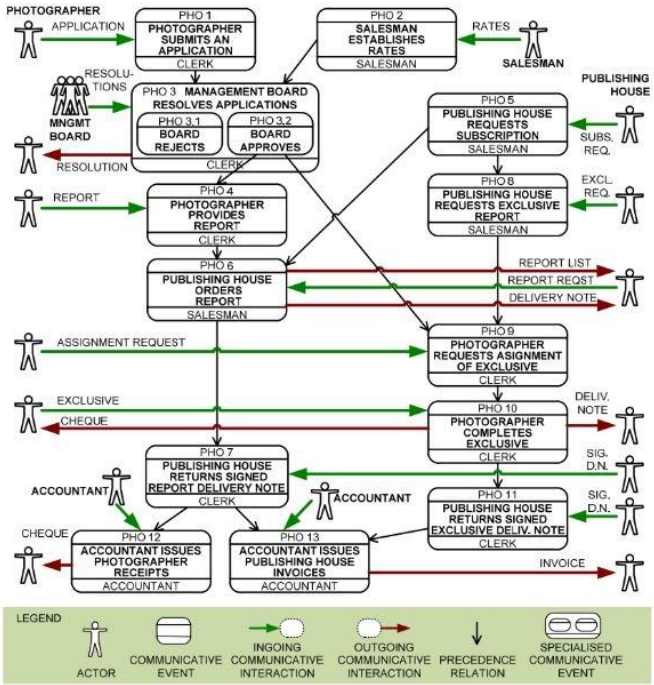


Figura 44. Ejemplo de diagrama análisis comunicacional

En la imagen podemos apreciar un diagrama similar al de esta tesis, a diferencia que en los actores del sistema encontramos uno llamado “NMGMT BOARD” donde se refiere a un grupo de actores en vez de a un actor en concreto. En Great la forma de plasmar eso sería indicar ese grupo de actores como un actor y especificarlo en el nombre o en la posterior documentación del esquema.

Valorando la información expuesta podemos afirmar que Great nos ofrece una buena cobertura de las funcionalidades necesarias para realizar diseños de diagramas de análisis comunicacional. No solo siendo capaz de realizar transformaciones, sino también ofreciendo una forma válida de diseñar los esquemas para exponer cuestiones teóricas.
Por otro lado, también podemos observar la utilidad de Great como herramienta para verificar la validez de un diagrama de análisis comunicacional. Como aparece en la especificación de la medición, durante la primera parte del proyecto donde se ha usado Great se han realizado diferentes diagramas para observar los comportamientos de la herramienta cuando se introducen las diferentes funcionalidades. De esta forma, se observa que cuando un diagrama tiene algún error o está incompleto no es capaz de generar un diagrama de OO-Method válido y acorde a la información que le habíamos introducido al diagrama de análisis comunicacional.

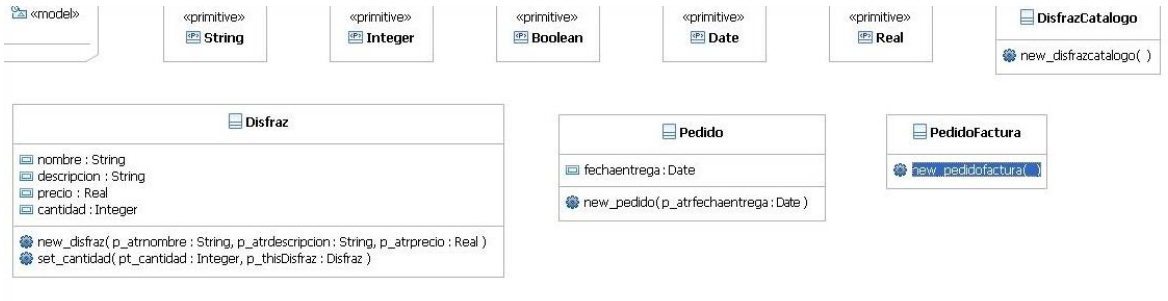


Figura 45. Diagrama con errores en Great

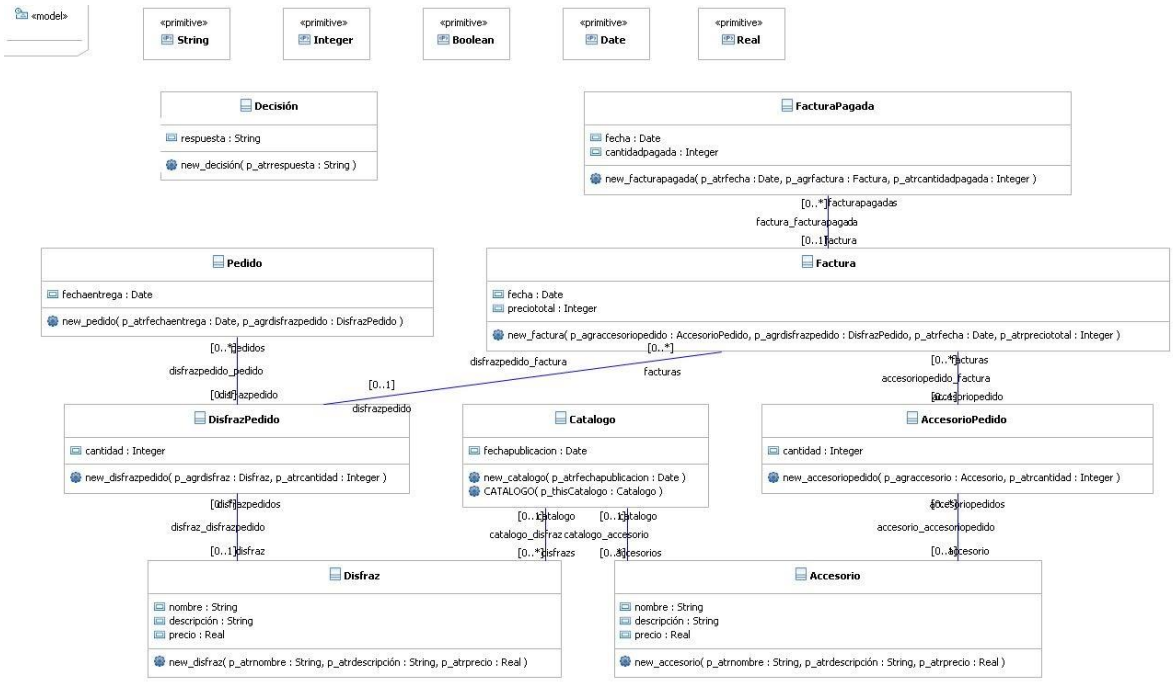


Figura 46. Diagrama correcto en Great

En las imágenes podemos apreciar los diagramas generados para dos esquemas de análisis comunicacional con los mismos eventos, pero donde en el primero había un error de diseño y en el segundo ese error está solucionado y revisado. Asimismo, se verifica el cumplimiento de la funcionalidad especificada y se puede afirmar que Great puede

facilitar el diseño de diagramas de análisis comunicacional no solo como herramienta de diseño, sino también como herramienta de validación.

Tras estas valoraciones estas es la evaluación de las métricas en esta sección:

- número de eventos comunicativos trazables desde CA a funcionalidades del sistema / número de eventos comunicativos totales = 10 / 12 = 0,833

Con esta métrica valoramos la calidad del modelo referente a las diferentes funcionalidades mostradas en este como por ejemplo en la figura 44 podemos observar el evento PHO3 podemos apreciar cómo se introduce la bifurcación introduciendo dos caminos, sin embargo, en el evento PHO2 no se introduce ninguna funcionalidad de análisis comunicacional. Es decir, este modelo trata de ser una muestra de todo lo que ofrece análisis comunicacional y prácticamente en cada evento comunicativo hay una funcionalidad importante introducida.

6.2. Resultados de la evaluación de brechas funcionales

Para verificar la correcta transformación desde el esquema comunicativo de eventos hasta el diagrama de clases el OO-Method nos proporciona unas reglas de transformación introducidas por Sergio España en su tesis (España, 2011). Estas reglas corresponden a la evaluación del modelo de negocio, el cual es la parte que se pretende observar el funcionamiento y valorar su eficacia.

Regla OM1. Determinación del alcance de la derivación.

- Definición de cuáles van a ser los eventos comunicacionales que se van a incluir en el diagrama de eventos y la estructura que van a tener desde el nodo inicial hasta el fin del diagrama.
- **¿Se cumple?** Sí se cumple, ya que el diagrama que se ha creado es completo y con sentido, con todas las relaciones indicadas desde el nodo inicial hasta el nodo final. En la figura 47 podemos ver un diagrama de eventos comunicacional completo diseñado en la herramienta Great.

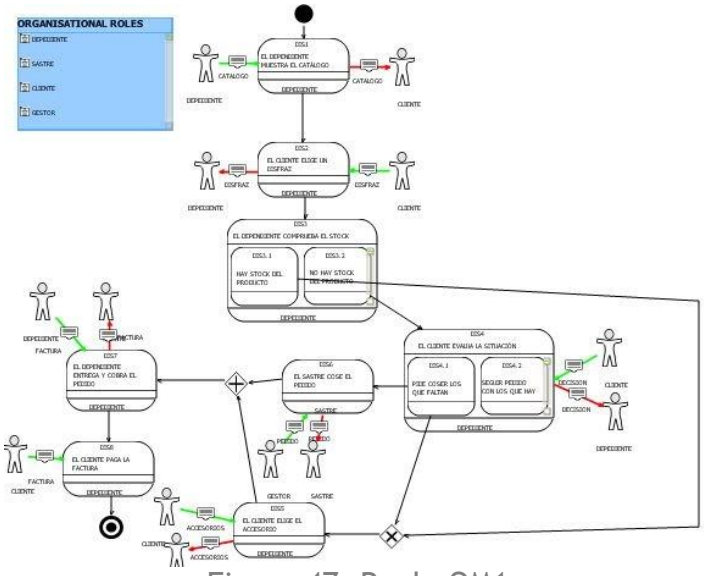


Figura 47. Regla OM1

Regla OM2. Identificación de los *reference field*

- Capacidad de referenciar desde un objeto a otro mediante la creación de un *reference field*. Aumentando la reutilización y facilitando el diseño.
- **¿Se cumple?** Sí se cumple. Cuando se crea la estructura del mensaje entre un actor y el evento comunicacional, dentro de una agregación se puede crear un *reference field*. Dentro de las propiedades de este podemos encontrar el campo *Domain*, el cual al pulsarlo aparece un desplegable con las opciones a referenciar, como se puede apreciar como en la Figura 48.

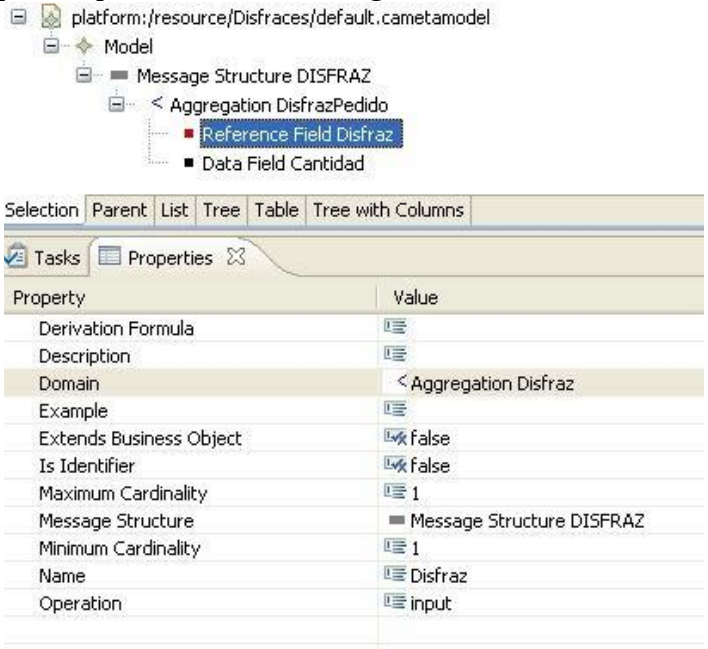
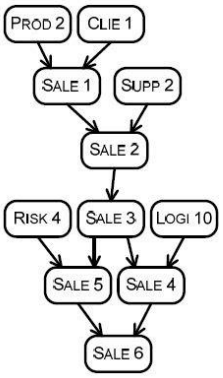


Figura 48. Regla OM2

Regla OM3. Ordenación de eventos

- Capacidad de crear una lista con las posibles ordenaciones de los eventos comunicacionales a partir del diagrama de eventos comunicacional.
- **¿Se cumple?** No se puede verificar debido a que son cálculos que se realizan internamente. La Figura 49 es un ejemplo de cómo se podrían generar las posibilidades, pero en Great no se puede visualizar.



PROD 2, CLIE 1, SALE 1, SUPP 2, SALE 2, RISK 4, SALE 3, LOGI 10, SALE 5, SALE 4, SALE 6
CLIE 1, PROD 2, SALE 1, SUPP 2, SALE 2, RISK 4, SALE 3, LOGI 10, SALE 5, SALE 4, SALE 6
PROD 2, CLIE 1, SUPP 2, SALE 1, SALE 2, RISK 4, SALE 3, LOGI 10, SALE 5, SALE 4, SALE 6
CLIE 1, PROD 2, SUPP 2, SALE 1, SALE 2, RISK 4, SALE 3, LOGI 10, SALE 5, SALE 4, SALE 6
PROD 2, SUPP 2, CLIE 1, SALE 1, SALE 2, RISK 4, SALE 3, LOGI 10, SALE 5, SALE 4, SALE 6
CLIE 1, SUPP 2, PROD 2, SALE 1, SALE 2, RISK 4, SALE 3, LOGI 10, SALE 5, SALE 4, SALE 6
SUPP 2, PROD 2, CLIE 1, SALE 1, SALE 2, RISK 4, SALE 3, LOGI 10, SALE 5, SALE 4, SALE 6
SUPP 2, CLIE 1, PROD 2, SALE 1, SALE 2, RISK 4, SALE 3, LOGI 10, SALE 5, SALE 4, SALE 6
PROD 2, CLIE 1, SALE 1, SUPP 2, SALE 2, SALE 3, RISK 4, LOGI 10, SALE 5, SALE 4, SALE 6
CLIE 1, PROD 2, SALE 1, SUPP 2, SALE 2, SALE 3, RISK 4, LOGI 10, SALE 5, SALE 4, SALE 6
PROD 2, CLIE 1, SUPP 2, SALE 1, SALE 2, SALE 3, RISK 4, LOGI 10, SALE 5, SALE 4, SALE 6
CLIE 1, PROD 2, SUPP 2, SALE 1, SALE 2, SALE 3, RISK 4, LOGI 10, SALE 5, SALE 4, SALE 6
PROD 2, SUPP 2, CLIE 1, SALE 1, SALE 2, SALE 3, RISK 4, LOGI 10, SALE 5, SALE 4, SALE 6
CLIE 1, SUPP 2, PROD 2, SALE 1, SALE 2, SALE 3, RISK 4, LOGI 10, SALE 5, SALE 4, SALE 6
SUPP 2, PROD 2, CLIE 1, SALE 1, SALE 2, SALE 3, RISK 4, LOGI 10, SALE 5, SALE 4, SALE 6
SUPP 2, CLIE 1, PROD 2, SALE 1, SALE 2, SALE 3, RISK 4, LOGI 10, SALE 5, SALE 4, SALE 6
PROD 2, CLIE 1, SALE 1, SUPP 2, SALE 2, SALE 3, LOGI 10, RISK 4, SALE 5, SALE 4, SALE 6

Figura 49. Regla OM3

Regla OM4. Derivación de una nueva clase a partir de una agregación

- Toda agregación introducida debe generar una clase nueva a partir de ella, a excepción de que especifique que esa agregación es una extensión de otra agregación.
- **¿Se cumple?** Sí se cumple. Cuando en una estructura de mensaje introduces una agregación, como se aprecia en la primera parte de la figura 50, al generar el diagrama de clases esta agregación aparece en el diagrama de clases con toda la información que se ha introducido en su interior, como en la segunda parte de la figura 50. También se cumple la excepción. Esto se verifica cuando al introducir un *reference field* en una agregación y poner el campo *Extends business project* en *True* se generan los atributos de la clase de la agregación dentro de la clase referenciada.

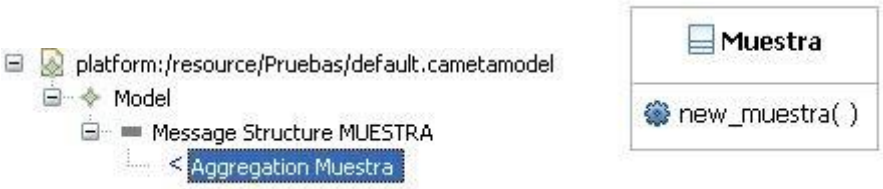


Figura 50. Regla OM4

Regla OM5. Derivación de un atributo a partir de un *data field*

- Todo *data field* introducido debe generar atributos dentro de la agregación en la que está incluido.
- **¿Se cumple?** Sí se cumple. Al igual que en la regla anterior cuando introduces un *data field* dentro de una agregación se genera un atributo dentro de la clase de la agregación. En este caso también está la excepción del *reference field* nombrada anteriormente, cuando en la misma clase que está el *data field* hay un *reference field* estos atributos se generan en la clase referenciada.

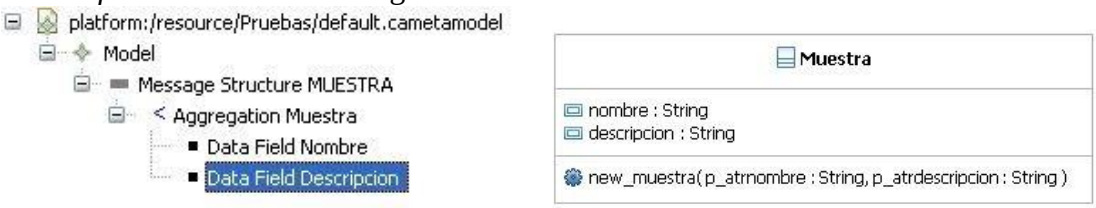


Figura 51. Regla OM5

Regla OM6. Selección del identificador de clases

- A partir de cada evento comunicativo se genera un id correspondiente en el diagrama de clases.
- **¿Se cumple?** No he encontrado ningún identificador ni en las clases ni en los atributos. En el diagrama de eventos comunicacional si encontramos un identificador para los eventos, pero luego no se ve reflejado en el diagrama de clases. Esta regla está más enfocada a la extracción de los identificadores a partir del texto que se quiere trabajar.

Regla OM7. Especificación del tipo de un atributo

- Posibilidad de identificar un *data field* como una constante o variable en función a las propiedades que tiene y posibilidades de modificación.
- **¿Se cumple?** No hay ninguna evidencia que se pueda identificar como constante o variable. Como se puede apreciar en la figura 52, en el *data field* existe un campo

para determinar si es identificador o no, pero posteriormente en el diagrama de clases tampoco hay ninguna diferencia entre los atributos que son identificadores y los que no. Respecto a la posibilidad de modificar el valor no es posible verificarlo, no se está trabajando con valores, en la figura 53 se pueden observar los valores que se pueden modificar dentro del diagrama de clases.

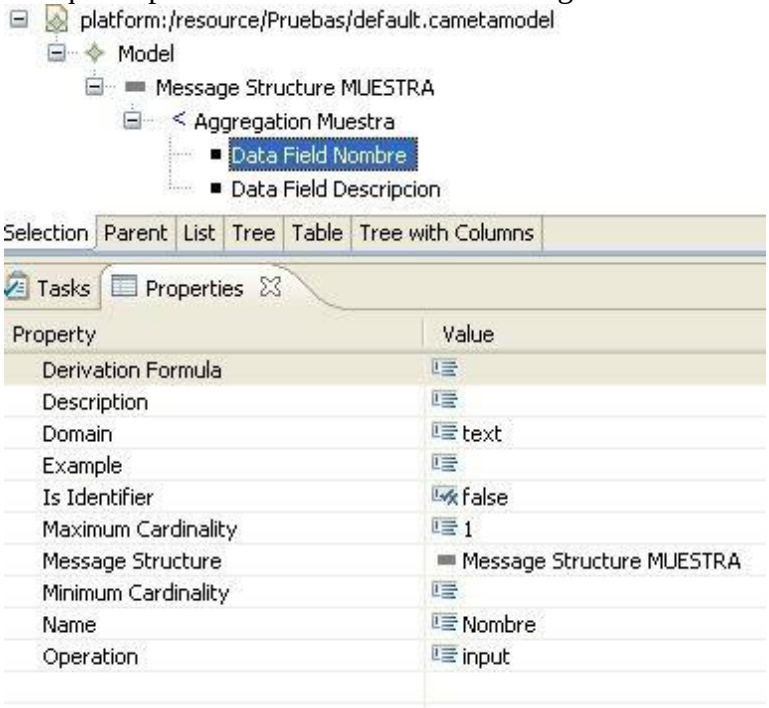


Figura 52. Regla OM7 análisis comunicacional

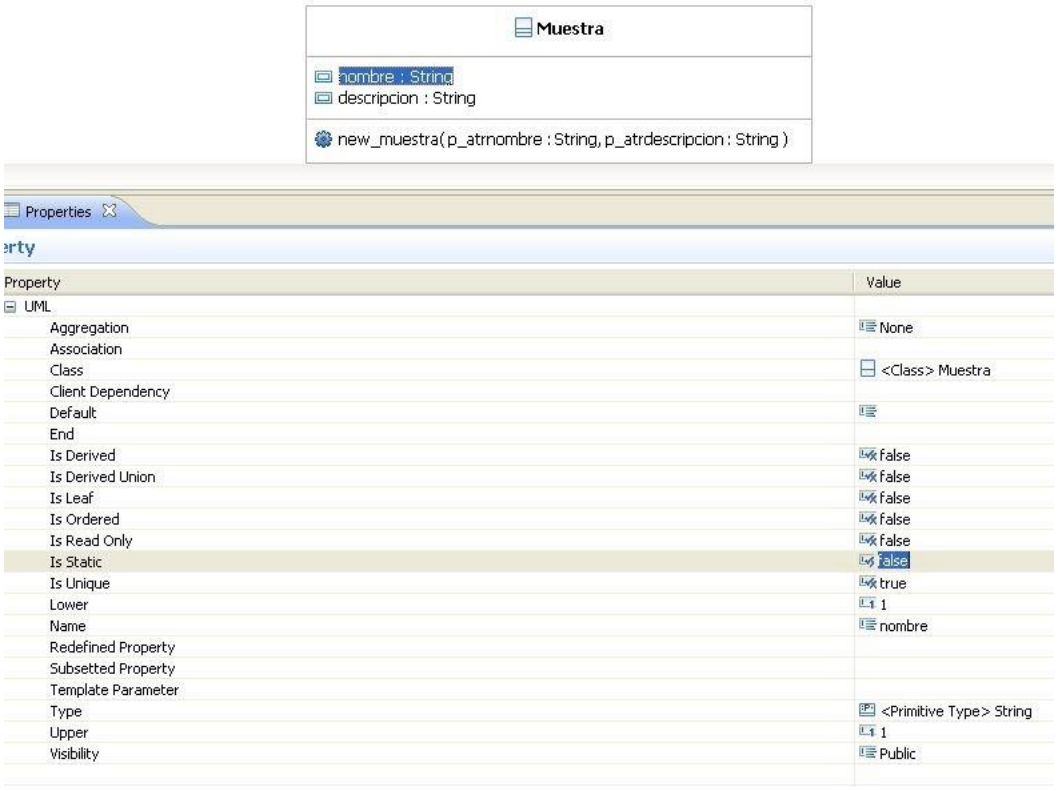


Figura 53. Regla OM7 OO-Method

Regla OM8. Especificación del tipo de un atributo

- Posibilidad de identificar a un atributo un tipo concreto. Dentro del OO-Method existen dominios para indicar el tipo al que pertenece un data field en un diagrama comunicacional, al transformarlo en un diagrama de clases estos dominios se transforman en tipos utilizados en leguajes de programación de alto nivel. En la figura 54 podemos observar la tabla de conversión entre los dominios del diagrama comunicacional y los tipos del atributo.

Communication Analysis Data field domain	OO-Method Attribute data type
number	Nat, Int, Real, Autonumeric
text	String, Text
date, time	Time, Date, DateTime
money	Real
not considered	Bool, Image, Blob

Figura 54. Regla OM8

- **¿Se cumple?** Sí se puede. Cuando se crea un *data field* en sus propiedades hay un campo *Type* donde se pueden indicar las diferentes posibilidades que proporciona el método, observable en la figura 55. Al generar el diagrama de clases los atributos que se generan tienen el tipo que se ha indicado previamente.

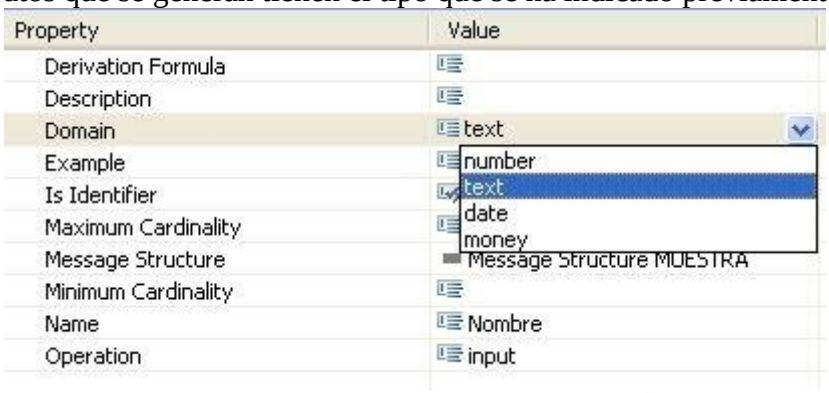


Figura 55. Regla OM8 comprobación

Regla OM9. Decisión sobre si se solicita un atributo en el momento de la creación

- Cuando se especifican las propiedades de un *data field* debe existir la posibilidad de indicar de que sea requerido. Es decir, cuando se crea el atributo derivado será necesario indicar el valor de este.
- **¿Se cumple?** No se ha encontrado ningún campo respecto a la posibilidad de requerir el valor en el diagrama de eventos comunicacional, como se observa en la figura 54. Al transformarlo en el diagrama de clases, como se aprecia en la figura 55 todos los *data field* que pertenecían a la agregación son necesarios para crear el objeto, pero no hay forma de indicar que no sea necesario su uso.

Regla OM10. Decisión sobre si se permite valores nulos

- Cuando se especifican las propiedades de un *data field* debe existir la posibilidad de indicar de que admita valores nulos.
- **¿Se cumple?** No se ha encontrado ningún campo respecto a la posibilidad de admitir valores nulos en los *data field* en el diagrama de eventos comunicacional. Al transformarlo en el diagrama de clases tampoco aparece ningún campo. En figura 50 e figura 51 se pueden observar las propiedades de los *data field* y de los

atributos correspondientes, y en ningún caso aparece nada relacionado con el valor nulo de estos.

Regla OM11. Derivación de una relación estructural a partir de subestructuras anidadas

- Dado que una agregación es parte de otra, las clases correspondientes generadas están relacionadas por medio de una relación estructural. Además, cuando otra agregación es una iteración, entonces la relación estructural tiene la máxima cardinalidad * del lado de la clase iterada (la iteración específica que puede tener muchas repeticiones). Dado que no se pueden derivar cardinalidades de las restricciones estructurales.
- **¿Se cumple?** Sí se cumple. Cuando dos agregaciones están relacionadas por la inclusión de una dentro de la otra se genera una relación entre estas. Cuando esta relación se debe a que una es una iteración dentro de la otra la relación es de 0:1 a 0:*. En la figura 56 se ve un caso como el nombrado, en este caso hay dos agregaciones iteradas dentro de la agregación principal. Al generar el diagrama de clases se crean las clases de la segunda imagen de la figura 56, en ambas vemos que se cumplen las condiciones nombradas.

Figura 56. Regla OM11

Regla OM12. Derivación de una relación estructural a partir de un *reference field*

- Al incluir una agregación dentro de otra por medio de un *reference field* y generar el diagrama de eventos comunicacional, la agregación referenciada tendrá una cardinalidad máxima de 1, por otra parte, la otra agregación tendrá una cardinalidad 0:*.
- **¿Se cumple?** Sí se cumple, cuando se incluye un *refence field* dentro de una agregación las cardinalidades que se generan en el diagrama de clases es de 0:1-0:*. Un ejemplo de esta transformación se aprecia en la figura 57 cuando se introduce un *reference field* dentro de una agregación, al realizar la transformación, se cumple la condición especificada, observable en la segunda parte de la figura 57.

Figura 57. Regla OM12

Regla OM13. Inclusión de un servicio de creación a una clase recién creada

- Posibilidad de incluir un servicio de creación cuando generas una clase a partir de una agregación del diagrama de eventos comunicacional, a excepción de que la agregación sea incluida dentro de otra.
- **¿Se cumple?** Sí se cumple. Cuando se genera el diagrama de clases, toda agregación que no sea incluida por otra se crea un método *new* generador de objetos. En la figura 48 y la figura 49 podemos ver como cuando se crea una agregación y se genera el diagrama de clases, la clase relacionada a esa agregación tiene un servicio de creación.

Regla OM14. Inclusión de un servicio de fin o edición para mensajes complejos

- Posibilidad de incluir un servicio de edición o de fin cuando generas una clase a partir de una agregación del diagrama de eventos comunicacional.
- **¿Se cumple?** No se ha encontrado ninguna forma de generar métodos para editar y borrar objetos. Tampoco hay forma de verificar que existan estos servicios.

Regla OM15. Selección de una clase extendida

- Cuando estamos creando una clase mediante una agregación en el diagrama de eventos comunicacional, tenemos la opción de indicar que extiende un *bussiness object*, de esta forma esta agregación formará parte de la agregación en la que se introduce la agregación extendida. Al generar el diagrama de clases la clase donde se introduce el *reference field* formará parte de la extendida y todos sus atributos también pasarán a formar parte.
- **¿Se cumple?** Sí se cumple. Cuando introduces en una agregación un *reference field* con la opción *Extends bussiness object* en *True*, al generar el diagrama de clases todos los atributos que están en la agregación referenciadora forma parte a la clase de la agregación referenciada. Esto se puede observar en figura 58 cuando se introduce el *reference field* con la propiedad nombrada en *True*, se genera una clase con las características nombradas al principio del párrafo, como se aprecia en la segunda parte de la figura 58.

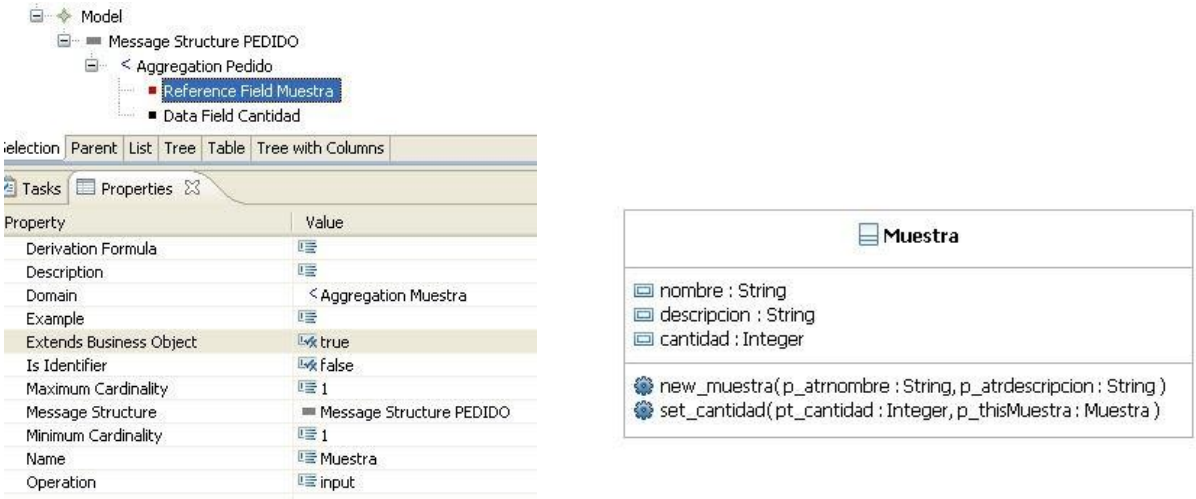


Figura 58. Regla OM15

Regla OM16. Inclusión de un servicio de edición de una clase extendida

- Posibilidad de incluir un servicio de edición cuando generas una clase extendida cuando generas el diagrama de clases a partir del diagrama de eventos comunicacional.
- **¿Se cumple?** No se ha encontrado ninguna forma de generar métodos para editar objetos. Tampoco hay forma de verificar que existan estos servicios. Cuando se referencia una clase dentro de otra se añade el servicio set_* que se utiliza como constructor de la clase que se introduce en la otra. Esta propiedad de Great se puede apreciar en figura 51.

Regla OM17. Definición de una transacción

- Posibilidad de incluir una transacción en nuestro diagrama. Una transacción es un intercambio de flujo atómico, es decir, que se debe realizar en el mismo espacio de tiempo debido a que es información crítica, por tanto, debe activar los servicios de los atributos que quiera modificar.
- **¿Se cumple?** No se ha encontrado ninguna evidencia de que se puedan generar servicios asignados a editar atributos, por tanto, no se puede realizar transacciones con la herramienta Great.

Regla OM18. Determinación de los contactos y eventos de reacción

- Dentro del diagrama de eventos comunicacional a los actores se les nombra como *Organisation roles*, a partir de estos se generan los contactos al transformarlos en el diagrama de clases. Los eventos de reacción es información que se obtiene a partir del texto que se quiere trabajar utilizada para dar permiso a los contactos.
- **¿Se cumple?** No se ha encontrado ninguna forma de verificar que se introducen los contactos ni que se puedan introducir eventos de reacción.

Regla OM19. Introducción de una clase agente a partir de un rol organizacional

- Capacidad de generar una clase a partir de un rol organizacional introducido en el diagrama de eventos comunicacional.
- **¿Se cumple?** No se ha encontrado forma de generar clases agentes a partir de roles organizacionales. Las clases agente se deberán introducir manualmente a posteriori.

Regla OM20. Derivación de relaciones de agentes para roles de interfaz

- Posibilidad de introducir en el diagrama de clases relaciones entre agentes.
- **¿Se cumple?** No se ha encontrado forma de relacionar roles organizacionales para que al generar el diagrama de clases se introduzca una relación.

Regla OM21. Creación de un agente para todo el sistema

- Posibilidad de introducir un actor Administrador que se asocie a la clase principal como agente y que esta clase se extienda a todo el sistema.
- **¿Se cumple?** No hay posibilidad de asignar un administrador a todo el sistema ni se ha encontrado forma de relacionar roles organizacionales para que al generar el diagrama de clases se introduzca una relación. Los roles se introducirán manualmente a posteriori.

Tras mostrar los resultados obtenido en las diferentes reglas procederemos a evaluar las métricas:

- Número de reglas que se cumplen / Número de reglas totales = 9 / 19 = 0.47

Esta métrica muestra cuanto se ajusta Great y su generación a las premisas establecidas en análisis comunicacional y como observamos en la evaluación de la métrica el número es bajo para lo que debería ser un programa que trata de ceñirse estrictamente a esas premisas. El número total es de 19 debido a que hay 2 que no se puede verificar y por tanto no se han incluido. Destacar que el método original de transformación señala que no puede ser automatizado completamente, por lo que se requieren actividades adicionales para alcanzar la transformación de todas las reglas.

6.3. Resultados del modelo generado

Para evaluar los resultados del modelo generado, es decir el modelo de OO-Method obtenido de la transformación del modelo de análisis comunicacional generado a partir de la herramienta Great, realizaremos una revisión de la información obtenida durante el proceso de transformación y del modelo transformado en sí y lo analizaremos.

Por una parte, analizando el diagrama obtenido en el proceso de esta tesis respecto al cumplimiento de las reglas básicas de OO-Method podemos destacar algunos aspectos relevantes. Por regla general podemos observar que el diagrama contiene todos los aspectos básicos que un diagrama de OO-Method debe tener, con las clases, asociaciones y cardinalidades correctamente generadas y documentadas. Sin embargo, hay una funcionalidad que o no está implementada o Great no nos proporciona una forma clara de poder visualizarla. Esta funcionalidad se trata de las clases agente y los agentes del sistema, un punto que para poder integrarlo con IntegraNova al trasladar la información del modelo Great no ofrece ninguna opción para visualizar qué clase agente está relacionada con cada clase, creando de esta forma incongruencias entre un modelo y el otro y obligando al usuario a necesitar documentarse y examinar el diagrama de análisis comunicacional para añadir nueva información a la que nos proporciona el diagrama anterior.

Por otra parte, realizando un análisis de la completitud y de la validez del diagrama generado se han obtenido valoraciones positivas. Como comento es toda una hazaña poder obtener un diagrama prácticamente válido de una tecnología a partir de otra y el diagrama genera las clases y las relaciones entre clases que el usuario pretenda crear. Sin embargo, en el modelo generado encontramos varias ausencias que parecen imprescindibles a la hora de diseñar un modelo de OO-Method completo. Por un lado, las clases agente comentadas anteriormente, las cuales se pueden obtener observando el diagrama de análisis comunicacional, pero se puede considerar una ausencia en ese diagrama. Por otro lado, no se ha encontrado forma de crear servicios que no sean los de creación de la clase, es decir, no se pueden generar ni servicios de edición ni de borrado. Este es otro de los puntos que ha generado más indiferencia ya que un modelo completo no solo tiene servicios de creación, también tiene añadidos de servicios de modificación, ya que la información en un sistema normalmente no solo se crea, sino que va evolucionando y cambiando a lo largo de un proceso industrial.

Como he indicado en la especificación de la medición, para realizar la evaluación de esta parte también se va a utilizar el *Gold Standard*, una forma de medición propuesta por el grupo PROS que presenta los casos de uso que debería tener el sistema para gestionar las entidades de información involucradas en el proceso de negocio descrito. Estos casos de uso se contrastan con las actuales funcionalidades del modelo generado para valorar la calidad de generación que tienen las herramientas utilizadas. Está medición está compuesta de un diagrama de casos de uso con los casos de uso directos en color blanco y los derivados en un tono amarillo, después hay una tabla explicando en que consiste cada caso de uso. Esa información servirá para evaluar el sistema con conclusiones y métricas extraídas.

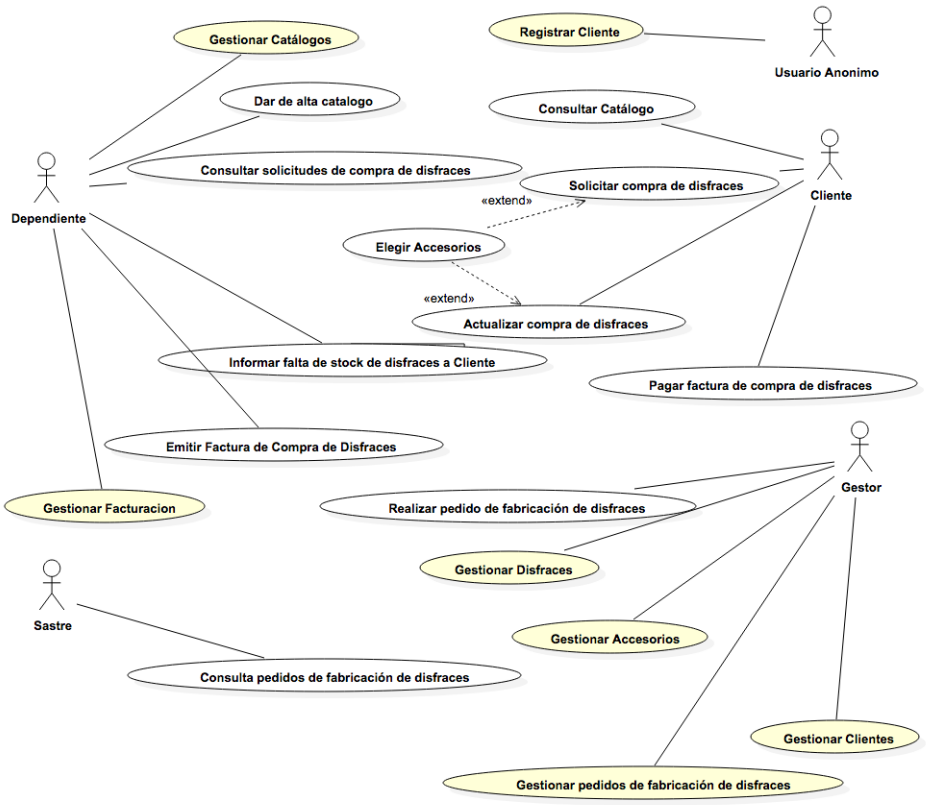


Figura 59. Diagrama de casos de uso para el Gold Standard

Caso de Uso	Propósito	Operaciones y clases
Registrar Cliente	Permite a un usuario anónimo darse de alta el sistema, para que pueda acceder a la funcionalidad de los Clientes.	Crear Cliente
Dar de alta catálogo	Permite al Dependiente crear un nuevo catálogo de disfraces y accesorios, que pueda ser consultado en línea por los Clientes.	Crear Catálogo
Gestionar catálogos	Permite al Dependiente consultar, editar y eliminar catálogos existentes	Consultar, editar, eliminar Catálogo
Consultar Catálogo	Permite a los Clientes ver los disfraces y accesorios disponibles	Consultar Catálogo
Solicitar compra de disfraces	Permite al Cliente crear una nueva orden de compra de disfraces.	Crear compra
Consultar solicitudes de compra de disfraces	Permite al Dependiente consultar las solicitudes de compra de disfraces, y ver el stock disponible para evaluar si es posible satisfacerla sin requerir fabricar nuevos disfraces.	Consultar compra Consultar stock
Elegir Accesorios	Permite al Cliente especificar los accesorios a incluir para una compra de disfraces.	Editar compra (para agregar accesorios)
Informar falta de stock al cliente	Permite al dependiente informar al cliente de falta de stock y ofrecer las opciones de acotar el pedido a las existencias, o requerir fabricación de los disfraces faltantes (lo que aumentaría el tiempo de entrega)	Editar compra (cambiar de estado)
Actualizar compra de disfraces	Permite al Cliente editar una orden original, actualizando la cantidad de disfraces pedidos de acuerdo a las limitaciones de stock informadas, o confirmando que desea ordenar y esperar la fabricación de las unidades restantes.	Editar compra (disfraces, catálogos, cambiar de estado)
Emitir factura de compra de disfraces	Permite al Dependiente crear y enviar la factura de compra de disfraces.	Crear factura Editar compra (cambiar de estado)
Pagar factura de compra de disfraces	Permite al Cliente pagar la factura y así finalizar el proceso de compra	Editar factura (cambiar de estado) Editar compra (cambiar de estado)

Gestionar facturación	Permite consultar facturas emitidas y pagadas, eliminar y editar facturas no pagadas.	Consultar, editar, eliminar facturas
Realizar pedido de fabricación de disfraces	Permite al Gestor crear un nuevo pedido de fabricación de disfraces, para ser recibido y procesado por el Sastre.	Crear pedido de fabricación de disfraces.
Consultar pedidos de fabricación de disfraces	Permite al Sastre ver los pedidos de fabricación de disfraces realizados por el Gestor.	Crear pedido de fabricación de disfraces.
Gestionar Disfraces	Permite consultar, crear, editar y eliminar los disfraces existentes.	Crear, editar, eliminar, consultar Disfraces.
Gestionar Accesorios	Permite consultar, crear, editar y eliminar los accesorios existentes.	Crear, editar, eliminar, consultar Accesorios
Gestionar Cliente	Permite consultar, crear, editar y eliminar los clientes existentes.	Crear, editar, eliminar, consultar Clientes
Gestionar pedidos de fabricación de disfraces	Permite consultar, editar y eliminar los pedidos de fabricación de disfraces existentes.	Editar, eliminar, consultar pedidos de fabricación de disfraces

Como podemos observar en el diagrama de casos de uso aparecen muchas más funcionalidades de las que aparecen en el modelo de OO-Method, eso muestra las actuales ausencias que encontramos en la generación de Great, donde solo se pueden generar servicios de creación, pero es necesaria una actualización de Great para que permita generar otro tipo de servicios.

A continuación, procederemos a evaluar las métricas establecidas en esta sección:

- número de conceptos del modelo generado / número total de conceptos del modelo de IntegraNova = 14 / 15 = 0.93
- casos de uso implementados / casos de uso totales = 9 / 18 = 0.5

Los resultados en estas métricas son interesantes porque nos muestran dos variantes de información diferentes pero que son importantes a destacar. En primer lugar, la primera métrica nos muestra que el modelo que se ha generado es prácticamente suficiente para ser un modelo válido, ya que solo le falta tener clases agentes para que sea un modelo completo y usable para los siguientes pasos. Sin embargo, la otra métrica nos muestra que, aunque sea un modelo válido, este modelo no tiene todas las funcionalidades que se necesitan para que sea un programa completo y usable en un sistema de información, ya que solo se han creado servicios de creación y para crear un entorno válido para manejar información hacen falta servicios de todo tipo. Considerando el método aplicado, la conclusión es que, para generar la totalidad de los casos de uso, se requiere modelar otros

procesos en GREAT, como, por ejemplo, el registro de nuevos disfraces y accesorios, que están fuera del alcance del proceso originalmente modelado.

6.4. Resultados del código generado

Respecto a los resultados obtenidos del código generado a partir del diagrama de OO-Method diseñado en IntegraNova, de la misma forma que con los resultados obtenidos en la sección anterior hay valoraciones contrapuestas.

Respecto a la completitud y adaptación a las reglas del lenguaje de programación, encontramos que el código que genera es de una gran calidad y en su mayor parte sin errores en la parte de compilación sobre todo en la parte del frontend. En la parte de backend hay algunas generaciones, como la de .net que prácticamente no generan errores, y los que generan son de incompatibilidades con bibliotecas de Visual Studio. Por otra parte, hay otras generaciones como las de C# que solo al generar ya tienen muchos errores de compilación que deben ser corregidos para la viabilidad del código. Estos problemas son en gran parte dependencias técnicas, ya que la documentación es limitada.

Sin embargo, el proyecto completo que se genera es sorprendentemente amplio y completo. Por una parte, en el frontend IntegraNova es capaz de generar una interfaz adaptada a la información que le hemos introducido con un *login* y diferentes ventanas donde se pueden ejecutar los servicios que se han creado en el modelo.

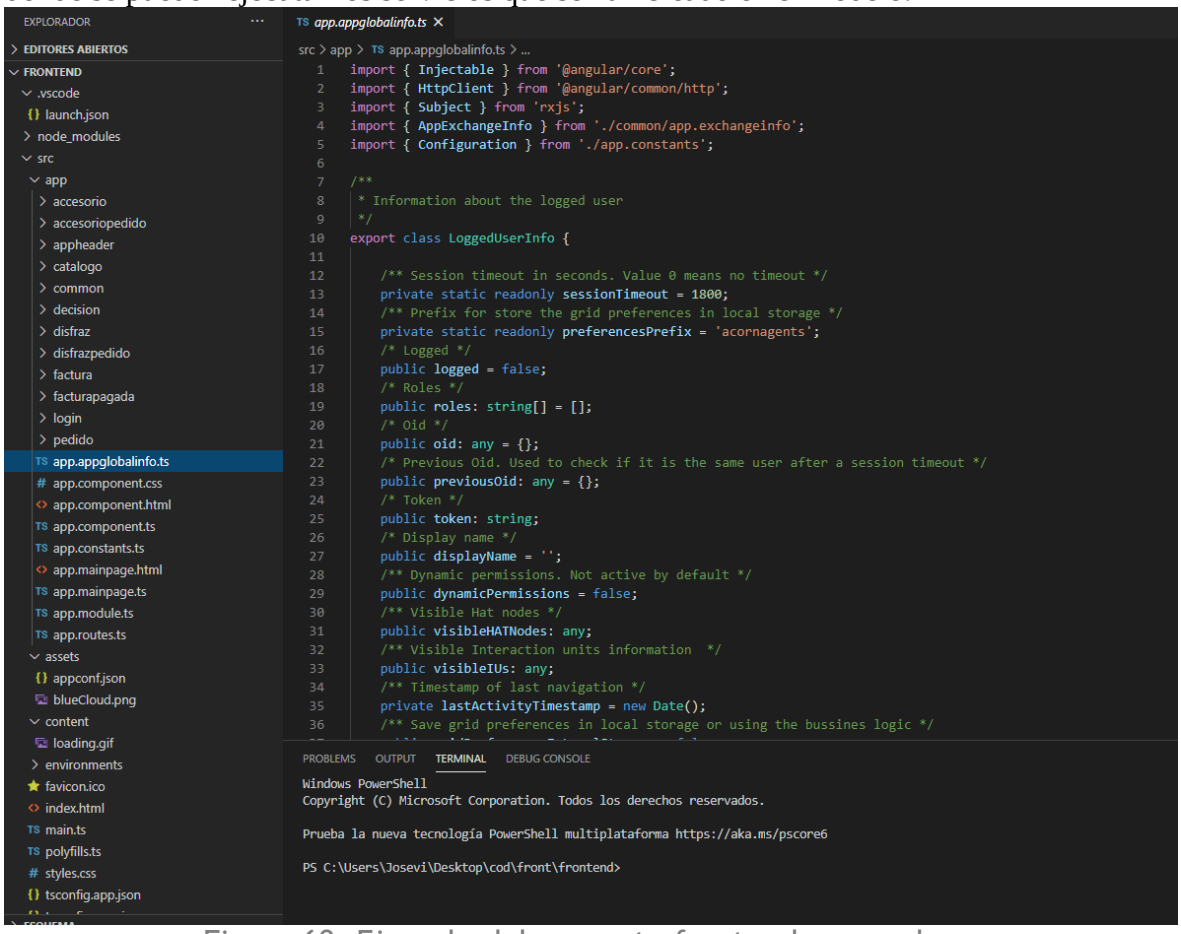


Figura 60. Ejemplo del proyecto frontend generado

Por otra parte, la parte del backend no necesita más modificaciones que las que haya que hacer para eliminar los errores. El proyecto generado contiene de una forma organizada y con un código limpio todos los elementos que necesita esta parte, ya sean modelos, controladores u otros componentes.

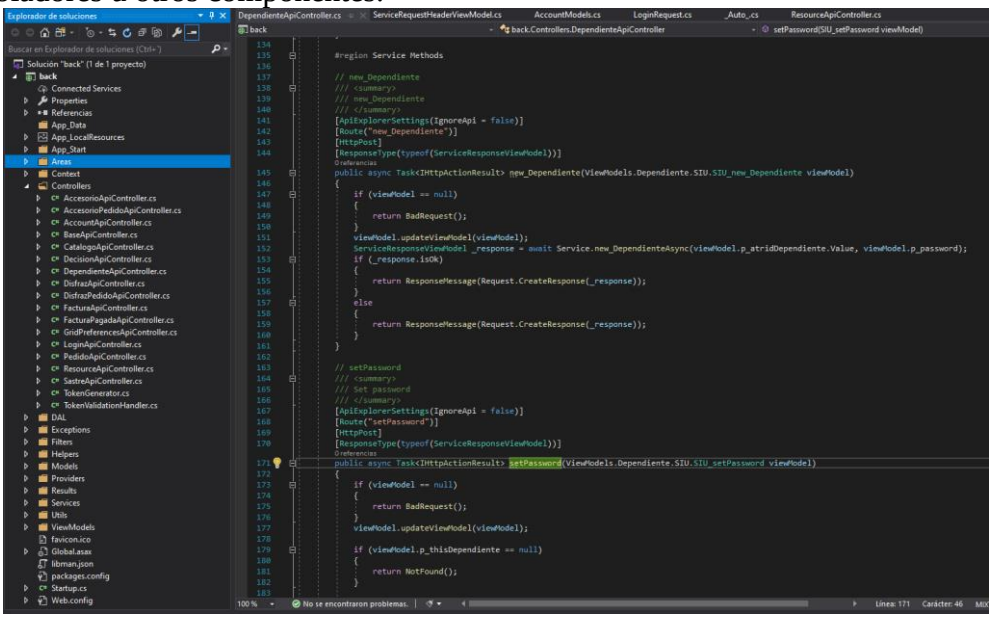


Figura 61. Ejemplo del proyecto backend generado

Para evaluar la amplitud mostrada del código se ha utilizado una herramienta que proporciona Visual Studio para mostrar el diagrama de clases de un proyecto. A continuación, se muestra el tamaño del proyecto generado para el backend. El proyecto del frontend será de un tamaño similar.

Figura 62. Diagrama de clases del proyecto backend generado

Todas las clases que aparecen en el diagrama están pobladas con código, ya sean peticiones a servicios, controladores, etc. Analizando de una forma más extendida dentro del diagrama de clases podemos observar lo que se comenta previamente.

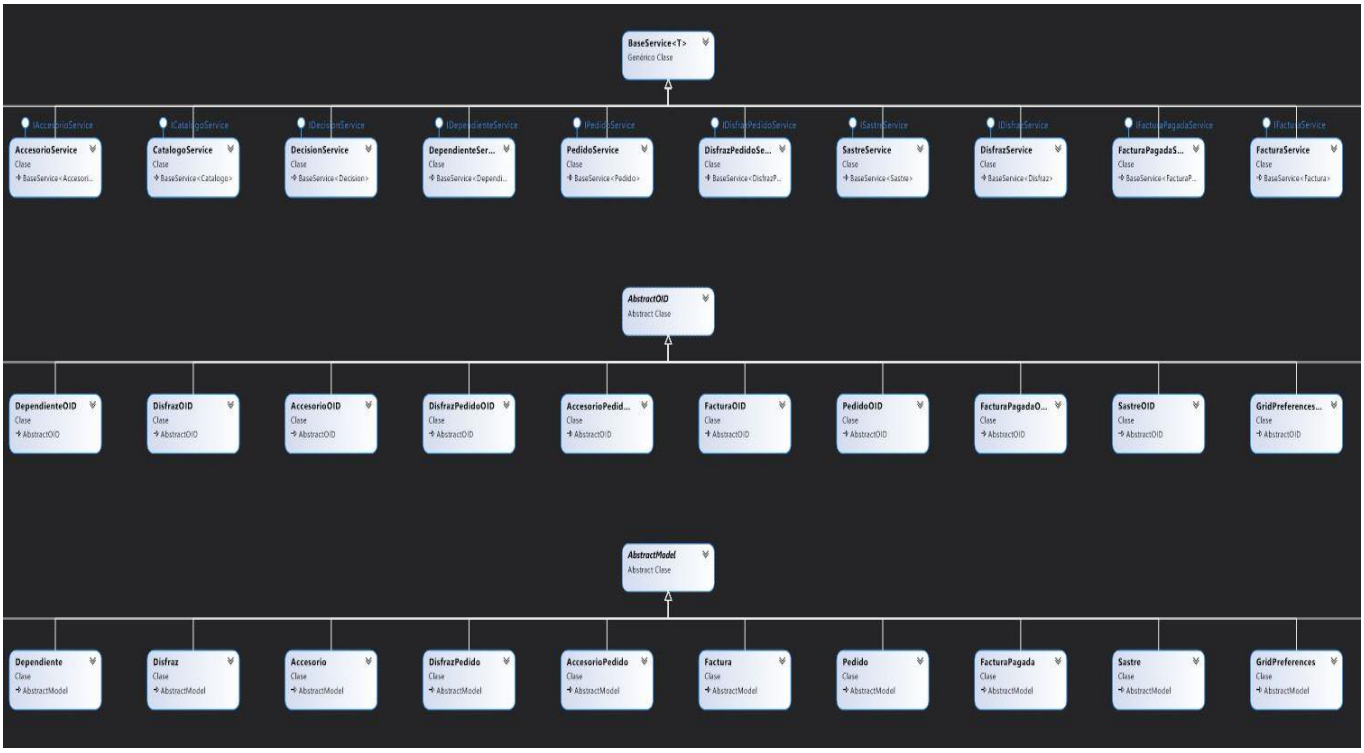


Figura 63. Servicios y modelados en el diagrama de clases del proyecto

En la figura 63 podemos observar como dentro del proyecto hay servicios donde se encuentran las peticiones de todas las clases especificadas en el diagrama de OO-Method, incluso del agente. Asímismo, también observamos que hay modelados de todas las clases indicadas que, aunque en la figura no aparecen todas, en el diagrama sí que están.

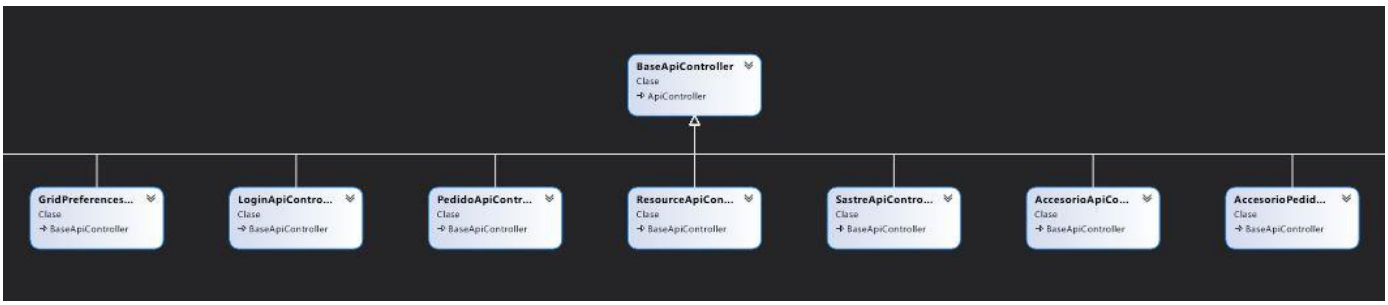


Figura 64. Controladores del diagrama de clases del proyecto backend

De la misma forma, también encontramos controladores de todas las clases diseñadas con las respuestas a las peticiones especificadas en el frontend para la correcta unión de ambas partes. En resumen, podemos concluir que el código generado a partir del diagrama de OO-Method es de una gran calidad, y muy completo. Desde un esquema con apenas 10 clases es capaz de generar un proyecto de gran amplitud y listo para su uso profesional tras pocos retoques.

7. Conclusiones

En esta tesis se ha realizado un estudio de las herramientas Great e IntegraNova, investigando sobre sus correspondientes tecnologías, análisis comunicacional y OO-Method y se ha evaluado la viabilidad de utilizarlas como herramientas de desarrollo software de forma continuada.

Para realizar este estudio el alumno se ha documentado sobre las tecnologías y las herramientas y ha realizado un caso práctico tratando de mostrar todas las opciones que se pueden generar partiendo de un diagrama de análisis comunicacional. El objeto de estudio es un caso de uso de una empresa de disfraces enfocado a mostrar todas las funcionalidades que la herramienta Great contiene. Por consiguiente, el alumno ha aprendido a utilizar tanto las herramientas Great como IntegraNova y ha documentado en la tesis toda esa información que ha obtenido.

Esa información se ha dividido en cuatro partes principales, en estas partes se ha estudiado el proceso realizado y los resultados obtenidos. En primer lugar, se ha evaluado la trazabilidad conceptual, donde se ha evaluado la cobertura funcional de Great con respecto a la capacidad de poder diseñar modelos de análisis comunicacional donde se han obtenido resultados positivos, ya que Great contiene prácticamente todos los elementos necesarios para un correcto diseño de un diagrama de ese tipo. En segundo lugar, respecto a la evaluación de brechas funcionales los resultados no han sido tan positivos, ya que en el diseño interno y la transformación a OO-Method, se han encontrado bastantes evidencias de que Great es una herramienta por madurar que necesita generar más información o dar evidencias de que se ha generado esta. Tercero, respecto a la evaluación del modelo generado se han realizado dos evaluaciones, una evaluación estándar y otra utilizando el *Gold Standard*, gracias a la combinación de ambos se ha observado que el modelo generado es correcto y contiene toda la información necesaria en un modelo de ese tipo para que sea completo. Sin embargo, todo lo que contiene no es suficiente para manejar un sistema de información, ya que los servicios que se generan son servicios de creación, y para manejar un entorno de este tipo se necesitan servicios de todo tipo, para complementar estos servicios, se requiere de un modelado de otros procesos de negocio usando GREAT, particularmente los que permiten gestionar las entidades de datos usadas en el proceso de venta de disfraces y accesorios. Por último, se ha realizado una evaluación del código generado en IntegraNova a partir del diagrama de OO-Method. Respecto a esta parte también hay valoraciones contrapuestas, ya que el código generado es de gran calidad, tanto el *frontend* como el *backend*, en el que en los diferentes lenguajes encontramos proyectos de gran amplitud que podrían utilizarse perfectamente en entornos profesionales. Sin embargo, para ejecutar las partes de *backend* hay muchos impedimentos por incompatibilidades con programas actuales y para la instalación del código generado existe necesidad de realizar tareas técnicas y manuales que requieren conocimiento de la plataforma, lo que se podría mejorar mediante automatización de estos pasos.

Otro punto a destacar es cuál ha sido el resultado de las preguntas de investigación planteadas al inicio de la tesis. Con estas respuestas se observará de una forma externa el cumplimiento o no de las posibles expectativas planteadas al principio del proyecto.

¿Es posible automatizar el proceso de programación para poder generar código a partir de un diagrama de eventos comunicativos?
Como se ha desarrollado en esta tesis es posible, se ha realizado un caso práctico documentado y se ha conseguido generar código correctamente.

¿Un usuario sin conocimientos de programación, pero con algunos conocimientos de análisis comunicacional sería capaz de generar código?
En el momento en el que se escribe la tesis para poder obtener código y poder ejecutarlo necesitas tener conocimientos básicos de desarrollo dirigido por modelos y de programación, ya que para ejecutar el código necesitas realizar pequeños retoques sobre él.

¿La documentación que aporta esta tesis es suficiente para que alguien sin conocimientos sea capaz de realizar el proceso?
Sí, la documentación es apropiada. Ha sido revisada por investigadores del PROS y ellos han dado el visto bueno a la tesis y a la documentación aportada. Uno de los objetivos de la tesis era aportar documentación del proceso realizado y se ha intentado que esta documentación sea exhaustiva para que una persona con los mínimos conocimientos de la materia sea capaz de obtener un resultado similar.

¿Esta tesis proporciona conocimientos nuevos a las materias de análisis comunicacional y OO-Method?
Sí, porque, aunque haya tesis con trasformaciones separadas de estos métodos. Usarlos ambos como una pieza única para generar código es nuevo, por tanto, esta tesis aporta una nueva visión en el tratamiento de estas tecnologías.

También se plantean preguntas más enfocadas a los resultados obtenidos. Como las siguientes:

¿Cuáles son los beneficios de utilizar este proceso?
El principal beneficio de utilizar este proceso en vez de programar directamente en el código es la sencillez con la que IntegraNova crea toda la estructura necesaria para que un proyecto funcione en diferentes lenguajes. Esta creación tradicionalmente es un proceso tedioso e importante para que los futuros cambios no den problemas, y usualmente para usuarios junior puede complicarse.

¿Qué brechas funcionales hay actualmente realizando este proceso?
Actualmente hay varios problemas técnicos que deben ser mejorados para que el proceso sea viable:

1. Incompatibilidad de Great con Windows 10 y las últimas versiones de Java
2. Proceso difícil y poco documentado para poder ejecutar el código generado en IntegraNova

¿Vale la pena dejar usar este proceso a la implementación de código tradicional?
Creo que la generación de código en la estructura del proyecto es muy buena, pero no es viable utilizar únicamente esta opción como forma de desarrollo. Con algunos cambios podría plantearse un uso híbrido, donde un programador después añadiera detalles que no se pueden implementar utilizando este proceso.

Respecto al cumplimiento de los objetivos de la tesis los resultados son muy positivos, ya que los objetivos eran realizar un caso práctico completo utilizando las herramientas y documentándolo, este objetivo tenía como propósito ayudar a la comunidad de modelamiento a comprender como funciona Great, herramienta de la que apenas hay información de cómo se usa, además de como enlazar la información obtenida con IntegraNova y como generar código a partir del diagrama diseñado de OO-Method.

El otro objetivo principal de la tesis era mostrar las oportunidades y los problemas que nos ofrece realizar este proceso como si fuera una técnica de programación real y si sería capaz de sustituir a los métodos de programación tradicionales. En este apartado mis valoraciones son realmente positivas, ya que aunque sea la primera vez que he realizado el proceso, el código que ha sido capaz de generar a partir de la información que le he introducido es superlativo, y aunque creo que tiene que mejorar en algunas cosas, con una correcta documentación, como la de esta tesis y un proceso más pulido, sería una opción realmente a tener en cuenta, principalmente por el tiempo que ahorra en crear la infraestructura que normalmente un sistema de información te requiere.

Como conclusión estoy muy contento de haber podido participar en este gran proyecto que pretende cambiar completamente la forma en la que se crean programas, también de que mi primera tesis trate de un tema tan interesante como este. Me ha apasionado todo lo que he investigado y la forma de trabajar junto a mis tutores. Han sido tiempos difíciles, pero creo que siempre hay que anteponerse a los obstáculos que aparezcan, como dijo el filósofo Eckhart Tolle “La vida te dará la experiencia que sea más útil para la evolución de tu conciencia”, así que confío en que todo esto me haya servido de evolución para poder ser un gran profesional y una gran persona.

7.1. Relación con los estudios cursados

Para el correcto desarrollo de este proyecto he utilizado información que he ido aprendiendo a lo largo de la carrera de diferentes asignaturas. El ejemplo más claro de asignatura de la que he usado información estudiada es desarrollo dirigido por modelos, es decir, la asignatura de la carrera que abarca el proyecto en el que me ha introducido el grupo PROS. En esta asignatura he aprendido a diseñar modelos similares a los que he creado en esta tesis, a crear diagramas de casos de uso y a especificarlos y muchos más aspectos en los que me he involucrado que me han servido. En las prácticas de esa asignatura se utilizan herramientas de modelado basadas en Eclipse, o el propio *Eclipse Modelling tools*, conocimientos que me han servido para asimilar más fácilmente como utilizar el entorno de Great.

Otro aspecto importante de esta tesis relacionado con los estudios cursados ha sido el conocimiento de lenguajes de programación aprendido en asignaturas como Introducción a la programación, programación, ingeniería del software, diseño software… Todas estas asignaturas más algunas de la rama de ingeniería del software me han servido para tener conocimientos de cómo funciona un sistema de información y un proyecto software, útil para analizar el código obtenido y evaluar la información obtenida en ese código con respecto a la que me ha aportado.

Por último, destacar algunas asignaturas que me han ayudado con algunos conocimientos que no están directamente enlazados con el contenido de la tesis, pero sin ellos sería más complicado llevarla adelante. Algunos ejemplos son gestión de proyectos y proceso del software por darme una idea de cómo debía enfocar todo el proceso, las asignaturas de inglés, ya que sin ellas no podría documentarme de la misma forma. Y muchas otras que me han ayudado en diferentes facetas como la preparación y escritura de la tesis o la preparación del entorno y base de datos…

8. Trabajos futuros

Tras finalizar este trabajo de fin de grado y observar los problemas que tienen estas herramientas se abren diferentes caminos a posibles trabajos que podría realizar para continuar la investigación en el campo. Algunos ejemplos son los siguientes:

- Desarrollo de una herramienta para pasar de texto a análisis comunicacional
- Integración de las herramientas con extensiones
- Documentación y análisis en extensión del código producido y ejecución de este
- Uso de otras herramientas e integración con IntegraNova
- Añadir nuevas funcionalidades a Great en colaboración con su equipo de trabajo

9.Bibliografía

ATL. (2020). *eclipse.org*. Obtenido de eclipse.org: https://www.eclipse.org/atl/

Catalunya, U. O. (2020). *uoc.edu*. Obtenido de uoc.edu: http://design-toolkit.recursos.uoc.edu/es/espacio-del-problema-y-espacio-de-la-solucion/

España, S. (2009). *Communication Analysis: a Requirements Engineering Method for Information Systems.*

España, S. (2011). *Methodological integration of Communication Analysis into a model-driven software development framework.*

Gómez, D. S. (2010). *Aplicación práctica de MDA con Olivanova para una aplicación de evaluación de recursos humanos.*

informativos.net. (2020). *informativos.net*. Obtenido de informativos.net: https://www.informativos.net/tecnologia/integranova-ha-presentado-la-maquina-de-programar_46556.aspx

IntegraNova. (2020). *http://www.integranova.com/es/*. Obtenido de http://www.integranova.com/es/: http://www.integranova.com/es/

MDETOOLS. (2 de Febrero de 2020). *www.mdetools.com*. Obtenido de www.mdetools.com: http://www.mdetools.com/tool.php?catid=5

MetaCase. (2020). *www.metacase.com*. Obtenido de www.metacase.com: https://www.metacase.com/solution/

Moreno, A. (2020). Obtenido de Time of software: http://timeofsoftware.com/proceso-mda/

OMG. (2020). *https://www.omg.org/mda/*. Obtenido de https://www.omg.org/mda/: https://www.omg.org/mda/

Pastor, O. (2001). The OO-method approach for information systems modeling: from object-oriented conceptual modeling to automated programming. *Sciencedirect*.

PROS, G. (2020). *http://www.pros.webs.upv.es/*. Obtenido de http://www.pros.webs.upv.es/: http://www.pros.webs.upv.es/

Sergio España, M. R. (2010). *Integration of Communication analysis and the OO-Method: The Superstationery.*

Softeam, M. (2020). *www.modeliosoft.com*. Obtenido de www.modeliosoft.com: https://www.modeliosoft.com/en/technologies/mda.html

Team, M. (2020). *nomagic.com*. Obtenido de nomagic.com: https://www.nomagic.com/products/magicdraw

UML. (2020). *https://www.uml.org/*. Obtenido de https://www.uml.org/: https://www.omg.org/spec/UML/

Vicente-Chicote, C. (2012). *Desarrollo de Software Dirigido por Modelos.* Ra-Ma.

Wikipedia. (2020). *en.wikipedia.org*. Obtenido de en.wikipedia.org: https://en.wikipedia.org/wiki/Method_(computer_programming)

Wikipedia. (2020). *es.wikipedia.org*. Obtenido de es.wikipedia.org: https://es.wikipedia.org/wiki/Sistema_de_informaci%C3%B3n

Pastor, O., & Molina, J. C. (2007). Model-driven architecture in practice: a software production environment based on conceptual modeling. Springer Science & Business Media.

Lopez, O. P., Hayes, F., & Bear, S. (1992, May). Oasis: An object-oriented specification language. In International Conference on Advanced Information Systems Engineering (pp. 348-363). Springer, Berlin, Heidelberg.

Wieringa, R. J. (2014). Design science methodology for information systems and software engineering. Springer.

González, A., España, S., Ruiz, M., & Pastor, Ó. (2011). Systematic derivation of class diagrams from communication-oriented business process models. In Enterprise, Business-Process and Information Systems Modeling (pp. 246-260). Springer, Berlin, Heidelberg.

Ruiz, M. (2018). TraceME: A Traceability-Based Method for Conceptual Model Evolution. Springer International Publishing.

10. Anexo de figuras