UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DSIC DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Departamento de Sistemas Informáticos y Computación
Universitat Politècnica de València

Extensión y mejora de un método de especificación de valor para la derivación y priorización de procesos de negocio

Trabajo Fin de Máster

Máster Universitario en Ingeniería y Tecnología de Sistemas Software

Autor: Cano Genovés, Carlos

Tutor: Insfrán Pelozo, César Emilio

Cotutor: Abrahão Gonzales, Silvia Mara

2016 - 2017

Agradecimientos:

A mis tutores Emilio y Silvia por todo el apoyo, consejos y ayuda prestada en este trabajo.

A Raphael por tener tanta paciencia conmigo y enseñarme a diseñar, ejecutar y analizar experimentos.

Y por último, pero no por ello menos importante a mi familia y compañeros de clase por todo el apoyo mostrado.

Resumen

Hoy en día gran cantidad de aplicaciones se desarrollan de forma incremental, dando lugar a la necesidad de priorizar los elementos que se incluirán en cada incremento. Para tal propósito, en el grupo de investigación ISSI de la UPV se ha propuesto un ***método para la especificación de valor en procesos de negocio*** el cual no sólo ayuda a priorizar los elementos de cada incremento con respecto a su *valor*, sino que también puede ser utilizado para la generación de modelos de procesos de negocio.

Este TFM tiene dos objetivos: por un lado, se pretende realizar una **extensión y mejora del método** con respecto a la priorización de procesos de negocio basado en el valor y definir los mapeos para la transformación entre el modelo de valor y los procesos de negocio. Por otro lado, **realizar la validación empírica del lenguaje de modelado desarrollado** Value@GRL en comparación con el lenguaje de modelado de objetivos de i*. Este trabajo se desarrollará en el contexto del proyecto de Value@Cloud.

Palabras clave: Ingeniería del software, desarrollo dirigido por modelos, procesos de negocio, valor, priorización

Abstract

Nowadays, a great number of applications are developed incrementally, giving rise to the need to prioritize the elements that will be included in each increment. For this purpose, a **method for the specification of value in business processes** has been proposed by the ISSI research group at the UPV. This method not only help to prioritize the elements of each increment with respect to its value, but can also be used for the generation of business process models.

This Master Final Work (TFM) has two objectives: on the one hand, it intends to carry out an **extension and improvement of the method** with respect to the prioritization of business processes based on the value and to define the mappings for the transformation between the value model and the business processes. On the other hand, **to perform the empirical validation of the modeling language developed** Value@GRL in comparison to the objective modeling language of i*. This work will be developed in the context of the Value@Cloud project.

Keywords: Software engineering, model driven development, business processes, value, priorization

Tabla de contenidos

Índice de figuras

Índice de tablas

1. Introducción

1.1. Contexto de la investigación

Esta tesis de máster se ha desarrollado en colaboración con el grupo de investigación de Ingeniería del Software y Sistemas de Información (ISSI) del Departamento de Sistemas Informáticos y Computación de la Universitat Politècnica de València (UPV).

El trabajo de esta tesis se engloba en el proyecto de I+D financiado con fondo público Value@Cloud. En este proyecto, colaboran no solo la UPV, sino también otras empresas interesadas en el proyecto, así como otras universidades externas.

1.2. Contexto

Hoy en día con el auge de la informática, hay una gran cantidad de aplicaciones que se están desarrollando. Una de las formas de desarrollar aplicaciones que últimamente está de moda es la de desarrollo ágil basado en incrementos, básicamente es que la aplicación se desarrolla por versiones (partes) donde cada versión aporta nuevas funcionalidades, o cambia algo hecho previamente.

Esta forma de desarrollo ágil por incrementos permite que los clientes obtengan de forma rápida una aplicación con funcionalidades mínimas para poder trabajar y que poco a poco obtengan nuevas funciones. Uno de los grandes problemas que tiene este tipo de desarrollo es el de decidir qué elementos, o funcionalidades implementar en cada versión. Con tal de solventar el problema de decidir cuales se deben implementar, se está desarrollando el **método para la especificación de valor en procesos de negocio y la derivación incremental**, procedente de un trabajo de fin de máster anterior.

1.3. Objetivo

Este trabajo de fin de máster tiene dos objetivos principales relacionados con el método para la especificación de valor en procesos de negocio y la derivación incremental.

Por un lado, se pretende **mejorar** la actividad de definición de modelado de valor del método, donde a cada uno de los elementos del modelo de objetivos se les asigna una importancia y luego se calcula su valor en base a los elementos con los que están relacionados. Así como también se pretende **extender** la actividad de definición de procesos de negocio de alto nivel, donde el modelo de objetivos se transforma en un modelo de procesos de negocio para poder añadir un orden y las actividades faltantes no representadas en el modelo de objetivos.

Por otro lado, se pretende realizar una **evaluación empírica** del lenguaje de modelado de valor, Value@GRL, el cual se ha desarrollado explícitamente para este método. Este lenguaje de modelado está basado en GRL, al cual se le ha añadido el atributo de valor. Para la realización de esta validación se compara Value@GRL (sin la parte de valor) con otro lenguaje de modelado de objetivos i* (i-star).

1.4. Método de investigación

Las actividades de investigación que han dado lugar a este trabajo de fin de máster están estructuradas siguiendo una extensión del modelo para la transferencia de tecnología propuesto por Gorschek et al. [1] basado en las necesidades de la industria incluyendo actividades de evaluación y de observación. Este modelo de investigación y transferencia de tecnología se basa en ocho actividades relacionadas donde la búsqueda de soluciones adecuadas se realiza de forma iterativa por medio de la formulación de soluciones candidatas y la correspondiente validación empírica que permite dirigir los esfuerzos hacia una solución realista (Ver **Figura 1**). Las actividades son las siguientes:

1. **Identificación del problema**: Se pretende entender el problema que el socio industrial pretende resolver. Este paso es llevado a cabo a través de reuniones y workshops donde los expertos de la compañía hacen presentaciones acerca de los retos que ellos perciben.
2. **Formulación del problema**: Una vez que el problema está identificado, éste es formulado de manera más precisa y los factores contextuales y las asunciones del trabajo, son especificadas claramente.
3. **Revisión del estado del arte**: Se realiza una revisión crítica de la literatura, así como de las soluciones tecnológicas comerciales y *open source* disponibles para identificar hasta qué punto las metas han sido abordadas y cuáles son los problemas abiertos a resolver con la investigación a desarrollar.

4. **Solución candidata**: Se idean una o más soluciones potenciales. Dichas soluciones serán más tarde depuradas a través de las distintas etapas de refinamiento (actividades 6 y 7).
5. **Entrenamiento**: Se trata de una etapa incremental. En las primeras fases, el entrenamiento se enfoca a crear el conocimiento necesario para que los profesionales del área puedan formarse una opinión de la aplicabilidad de la propuesta para determinar si los constructos son naturales y sencillos de construir. En las fases tardías, el entrenamiento sirve para crear guías y pasos metodológicos detallados para aplicar las soluciones.
6. **Validación inicial**: Se lleva a cabo una evaluación preliminar de las soluciones, en un entorno de laboratorio o en una configuración industrial limitada. En el caso de entornos de laboratorio se llevarán a cabo estudios controlados (por ejemplo, experimentos controlados y estudios de caso con alumnos o profesionales).
7. **Validación realista**: Se llevan a cabo estudios de caso en configuraciones industriales, empezando por estudios piloto para después extenderse en usos más amplios. En esta fase se definirán guías prácticas y se desarrollarán las herramientas que soporten la propuesta.
8. **Lanzamiento de la solución**: En este último paso se valoran los resultados obtenidos. Las herramientas y el material de entrenamiento son preparados para su uso en contextos industriales.

Durante el transcurso de este trabajo final de máster, ha sido posible realizar hasta la actividad de validación inicial. Dentro del marco del proyecto Value@Cloud se pretende plantear a las empresas participantes del mismo la posibilidad de realizar con ellas las actividades de validación realista y liberación de la solución.

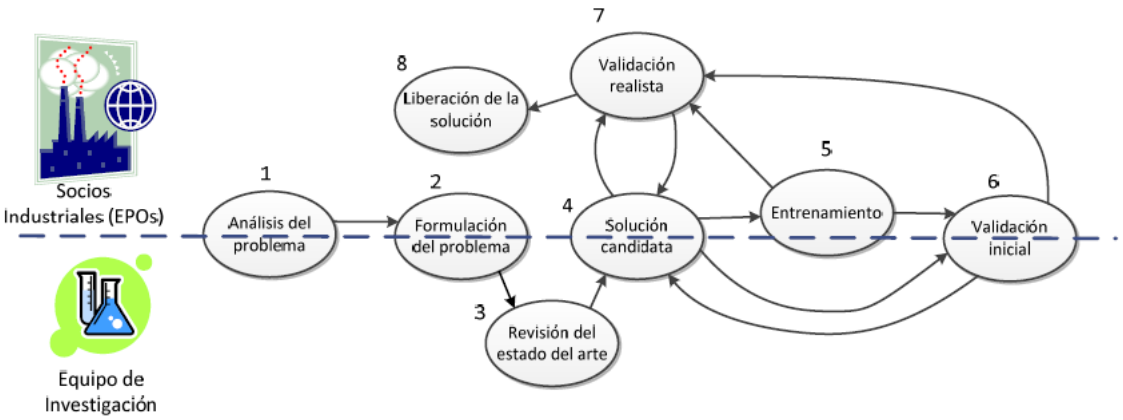


Figura 1: *Modelo de transferencia tecnológica.*

1.4.1. Experimentos

La experimentación es una fase crucial de la validación y puede ayudar a determinar si los métodos utilizados se ajustan a una teoría particular. Los experimentos controlados son apropiados para investigar diferentes aspectos, como la confirmación o prueba de teorías existentes, la evaluación de la precisión de modelos, validación de medidas, etc. Los experimentos que aparecen en esta tesis han sido diseñados siguiendo el framework para la experimentación en Ingeniería del Software [2].

La **Figura 2** muestra las actividades del proceso experimental sugerido por Wohlin et al. [2] y los artefactos generados en cada una de estas actividades.

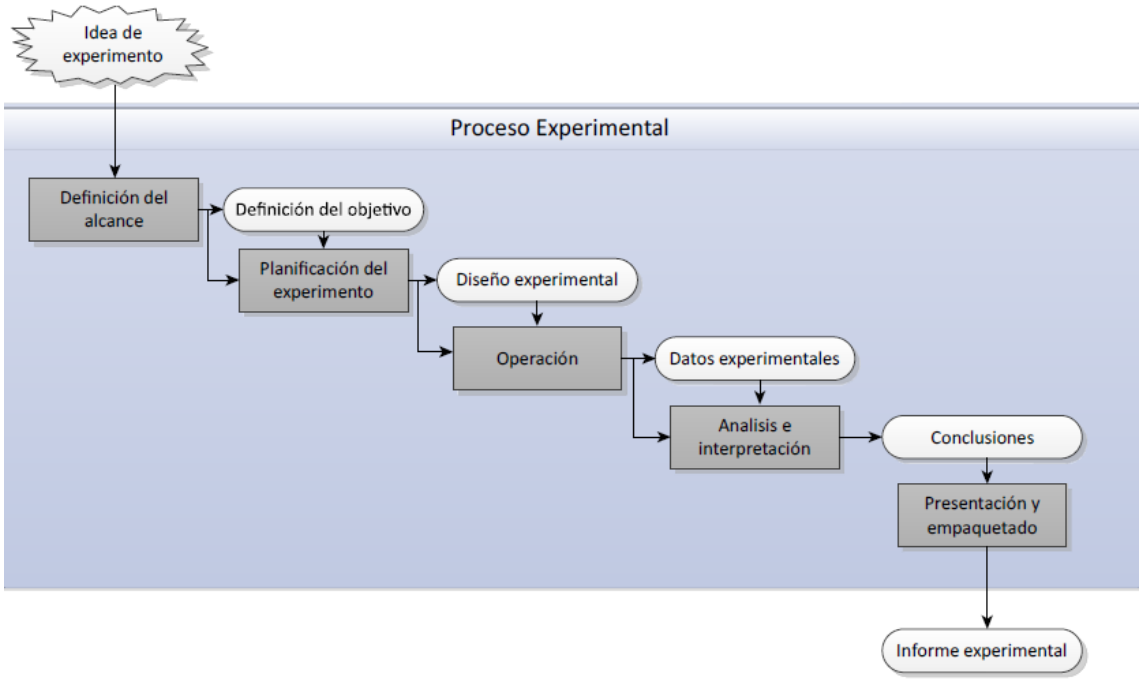


Figura 2: *Proceso experimental con los artefactos generados de las distintas actividades.*

La primera actividad es la **definición del alcance**, aunque todavía no estén definidas formalmente, sí deberán estar claras las hipótesis del experimento, así como el objetivo y las metas. La meta del experimento es formulada a partir del problema a resolver siguiendo el framework Goal Question Metric (GQM) [3] con el esquema:

Analizar	<Objeto de estudio: ***¿Qué se analiza?***>
Con el propósito	<Propósito: ***¿Cuál es la intención?***>
Con respecto a su	<Enfoque de calidad: ***¿Cuál es el efecto estudiado?***>
Desde el punto de vista	<Perspectiva: ***¿Qué vista?***>
En el contexto de	<Contexto: ***¿Dónde tiene lugar el estudio, sobre qué artefactos y con qué tipo de sujetos?***>

La siguiente actividad, es la **planificación del experimento**, donde se decide el diseño. En esta actividad se define en profundidad el contexto, que incluye tanto la componente personal (perfiles tendrán los sujetos) como el entorno en el que tendrá lugar. También se especifican formalmente las hipótesis experimentales, incluyendo las hipótesis nulas e hipótesis alternativas, así como las variables, tanto las independientes (entradas) como las variables dependientes (salidas). Además, se seleccionará un diseño experimental y se identificará y preparará la instrumentación a utilizar. El diseño experimental describe, por ejemplo, cómo se llevarán a cabo los ensayos (online vs offline) o la aleatorización de los sujetos. Por último, en la fase de planificación, es donde se ha de considerar la cuestión de la validez de los resultados que cabe esperar.

Durante la **operación** del experimento es donde se van a preparar, ejecutar y validar los datos recogidos. Durante la ejecución debe asegurarse que se lleva a cabo siguiendo el diseño definido en la fase de planificación y que se recogen los datos experimentales. Por último, se validan los datos recogidos para asegurar que son correctos y proveen una imagen real del experimento.

En el **análisis e interpretación** se emplean los datos recogidos y validados en la fase de operación. Se emplean estadísticos descriptivos con el fin de entender los datos de manera informal. El siguiente paso es analizar si el conjunto de datos a considerar debe ser reducido, bien eliminando puntos o bien eliminando variables, tras analizar si hay variables redundantes que nos ofrecen la misma información. Una vez reducido el conjunto de datos, se llevarán a cabo las pruebas de las hipótesis. Se seleccionarán las pruebas en función del tipo de escala, variables de entrada y tipo de resultados que se buscan. La interpretación se centra en determinar si, en base a las pruebas, se pueden aceptar o rechazar las distintas hipótesis, esto es, determinar la influencia de las variables independientes sobre las variables dependientes en el caso en que se rechacen las hipótesis nulas.

Por último, la actividad de **presentación y empaquetado** está relacionada con la preparación de la documentación, ya sea un artículo de investigación para difundir los resultados o un paquete de laboratorio con el fin de llevar a cabo replicaciones del experimento.

En este trabajo de fin de máster, se ha realizado hasta el apartado de análisis e interpretación, pero dentro del marco de este proyecto se pretende continuar con el mismo y finalizar todos los apartados

1.5. Estructura del documento

La estructura de este documento es la siguiente:

- El primer capítulo es la introducción, donde se plantea el objetivo de esta memoria, así como el contexto.
- En el segundo capítulo se expone el marco tecnológico, donde se habla de las distintas tecnologías empleadas a lo largo de este trabajo, así como también sobre conceptos clave relacionados con el mismo.
- En el tercer capítulo, se presentan los trabajos relacionados, donde se habla del valor, priorización, el valor en el modelo de procesos y las herramientas.
- En el cuarto capítulo, se hace una presentación del método que se ha mejorado y extendido en esta tesis de máster, de una forma conceptual.
- En los capítulos quinto, sexto, séptimo y octavo corresponden con cada una de las actividades del método, las cuales se explican detenidamente con ejemplos de cómo funciona el método.
- En el noveno capítulo se observa un ejemplo ilustrado del método, donde se presenta un caso de estudio al cual se le aplica el método.
- En el décimo capítulo, se muestra la evaluación empírica del lenguaje de modelado de valor desarrollado.
- En el undécimo capítulo se presentan las conclusiones de este trabajo, así como también el trabajo futuro y las publicaciones realizadas durante el transcurso de este trabajo.

2. Marco Tecnológico

2.1. El concepto de valor

El término "valor" es definido de forma vaga en la literatura debido a que muchas veces depende de la interpretación subjetiva. Existen muchas referencias en cuanto al valor de una organización, de un cliente y cómo realizar soluciones que contribuyan a aumentar el valor. Las secciones correspondientes al valor y priorización del marco tecnológico de este trabajo han sido extraídas de [4] y adaptadas para este trabajo, ya que este trabajo consiste en una mejora y extensión del mismo y por lo tanto se realiza dentro del mismo marco tecnológico.

El término valor, se encuentra ampliamente utilizado en muchas publicaciones, tanto del ámbito del negocio, como el de la ingeniería del software y la gestión de proyectos. Es un término que no cuenta con una definición común y que a veces se da como un concepto sobreentendido. En [5] se recopilan algunas de las definiciones aparecidas en la literatura:

- Barnett [5]: "… el valor de negocio, como medida de los ingresos del negocio, el precio de las acciones, cuota de mercado u otras métricas de negocio. El Valor está en los ojos del cliente…"
- Patton [6]: "Valor de negocio es algo que entrega beneficios a la organización que paga por el software en la forma de un aumento de los ingresos, evitación de costes o una mejora en el servicio".
- Pettit [7]: "Valor de negocio es un vehículo de comunicación: usamos el valor de negocio para comunicar valor, prioridades, motivación…".
- Rawsthorne [8]: "Valor de negocio es por lo que la dirección está dispuesta a pagar; el valor solo puede ser definido por el cliente final. Y solo es significativo cuando se expresa en términos de un producto específico (un bien o un servicio y a menudo ambos a la vez), que cumple las necesidades de un cliente a un precio concreto en un momento concreto".
- Poole [9]: "Podría no ser posible definir el valor de negocio de TI independientemente de otras actividades. ¿Qué es valor de negocio?:

  Valor de negocio = F(x) + F(y) + F(z) + ….

  Esto es, una función compleja donde debemos equilibrar múltiples cosas… ¡Mientras cambian!
- OMG [10]: Valor es un factor de beneficio medible entregado, asociado a un entregable, a un receptor.

A pesar de que en [5] se investiga desde la perspectiva ágil, las definiciones son lo suficientemente generales para ser tenidas en cuenta en cualquier ámbito. En el mismo trabajo, se resalta que estas definiciones son de artículos de profesionales, de hecho, no se pudo encontrar un artículo científico en el que explicara el valor en un contexto ágil ya que los autores piensan que valor es un concepto evidente por sí mismo ya que ha sido intensamente estudiado por las ciencias económicas.

De estas definiciones se puede concluir que el valor de negocio en la práctica tiende a ser cualitativo, para ver como una práctica crea o no valor hace falta asociarla a una ganancia tangible del negocio, lo cual es problemático. Aunque sí se han encontrado casos en los que se trata el valor de manera cuantitativa, en general se trata de un valor cualitativo.

También, el valor de negocio tiende a ser subjetivo, tanto porque depende del punto de vista del *stakeholder* como porque se expresa de manera informal como "Valor es algo que me hace sentir bien". Esto es problemático cuando la fuente del valor del cliente, su representante ante el equipo de desarrollo, no representa la totalidad de los stakeholders y guía la priorización de requisitos: los requisitos son priorizados en base a ese valor y según la importancia que estos tengan para el proyecto, pudiendo llegar a situaciones en las que una vista parcial del sistema se impone al resto. Este es un aspecto especialmente relevante en el presente trabajo y que se abordará en los siguientes capítulos.

El valor que entrega la solución TI depende de otros procesos fuera de la propia solución (procesos de negocio internos, de otros stakeholders, otras organizaciones…) por lo que es difícil de definir el valor de negocio independientemente del resto de actividades de negocio no informatizadas, otros sistemas existentes o los stakeholders implicados.

Otro aspecto inferido es que el valor de negocio de la solución de TI requiere un grado de confianza en los indicadores de valor que se presentan ya que pueden no ser precisos.

El valor del cliente puede ser visto desde diversas perspectivas, todas ellas complementarias. El valor del cliente, es decir aquello que el cliente valora y que ayuda a cumplir con sus objetivos y que se impone como una restricción o condición a un proyecto o el valor utilizado en la organización para priorizar entre distintos proyectos puede observarse en términos del negocio o puede estudiarse de manera que dirija los desarrollos de software (a fin de entregar productos con el máximo valor al cliente). A continuación, se profundizará en el concepto desde el punto de vista del negocio, de la ingeniería de software y especialmente del desarrollo de software ágil.

2.1.1. Perspectiva de negocio

Jim Highsmith en [11] identificaba el concepto de valor como un término informal que incluye todas las formas de valor que determinan la salud y bienestar de una organización a largo plazo. El valor de negocio expande el concepto de valor de una empresa más allá de lo económico para incluir otras formas de valor como el valor del empleado, el valor del cliente, el valor del proveedor, el del socio, del aliado, el administrativo y el social; muchos de ellos no medidos directamente en términos económicos.

El valor es importante para centrarse en qué es prioritario para la organización y realizar decisiones de priorización entre las oportunidades de negocio, proyectos y productos posibles. El valor tiene componentes tangibles (financieros) e intangibles ambos críticos para el éxito a largo término.

Una variación en el valor del negocio puede ser originada por una mejora de los procesos empresariales y podría ser medida por factores llamados *Value Drivers* como serían: el aumento de beneficios empresariales, mejora de la rentabilidad empresarial, cuota de mercado, aumento de la liquidez financiera, repuesta de las campañas de marketing, mejora de la fidelidad de los clientes, posición de liderazgo, portafolio de productos, cartera de clientes, mejora de la satisfacción de los empleados, aumento de la capacidad de innovación empresarial, etc.

El valor de negocio es utilizado clásicamente por las empresas para tomar decisiones estratégicas usando por ejemplo técnicas como el *Return Of Investment* (ROI) para determinar qué proyectos son más provechosos para una organización.

Una de las metodologías más utilizadas para medir y gestionar el valor (*business value*) es el *Balanced Scorecard Methodology* (BSC (*Balanced ScoreCard*) / CMI (Cuadro de Mando Integral)) [12]. Aunque el concepto también aparece en *Value Chain* (Cadena de valor) definido por Michael Porter [13] y *Management by Objectives* definido por Peter F.Drucker [14].

La *Balanced Scorecard Methodology* es un modelo de planificación estratégica que basa el desarrollo y la medida de la gestión empresarial en indicadores financieros. Esta metodología trata de alinear la estrategia del cliente con las iniciativas, operaciones (procesos) y presupuestos que implementan esta estrategia, proporcionando medidas estratégicas que permitan evaluar el éxito de la estrategia.

En CMI/BSC se descompone el valor del negocio en cuatro componentes: las proposiciones de valor del cliente, los ingresos, recursos y competencias y procesos.

Una proposición de valor del cliente (*Customer Value Proposition*, CVP) es una oferta que ayuda a los clientes a hacer un trabajo importante de forma más efectiva, conveniente o asequible que las alternativas. La estrategia (de negocio) está basada en una propuesta diferenciada de valor del cliente, la satisfacción del cliente es la fuente de una creación de valor sostenible.

La cadena de valor es una técnica de análisis de negocio [13] basada en la vista de la organización como un sistema, hecho de subsistemas en el que cada uno presenta unos puntos de entrada, realiza unos procesos de transformación y sus resultados aparecen como salidas. Es decir, la cadena de valor representa el flujo de valor dentro de los componentes organizativos de la empresa.

Esta técnica es utilizada para la planificación estratégica, dado que la captura de los elementos que intervienen en la cadena permiten razonar sobre las relaciones entre estos elementos y analizarlas en busca de mejoras; por ejemplo localizando dos elementos adyacentes de la cadena en lugares geográficamente cercanos o ideando nuevos modelos de negocio para eliminar intermediarios.

A pesar de la extensión del estudio del valor en el apartado de negocio, en muchas ocasiones el valor se confunde con las herramientas, algunas de las más populares han sido presentadas en esta sección y los indicadores que lo representan y es complicado llegar a la fuente que hace que los indicadores (y por tanto el valor) fluctúe. Esto puede ser un detalle menor cuando se toman decisiones estratégicas, dado que a fin de cuentas una acción que mejore los indicadores de valor y por tanto reporte más beneficios a la organización será positiva independientemente de que se conozca la fuente de valor. Sin embargo, si el objetivo es construir software que sea la fuente de ese valor o un canalizador del mismo, el enfoque de las herramientas para localizarlo y expresarlo será distinto.

2.1.2. Perspectiva de la ingeniería del software

Desde la crisis del software de los años 80, la ingeniería del software se ha preocupado por la investigación y uso de aquellas técnicas y métodos que permitan a los desarrolladores crear software útil y correcto para el cliente, dicho de otra manera, entregar software que satisfaga los requisitos de funcionalidad esperados por el cliente y aquellos requisitos no funcionales que permitan el uso satisfactorio del software. Debido a ello, se encuentran menciones al valor (del cliente) en el campo de la ingeniería de software en multitud de ocasiones, aunque muchas veces, como es habitual con el concepto en los otros campos, sin recurrir a ninguna definición estructurada y tratando el término cómo sobreentendido. Esto no es así, como por ejemplo demuestran en [15] siendo un tema de debate en la comunidad.

Sin embargo, dentro de la ingeniería del software una línea de pensamiento se ha centrado exclusivamente en la postura de que el software debe aportar valor al cliente y este debería ser el objetivo del proceso ingenieril. Este es el objetivo de la propuesta de Boehm con la creación de la *Value Based Software Engineering* [15] que "*lleva las consideraciones de valor al frente para soportar las decisiones de la Ingeniería del Software a todos los niveles para conseguir o reconciliar los objetivos explícitos de los stakeholders involucrados, desde el personal de mercadotecnia y los analistas de negocio a los desarrolladores, arquitectos y expertos de calidad y desde los expertos de procesos y mediciones hasta a los gestores de proyecto y los ejecutivos*".

En ese mismo texto Boehm aclara qué es lo que entiende por valor: valor relativo, utilidad o importancia y se usa como sinónimo de proposición de valor, función de utilidad y condición ganadora ("*value proposition*," *utility function*," and "*win condition*"). Esta perspectiva se pone en contraposición a la definición del diccionario (Webster) de "valor" como: "*el valor monetario de algo: su precio de mercado*", argumentando que esta definición dejaba fuera del valor, elementos de negocio que, sin un valor financiero, los expertos señalan que sí contribuye al valor de una organización. El autor argumenta que una definición menos estricta permite la inclusión de situaciones menos rigurosamente analizables, pero más amplias como las consideraciones personales, interpersonales o éticas. Jim Highsmith, desde una perspectiva de negocio, coincide con esta postura, como se ha comentado en la sección anterior.

Esta corriente de pensamiento coloca en el foco el valor para el cliente y lo utiliza para guiar la totalidad del proceso de desarrollo desde la captura de requisitos a las fases de pruebas.

En la literatura, el punto del ciclo de vida del software en el cual es más relevante el uso del valor o su discusión es en la fase temprana de la captura de requisitos y también su uso a la hora de priorizar los requisitos. En esta línea se encuentran ejemplos como [16][17] en los cuáles no se hace hincapié en una definición formal pero si se presentan técnicas y herramientas, formales o informales para identificar el valor e introducirlo en la captura de requisitos y en la priorización de los mismos. De forma general, la forma predilecta de capturar el valor es utilizar métodos cuantitativos o cualitativos relativos que permiten comparar los elementos de valor entre sí y discernir los más valiosos que el resto y realizar una clasificación. Este es el caso del uso de *Analytic Hierarchy Process* [16] en la priorización de requisitos por valor [18], que compara entre pares de elementos para otorgarles un valor fijo.

En el aspecto más estructurado y quizás más relacionado con la definición exhaustiva del contexto del sistema y la intencionalidad de los involucrados que lo forman, se encuentran los lenguajes de descripción de la intencionalidad o de objetivos como i* y GRL, donde los elementos de valor son capturados como objetivos de los distintos stakeholders, interconectados entre sí según el efecto

que tengan en otros, las necesidades de los stakeholders de esos elementos o su descomposición jerárquica.

En una dirección ligeramente similar a los movimientos de negocio e³-value y *Value Delivery Modeling Language* (VDML) [10] (también detallados en la sección 2.6 en sus respectivos apartados) tratan de capturar los flujos de valor entre las unidades operacionales, bebiendo directamente de técnicas de negocio como la cadena de valor.

Lo que es evidente es la existencia de la preocupación de capturar los elementos de valor para el cliente y que producirá el sistema de información objetivo de manera que este responda ante estas expectativas y solo así de verdad se estará construyendo software de utilidad para el cliente, independientemente del tipo de representación o técnica utilizada para lograrlo.

2.2. Alineamiento entre IT y negocio

El objetivo final de una solución TIC debe ser la satisfacción de los objetivos de la organización. Esto solamente se alcanza cuando se consigue el alineamiento estratégico entre TIC y los objetivos, actividades y procesos del negocio de manera que se encuentren en armonía [17].

Se ha demostrado que un alineamiento estratégico influencia de manera positiva la efectividad de las soluciones TIC por lo que la alineación de TIC y estrategia de negocio ocupa los puestos de alta prioridad para los directivos y ejecutivos TIC[17]. A pesar de ello y como se ha comentado en este capítulo, este hecho es ignorado en numerosas ocasiones por las técnicas de la ingeniería de requisitos, más centradas en la descripción de la operatividad y que confían en la experiencia de los desarrolladores a la hora de identificar requisitos y diseñar productos adecuados a las necesidades del cliente.

Varios aspectos del análisis de negocio se han incluido como se indica en [17] en la investigación de la ingeniería de requisitos. Algunos aspectos como la estructura organizacional y las relaciones de dependencia entre actores en un sistema, el análisis de valor económico y de negocio, el modelado de proceso de negocios dirigido por los objetivos de la organización y la elicitación de objetivos organizacionales de los cuales derivar requisitos tratan de establecer el puente la perspectiva de negocio y la de IT de manera que sus resultados alineen los objetivos de negocio y el sistema de información objetivo. Un ejemplo de estas técnicas es [17] donde se propone un acercamiento analítico para la verificación y validación de los requisitos en relación con su alineación con la estrategia de negocio utilizando el modelado de objetivos para representar la estrategia de negocio como requisitos.

Queda patente la construcción de sistemas TIC de valor, donde los desarrolladores deben ser conscientes del contexto del negocio, el dominio y los requisitos. Esto significa que deben conocer y comprender el modelo de negocio que debe soportar el sistema, las intenciones y metas de la organización que motivan la creación de valor y de un sistema de información que lo soporte. La historia reciente de los negocios electrónicos demuestra que fallar a la hora de entender el modelo de negocio a menudo resulta en negocios de poca duración o incluso en la quiebra [19].

Es necesario, por tanto, contemplar durante todo el proceso de desarrollo de software el objetivo general del desarrollo: que este sirva para que el negocio cumpla con sus objetivos de valor.

2.3. Desarrollo de software ágil

Bajo el término de Desarrollo de Software Ágil se recogen un conjunto de principios de la ingeniería de software basados en el desarrollo iterativo e incremental donde los requisitos y soluciones evolucionan mediante la colaboración de grupos auto-organizados y multidisciplinarios. Estos principios favorecen la rapidez, la entrega de valor temprana y la flexibilidad como respuesta al cambio. Los procesos de desarrollo ágiles se oponen a los métodos tradicionales siendo la flexibilidad y la autonomía propiedades importantes para los métodos ágiles. De manera que, en los desarrollos ágiles, a diferencia de los tradicionales el conjunto del proyecto no es planeado y planificado de antemano, si no que el proceso de desarrollo se divide en varias iteraciones en las cuales se diseñan e implementan partes del sistema software a desarrollar.

El término ágil es utilizado por primera vez dentro de la comunidad de desarrollo de software en 2001 en el llamado *Agile Manifiesto* [20] firmado por diecisiete personalidades de la industria y la investigación del campo del desarrollo de software. En él se enunciaban los principales valores y principios de esta forma de desarrollo de software, que marcarían las futuras prácticas que se integrarían en esta disciplina.

Valores ágiles expuestos en el *Agile Manifiesto*:

Individuos e interacciones sobre procesos y herramientas
Software funcional sobre documentación exhaustiva
Colaboración con el cliente sobre negociación del contrato
Respuesta al cambio sobre seguir un plan

La tendencia general es la de poner por delante los individuos y sus interacciones a los procesos y las herramientas, de manera que estas últimas no se conviertan en el fin si no en un medio para el proceso de desarrollo de software.

De la misma manera, se favorece el desarrollo de software que funcione sobre extensas documentaciones que no contribuyan a crear software funcional. Un aspecto muy relevante es la colaboración con el cliente al que se le incluye dentro del proyecto y no solamente como el firmante del contrato y el receptor del producto. De la misma manera, dado que la experiencia muestra que los cambios se producirán a lo largo del desarrollo se prefiere responder a él sobre seguir una planificación de forma estricta realizada de antemano.

El manifiesto ágil surge después de que a principios de los años noventa un conjunto de métodos de desarrollo "ligeros" comenzaran a surgir como alternativa a los métodos "pesados" anteriormente aparecidos y cuya principal crítica era que eran demasiado rígidos, altamente regulados, rígidos y micro-gestionados que comprometían los resultados del software, poniendo más esfuerzo en el cumplimiento pormenorizado de las tareas del proceso que en el desarrollo del software propiamente dicho. Estos nuevos métodos como Scrum (1995), Cristal Clear (1996), Extreme Programming (XP) (1996) o el diseño dirigido por características (*Feature Driven Development*) (1997) utilizaban conceptos de la gestión de proyectos y desarrollo de software anteriores como el diseño iterativo, la producción evolutiva de software y el desarrollo de software adaptativo. Estos métodos de desarrollo a pesar de ser anteriores al manifiesto se encuentran clasificados como métodos ágiles actualmente por ajustarse a los principios de estos.

12 principios ágiles por el Manifiesto Ágil [21]:

1. Nuestra más alta prioridad es satisfacer al cliente a través de **entrega de software de valor temprana y continua.**
2. **Bienvenidos los cambios en los requisitos**, incluso tardíamente en el desarrollo. Los procesos ágiles capturan el cambio para conseguir las ventajas competitivas del cliente.
3. Entregar frecuentemente software funcionando.
4. Gente del negocio y desarrolladores deben trabajar juntos diariamente durante el proyecto.
5. **Construir proyectos en torno a individuos motivados**. Darles el entorno y apoyo que necesiten y confiar en ellos para conseguir hacer el trabajo.
6. El método más eficiente y efectivo para transmitir información hacia y dentro de un equipo de desarrollo es la **comunicación cara a cara**.
7. Software funcionando es la medida principal de avance.
8. Los procesos ágiles promueven el **desarrollo sostenible**. Los promotores, desarrolladores y usuarios deberían ser capaces de mantener una paz constante indefinida.

9. La atención continua a la excelencia técnica y el buen diseño mejora la agilidad.
10. **Simplicidad**, el arte de maximizar la cantidad de trabajo NO realizado, es esencial.
11. Las mejores arquitecturas, requisitos y diseños emergen desde **equipos auto-organizados**.
12. Mejora continua.

Según los principios del manifiesto y en las metodologías desarrolladas bajo ellos, se puede generalizar que el principal objetivo es la entrega de software de valor, desde el comienzo del proceso de desarrollo y entregándolo de forma continua.

También es característico de las metodologías ágiles su aceptación de los cambios en los requisitos, a diferencia de otras metodologías (clásicas o de cascada) que se centran en que los requisitos para el producto deben de estar fijados y diseñados desde el inicio del proyecto. Esta propiedad considera que la construcción del software es iterativa y que el descubrimiento de los requisitos debería serlo también.

Una característica fundamental y que se aborda con más detalle a continuación, es la localización de especialistas del negocio junto a los desarrolladores. De forma que desarrolladores y especialistas del negocio, colaboren estrechamente en la solución TIC se alinee con los objetivos de negocio.

En la práctica, como métodos ágiles se han constituido un conjunto de procesos de desarrollo que utilizan prácticas, roles y herramientas que siguen estos principios ágiles y que se engloban bajo la terminología de "procesos de desarrollo ágiles". Algunos de los más conocidos son Scrum, Kanban, *Extreme Programming* (XP), etc. Aunque es habitual que en muchas organizaciones el método de desarrollo no se implemente por completo, sino que se adapta a la organización, seleccionando entre las prácticas ágiles que resulten más apropiadas o útiles en el contexto de esta [22].

En [22] también se analizan las prácticas más extendidas entre los practicantes de los métodos ágiles. En este trabajo se encuentra que XP y Scrum son las prácticas más utilizadas y se procede a analizar el grado de implementación de las mismas. En este trabajo, se indica que, aunque estas metodologías no se implementan completamente, los participantes encuentran satisfactorios sus efectos en el proceso de desarrollo.

Las prácticas más extendidas entre los profesionales indicadas en [22] son:

1. Desarrollo iterativo.
2. Reunión de planificación del sprint.
3. Refactorización.
4. Reunión diaria de Scrum.

5. Reunión de revisión del sprint.
6. *Product Backlog*.
7. Diseño incremental.
8. Línea de código base única.
9. Iteraciones/*Sprints*.
10. Gráficas *Burndown*.
11. Desarrollo dirigido por pruebas.
12. Integración continua.
13. *Backlog* del sprint.
14. Posesión colectiva del código.
15. Cliente localizado con el equipo.
16. Programación por parejas.
17. Pruebas de aceptación.

Entre estas prácticas se incluyen aquellas relativas a la gestión de proyectos, así como otras relativas a la producción del código. El contexto ágil de este Trabajo de Fin de Máster (TFM) viene dado por el proyecto marco bajo el que se engloba, por lo que el trabajo a realizar tiene como requisitos estos principios de agilidad y sigue varias prácticas básicas de la gestión de proyectos.

A continuación, se procede a describir con más detalle las prácticas que se contemplan en este TFM y a las que serán referidas de forma recurrente posteriormente:

- **Desarrollo iterativo**: Aproximación al desarrollo basada en un proceso cíclico de prototipado, *testing*, análisis y refinamiento de un producto software. Tiene como propósito desarrollar un sistema a través de ciclos repetidos de las actividades del proceso de desarrollo. El desarrollo iterativo propone varios ciclos en estas actividades, llegando a darse en ocasiones al mismo tiempo [23][24] y se entrega en cada ciclo no el producto completo sino una parte del mismo.
- **Diseño incremental**: Concepto ligado al desarrollo iterativo, en cada ciclo se desarrolla una porción más pequeña del producto y que suele denominarse “incremento”.
- **Iteraciones/Sprint**: Cada uno de los ciclos de desarrollo iterativos e incrementales en los que se desarrolla el software. Cada ciclo incluye las fases básicas del desarrollo de software: Requisitos, Diseño, Implementación, Verificación (Pruebas) y mantenimiento. Estos ciclos se desarrollan en un período reducido de tiempo, habitualmente de una duración de entre una semana y un mes. El término Sprint es propio de *Scrum*, pero es equivalente al clásico iteración.
- ***Product Backlog***: Lista priorizada de tareas para el equipo de desarrollo que es derivada de la hoja de ruta y los requisitos. Los ítems de trabajo más importantes se colocan al principio. El equipo de desarrollo no trabaja directamente con esta lista, sino que extrae un conjunto de

ítems de trabajo de ella sobre los que trabaja en un sprint/iteración, mientras que los encargados de crear estas tareas, el cliente y el encargado de dirigir la visión del producto, añaden estas al *backlog*, de manera que no entorpezcan el trabajo actual del equipo de desarrollo.

- **Sprint/*Iteration Backlog***: Subconjunto de ítems de trabajo asignados al equipo de desarrollo para realizar en un período de tiempo, la duración de un sprint o iteración [25].
- **Cliente localizado con el equipo**: El cliente es una parte vital del proceso de desarrollo por ello se le incluye en él teniéndolo próximo y siendo sujeto activo del proceso de desarrollo, es el encargado de definir las necesidades del producto y validar los resultados.

2.4. Procesos de negocio

Un proceso de negocio es una colección de actividades estructuradas o tareas que desempeñadas por un rol sirven a un objetivo particular. La gestión de procesos de negocio (*Business Process Management*, BPM) consiste en el entendimiento y la gestión de diversos procesos organizacionales e inter-organizacionales los cuales conectan a personas y sistemas automatizados. La gestión de procesos de negocio integra métodos, técnicas y herramientas para apoyar el diseño, ejecución, gestión y análisis de procesos de negocio operacionales.

El principal objetivo de la gestión de procesos de negocio no es simplemente la coordinación de cadenas de actividades, si no conseguir que esas actividades trabajen juntas para llevar a cabo un objetivo de negocio. En este trabajo, dichos objetivos tratarán de representarse en un modelo de valor para poder analizar qué procesos o tareas están contribuyendo (y en qué medida) a los objetivos de negocio, es decir, están creando valor.

El diseño y la gestión de procesos de negocio no son conceptos novedosos pero las tecnologías de la información han permitido facilitar la gestión de los procesos y la automatización de los mismos.

En este trabajo, se abordarán los procesos de negocio como herramientas de diseño de sistemas de información y como fuente para el refinamiento de requisitos y también como forma de comunicación con el cliente.

En la literatura se encuentran diferentes lenguajes de descripción de procesos de negocio y flujos de trabajo. Se pueden dividir, de manera general, en lenguajes centrados en el modelado de dominio y los lenguajes centrados en la ejecución.

Del primer grupo podemos encontrar *Event-driven Process Chain* (EPC) [26], Business *Process Model Notation* (BPMN) [27], *Use Case Maps* (UCM) [28], y *Yet Another Workflow Language* (YAWL)[29] y del segundo grupo, lenguajes centrados en la ejecución, específicamente *Business Process Execution Language* (BPEL) [30], redes de Petri [31] y acercamientos basados en reglas como ECA (*Event Condition Action*, Evento Condición Acción) [32]. La diferencia principal entre ambos grupos se encuentra en el nivel de detalle: los lenguajes de ejecución permiten especificar aspectos más técnicos con mayor detalle y por ello son capaces de ser ejecutados automáticamente. Aunque la frontera entre ambos grupos es cada vez más difusa ya que ambos tipos de lenguajes cuentan con una capacidad de expresión suficiente para representar cualquier tipo de flujo o cuentan con las extensiones necesarias o las transformaciones para convertirse en lenguajes ejecutables.

Un modelo de proceso de negocio presenta 5 perspectivas de modelado. La *perspectiva funcional* se centra en la ejecución de tareas concretas y típicamente considera conceptos para describir subprocesos y tareas atómicas. La *perspectiva de comportamiento* expresa la secuencia de los elementos del modelo. En la *perspectiva organizacional* se destaca quién o qué procesa las tareas y como se gestionará la distribución de las mismas. La *perspectiva informativa* expresa los elementos de datos relacionados con el proceso ya sean necesarios para el procesamiento de tareas o generados por estas. Por último, la *perspectiva de contexto* refleja una visión del conjunto o una meta perspectiva del proceso y contiene elementos relevantes a las características del proceso como objetivos de negocio o métricas (KPIs).

Los modelados de proceso de negocio se centran en el área funcional, en este TFM se pretende dar a esta expresión del *qué se va hacer*, un contexto de objetivos de negocios y dependencias de usuarios que permita suplir la falta de expresividad de los lenguajes de procesos de negocio en la perspectiva de contexto.

Así pues, para el caso que nos ocupa para la notación de este modelo de procesos de negocio se presentan dos fuertes candidatas: BPMN (estándar y altamente popular y ampliamente utilizado en la especificación de requisitos de desarrollo de software [33] y UCM (también estándar y con una estrecha relación con el modelado de objetivos).

Business Process Model and Notation (BPMN) es un estándar del Object Management Group (OMG), originalmente creado por la *Business Process Initiative* y desde 2005 mantenida por la OMG. Su última versión, 2.0.2, fue lanzada en enero de 2014 y es actualmente la vigente.

BPMN es una notación para la representación gráfica de flujos de procesos de negocios en un Diagrama de Procesos de Negocios (BPD) para su diseño e implementación. Está basado en diagramas de flujo similares a los diagramas de actividad de UML (*Unified Modeling Language*). Su principal objetivo es

soportar la gestión de procesos de negocio tanto por personal técnico como por personal del dominio de negocio. Combinando una semántica intuitiva para los usuarios no técnicos pero lo suficientemente expresiva para representar elementos del proceso de negocio más complejos. También provee una asociación entre elementos gráficos de la notación y una capa subyacente para lenguajes de ejecución como BPEL.

Por su parte, *Use Case Maps* (UCM) forma parte como GRL del estándar de la *International Telecommunications Union* (ITU) [34] denominado URN (*User Requirements Notation*) [28].

La notación de UCM es una aproximación orientada a escenarios que permite describir relaciones causales entre responsabilidades entre uno o más casos de uso. Esta notación permite ilustrar los componentes opcionales envueltos en un escenario en un diagrama similar a un mapa.

UCM utiliza los recorridos de escenarios para ilustrar la relación causal entre responsabilidades. UCM provee una vista integrada del comportamiento y la estructura permitiendo la superposición de caminos de escenario en una estructura de componentes abstractos. Esto permite el razonamiento arquitectural después del cual es posible refinar las especificaciones de UCM en modelos de escenarios más detallados.

UCM cubre el hueco de modelado entre requisitos y el diseño, y provee un marco de trabajo para especificar el comportamiento. Permite dejar la definición concreta de los componentes de un escenario para fases más apropiadas del proceso de desarrollo.

Una especificación en UCM (de la cual se muestra un ejemplo de su uso en un diagrama en la **Figura 3**) describe los puntos de entrada y de salida de un escenario en un formato gráfico que trata de ser apropiado para modeladores y personal no especialista en el área debido a su aspecto visual simple y su naturaleza intuitiva a la vez que trata de ser lo suficientemente robusto para soportar el análisis de las características de los requisitos de calidad (verificable, completo, consistente, sin ambigüedades, entendible, trazable y modificable). UCM permite la transición entre lo informal y lo formal conectando los modelos de objetivos y los requisitos en lenguaje natural y diseño de una manera explícita y visual. Su principal objetivo es proveer el nivel preciso de formalidad en el momento adecuado del proceso de desarrollo.

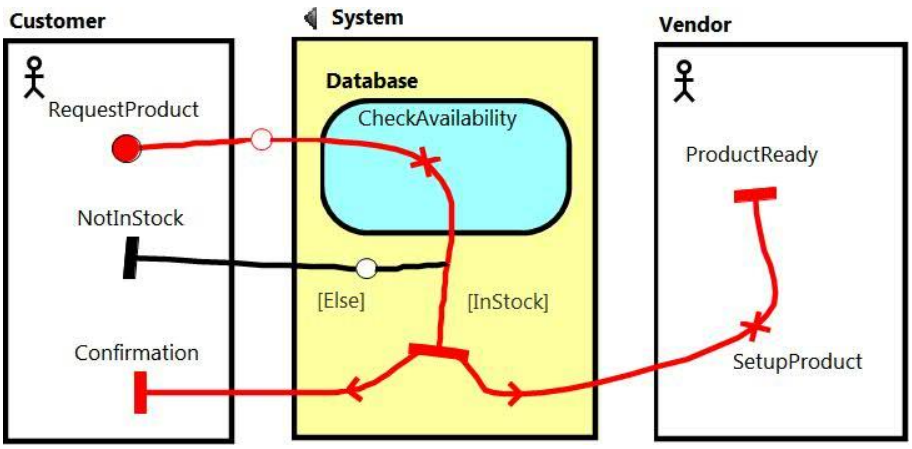


Figura 3: *Ejemplo de diagrama de UCM* [35].

2.5. Priorización de requisitos dirigida por el valor

La priorización de requisitos es un problema recurrente en la ingeniería del software en particular y la gestión de proyectos en general. La priorización es el proceso por el cual se ordena un conjunto de elementos según un criterio establecido, de manera que los elementos en los primeros puestos sean los más relevantes. En el presente caso, este criterio es el valor. De manera que los elementos más valiosos serán lo que tengan más prioridad para su inclusión en el proyecto debido a que estos aportan más valor en relación al resto.

En la literatura se encuentran multitud de técnicas de priorización que difieren únicamente en la aplicación de los criterios seleccionados para valorar la lista. Estos criterios pueden tener en cuenta diferentes características de los elementos siendo las más destacadas el coste, el valor o importancia y el riesgo. En esta sección se repasarán alguna de las técnicas halladas para la estimación de la importancia y el coste y se discutirá la selección de estas técnicas y su implementación en el proceso de desarrollo.

2.5.1. Estimación de importancia

La estimación de la importancia de un requisito es un tipo de estimación difícil de objetivar ya que la valoración de la importancia de un requisito depende de la perspectiva de los *stakeholders*, la experiencia de los mismos, e incluso puede variar dentro de las diferentes unidades dentro de la organización (pueden existir diferentes grupos dentro de la organización que asignarán un nivel de importancia diferente al mismo elemento debido a que este es más relevante según la posición que ocupan en la organización).

Aunque si es deseable una estimación de la importancia precisa, no necesita ser demasiado exhaustiva, ya que los tiempos del proceso de desarrollo son cortos y aunque de gran importancia para el proyecto, los riesgos por una estimación de la importancia de grano grueso son menores. Por ejemplo, al asignar un elemento en el quinto puesto y otro en el noveno puede no tener ningún efecto en el desarrollo ya que ambas se incluyen en la misma iteración. Por ello es común optar por establecer valores de importancia relativos, de manera que ayuda a determinar qué unidades del sistema tienen más importancia para el cliente, pudiendo priorizar entre ellas, sin establecer un método para especificar la importancia de manera mucho más precisa.

La priorización puede establecerse a nivel de requisitos funcionales, pero también a nivel de requisitos de calidad. Siendo el primero más común, en este último sentido en [36] se trata de descubrir cómo funciona la priorización de requisitos de calidad en la práctica en once compañías diferentes. Este estudio resalta que no existe un método en la práctica como los encontrados en la literatura y que las organizaciones tienden a priorizar entre los requisitos de manera *ad-hoc*.

A esta práctica le seguían otras técnicas de priorización, también comunes en la priorización de requisitos, como la asignación numérica, el método más tradicional y común basado en agrupar en diferentes categorías los requisitos, como las comunes “alto, medio y bajo”; las comparaciones entre pares como el *Analytical Hierarchy Process* (AHP), una técnica relativa en la que se comparan todos los elementos entre sí para signar el valor relativo entre cada par comparado; aproximamiento coste-valor, en el cual se compara en una gráfica de dos dimensiones el valor de los requisitos y su coste, de manear que cuentan con más importancia aquellos que tengan menor coste y mayor valor; el voto cumulativo (como la prueba de los 100$), consiste en la distribución de un valor fijo de puntos entre todos los requisitos de manera que los más valiosos reciban más puntos; hasta la más simple de ordenación en la que los requisitos se ordenan de mayor a menor importancia.

La mayoría de técnicas de priorización encontradas pueden ser clasificadas en uno o más de estos tipos ya que se tratan de variaciones o procesos diferentes sobre este mismo tipo de bases.

En [37] se presenta el estado del arte de las técnicas de priorización de requisitos encontrándose 49 técnicas de priorización siendo la más referenciada la ya mencionada AHP. En este trabajo también se establece una taxonomía de las escalas de las metodologías (escala ordinal, nominal, de intervalo, ratio).

Las técnicas que se encuentran clasificadas en estas taxonomías ordenan los requisitos en grupos o sub-grupos y números o inscripciones que son asignadas a través de todos los requisitos para reflejar su importancia relativa por lo que ningún requisito se espera a que pertenezca a más de un grupo o subgrupo al mismo tiempo.

En consecuencia, las frecuencias a través de estos grupos son calculadas y el requisito con la frecuencia más alta es tomado como el principal.

También en este trabajo, se describen los procesos utilizados en ellas que van desde la captura de requisitos, al ordenamiento, la exclusión, la aplicación de criterios de valoración, entrenar a los *stakeholders*, el agrupamiento, la representación de resultados finales etc.

Así mismo, en [37] también se encuentran limitaciones de los métodos existentes que se deben tener en cuenta para su selección y aplicación en la presente propuesta:

- **Escalabilidad**: Uno de los mayores escollos es la escalabilidad de los métodos, la complejidad computacional (tanto de los métodos automatizados como de los no automatizados) es un escollo para su implementación en proyectos de tamaño mediano o grande, siendo AHP el más lento y *planning game* el más rápido.
- **Gestión de cambios**: Los cambios en la valoración de un elemento o su ordenamiento son difíciles de incluir de forma sencilla y sin recurrir a volver a aplicar el método a la totalidad de los elementos.
- **Expresividad**: la mayoría de las técnicas no son aptas para comunicar los resultados con los *stakeholders* formados por personal no técnico.
- **Dependencias entre requisitos**: pocas técnicas permiten trazar dependencias entre varios elementos a valorar.
- **Poco precisas**: la mayoría de las técnicas tienden a tener errores, debido a que las reglas que gobiernan los procesos de priorización no son lo suficientemente robustas.
- **No implementadas en la práctica**: la mayoría de las técnicas revisadas no han sido implementadas para escenarios reales, se señala que probablemente debido a las complejidades asociadas con las priorizaciones y el tiempo requerido para generar requisitos priorizados.

Por otra parte, la priorización es uno de los componentes críticos en las prácticas ágiles, ya que como se ha visto anteriormente, su foco es entregar valor al cliente desde las primeras iteraciones por lo que seleccionar qué requisitos son de mayor valor para el cliente es un apartado crucial. Sin embargo, en la literatura cuesta encontrar ejemplos de cómo la priorización tiene lugar en la práctica[38]. En el estudio de Racheva *et al.* [38] sobre la priorización de requisitos en la práctica tratan de encontrar qué se refiere como valor y cuál es el proceso que en la industria se emplea para estimar este valor o importancia de cada requisito. En este estudio se refleja que la forma de valorar los requisitos, como en el anterior trabajo, suele ser *ad-hoc*, poco rigurosa y muchas veces limitada por los problemas para llegar a un acuerdo por parte de los diferentes *stakeholders* sobre la importancia de un ítem.

Un aspecto común entre las organizaciones investigadas es que la técnica de priorización emplee un criterio de coste/beneficio. Se descubrió en estas organizaciones que gran parte del esfuerzo para establecer la prioridad entre tareas lo llevaban a cabo los desarrolladores, mientras que los clientes se limitaban a validar las priorizaciones llevadas a cabo por estos, muchas veces siendo un solo representante de la organización cliente el que decidía sobre la validez de la estimación y por tanto aportando una visión sesgada de las necesidades reales de la organización.

De igual manera, el cliente suele tener problemas para involucrar a su personal con el proceso de desarrollo por falta de recursos, personal necesario para ayudar a establecer las prioridades entre otras tareas. También se resalta que, en muchas ocasiones, a pesar de conseguir los recursos para colaborar en el proyecto, los clientes no eran capaces de expresar sus necesidades, incluso llegando a afirmar que no existe un valor objetivo que sea posible tomar como entrada para el proceso de priorización.

No obstante, sí que existen técnicas de priorización, técnicas como el *Return of Investment* (ROI), que mide cuánto dinero se obtendrá de una inversión, se utilizan para estos mismos fines en proyectos tradicionales con éxito a nivel de proyecto. Sin embargo, en el caso de proyectos ágiles donde no es posible el uso de estas técnicas a nivel de proyecto: por sus costes, porque requieren de una gran cantidad de asunciones o porque ciertas actividades de valor para el cliente no tienen coste monetario.

En contraposición a este tipo de técnicas Jim Highsmith [39] propone una técnica llamada Análisis por Puntos de Valor (*Value Points Analysis*, VPA). Esta técnica consiste en valorar los ítems que puedan generar valor al cliente una vez implementadas o incluidas en el sistema de información, en el ejemplo de aplicación original, estos eran las historias de usuarios, con valores de una escala finita como la Secuencia de Fibonacci.

Establecer el valor relativo no solamente ayuda a la priorización de tareas, sino que también es útil para estimar el esfuerzo que es necesario invertir en cada ítem, si el valor es reducido, el coste de llevarlo a cabo también debería serlo ya que no es relevante o no tan relevante para el cliente como otro ítem de mayor valor en el que se debería invertir más esfuerzo si fuera necesario.

2.5.2. Estimación de esfuerzo

La estimación del esfuerzo es una técnica utilizada en el contexto del proceso de desarrollo cuyo objetivo es obtener una evaluación, lo más acertada posible de la cantidad de trabajo necesaria para desarrollar una unidad de trabajo (tarea, requisito, historia de usuario…).

Diferentes técnicas de estimación se han propuesto para esta tarea en la gestión de proyectos, siendo un tema recurrente no solamente en la ingeniería del software. Nótese que existe un solapamiento entre las técnicas para la estimación de la importancia y la del esfuerzo, sobre todo en las aplicadas en los métodos ágiles, esto es debido a que ambas técnicas se basan en los mismos principios para la estimación (por ejemplo, basar la estimación en las experiencias previas o el conocimiento del dominio) para realizar la tarea.

A continuación se citan algunas de las más utilizadas [40]:

- **El Juicio de Expertos**: en esta técnica se realiza la estimación en base a los conocimientos y experiencias previas de personas que han realizado un trabajo igual o semejante al cual se quiere determinar el esfuerzo. Este grupo de expertos determina el coste de cada una de las unidades de trabajo mediante consenso. Muchas veces se 'abusa' de esta técnica, debido a la falta de datos cuantitativos de proyectos anteriores.
- **Estimación por analogía (*Top-Down)***: se cuestiona el coste desde los elementos más generales a los más específicos. Es utilizada cuando se cuenta con experiencia en proyectos anteriores, análogos o similares, que pueden servir de referencia. Es una técnica menos costosa y más rápida, pero tiene como desventaja que es menos exacta y que se necesita de experiencia y documentación.

- **Estimación paramétrica**: consiste en una estimación con base a parámetros, aunque también puede utilizar datos de proyectos anteriores y datos de referencia. Su principal característica es que la estimación se realiza con base a la relación entre variables, por ejemplo, coste por cantidades producidas, horas hombre por desarrollo, transporte por recorrido, etc.
- **Estimación ascendente (*Bottom-Up*)**: este método de estimación asciende desde la estimación del detalle de cada elemento hacia el elemento más general del objetivo del proyecto. Es una técnica costosa ya que hay que examinar todos los elementos del proyecto en su último detalle para poder abstraer la estimación de los elementos generales. Trabajar a este nivel de detalle permite ser más preciso en las estimaciones.
- **Estimación con tres valores**: también llamada de tres puntos, derivada de la Técnica de Revisión y Evaluación de Proyecto, conocida como método PERT (*Project Evaluation and Review Techniques*), consiste en identificar tres posibles valores, el optimista, el pesimista y el más probable, para llegar a un único valor aproximado. Esta técnica es utilizada en escenarios inciertos y con diferentes antecedentes o muchas variables que puedan afectar el valor final. Existen 2 fórmulas principales, la primera basada en una distribución triangular que en forma de promedio toma la sumatoria del valor optimista, el pesimista y el más probable y lo divide entre 3. La segunda fórmula está basada en una Distribución Beta y es como se muestra a continuación:

  Coste Estimado = (Optimista + 4 (Más Probable) + Pesimista) / 6

- **El análisis de la reserva**: es una técnica complementaria a la estimación del coste, permite, basado en la incertidumbre, estimar una cantidad adicional al coste que se ha identificado, generando lo que se conoce como "reserva de contingencia". Se debe utilizar cuando a la actividad en la cual recae el coste le ha sido identificado algún riesgo. Para calcular esta reserva se utiliza el Análisis del Valor Monetario Esperado (VME), para lo cual, es necesario que el riesgo haya sido valorado de manera cuantitativa, es decir, que su impacto haya sido estimado en términos de dinero y/o tiempo. El Valor Monetario Esperado: es el resultado producto del impacto (consecuencia) del riesgo en dinero por su probabilidad de ocurrencia, esto es:

  VME = Impacto x Probabilidad

La Reserva de Contingencia será calculada a través de una combinación estadística de los Valores Monetarios Esperados. Para ello deberíamos combinar adecuadamente todos los sucesos, típicamente con herramientas de Simulación de Monte Carlo. Si identificamos un riesgo que eventualmente pueda impactar positivamente al proyecto (Oportunidad) entonces el VME será un valor negativo y en consecuencia disminuye la reserva de contingencia.

En cuanto a la precisión de las estimaciones de coste, se debe tener en cuenta el momento en el que estas se llevan a cabo en el ciclo de vida del proyecto. En el PMBOK [41] se indica que al inicio del proyecto las estimaciones son de tipo ROM (*Rough Order of Magnitude*, Orden de Magnitud Aproximado o Bruto) con una precisión de -50% a % a +75%, cifra que se va reduciendo según las fases del proyecto se van desarrollando (Ver **Figura 4**).

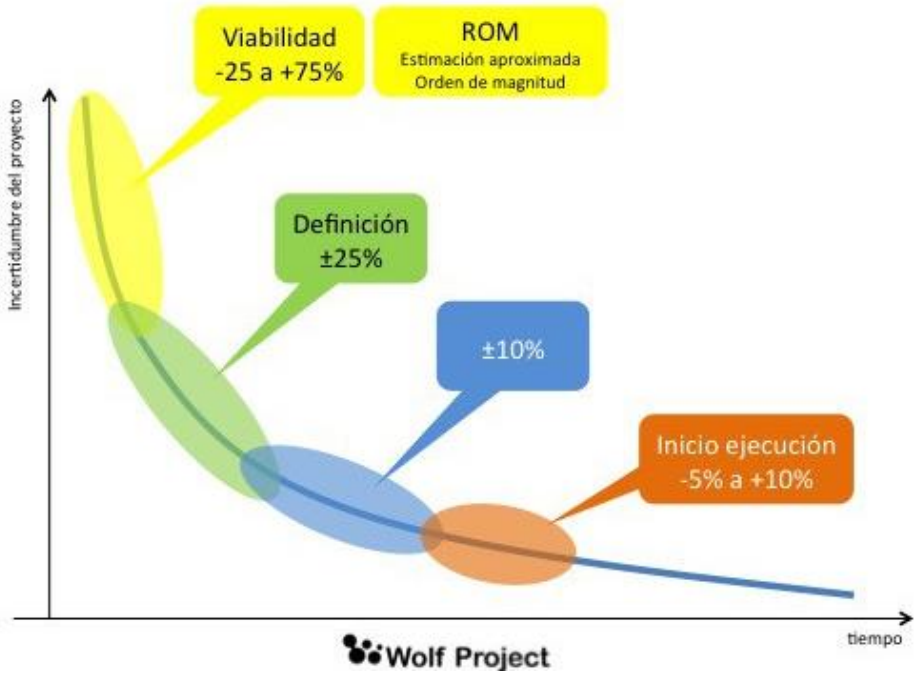


Figura 4: *Estimación aproximada de orden de magnitud a lo largo de un proyecto* [42].

Por otra parte, en el ámbito del Desarrollo de Software Ágil las técnicas de estimación tienen un gran peso debido al carácter iterativo y de entrega continua de los proyectos donde las estimaciones tienen que hacerse de forma progresiva. En [43] se realiza una clasificación general de los métodos de estimación de coste y esfuerzo:

- **Aproximamiento orientado al aprendizaje**: Basado en aprender de forma colectiva de las experiencias de estimación previas y el conocimiento de varios gestores obtenidos de los resultados de varios proyectos específicos. En el trabajo se critica a esta corriente porque suele proveer con estimaciones poco realistas.

- **Aproximamiento basado en expertos**: Un experto compara el proyecto con proyectos similares pasados basándose en la experiencia personal. El problema de este método es la alta subjetividad del mismo.
- **Métodos de regresión**: Este modelo se basa en datos de regresión y en desarrollar ecuaciones de regresión para hacer las estimaciones. Este método no promueve las buenas prácticas de la ingeniería de software.
- ***Bottom-up***: Cada componente del sistema software se estima de forma separada y los resultados son agregados para producir una estimación del sistema completo. La dificultad reside en cómo descomponer el sistema en componentes estimables.

En [44] se realiza una revisión de las técnicas de estimación, que pueden ser incluidas en esa clasificación. Algunas de las más comunes son la opinión de expertos (similar al ya nombrado aproximamiento basado en expertos), Analogía y disgregación (aproximamiento basado en aprendizaje y regresión) y el *Planning Poker* (aproximamiento basado en expertos).

Planning Poker es una técnica de estimación basada en el consenso y la gamificación. Consiste en que los miembros del grupo de desarrolladores hacen estimaciones mostrando cartas numeradas, de forma simultánea, en vez de hacerlo de forma oral. Las puntuaciones reveladas son entonces discutidas y en caso de fuertes desacuerdos se procede a repetir la votación teniendo en cuenta la discusión. El uso de esta técnica evita el pensamiento en grupo ya que, de forma oral, las primeras estimaciones sentarían un precedente para el resto [45]. En [44] se citan varios trabajos que concluyen que las estimaciones utilizando esta técnica mejoran a las realizadas de manera no estructurada o realizadas por expertos, aunque existe una dependencia de la experiencia de los practicantes de esta técnica.

2.6. Ingeniería dirigida por modelos (MDE)

La ingeniería dirigida por modelos (MDE Model Driven Engineering) es una disciplina de la ingeniería del software que se basa en el uso de modelos como artefactos de primera clase y que tiene como objetivo desarrollar, mantener y evolucionar software por medio de transformaciones de modelos. El proceso de desarrollo bajo MDE es llamado Desarrollo de Software Dirigido por Modelos (DSDM) [46].

MDE ofrece un enfoque más efectivo que la programación habitual, ya que los modelos son partes activas del proceso de desarrollo de software. Los modelos MDE tienen el significado exacto del código del programa, en el sentido de que la mayor parte de la aplicación final, no solo la clase y los esqueletos de los métodos pueden ser generados a partir de ellos [47]. En este caso los modelos ya no son solo la documentación, sino partes del software, lo que constituye un factor decisivo para aumentar la velocidad y calidad de desarrollo de software.

Un enfoque basado en modelos requiere lenguajes para la especificación de modelos, definición de transformación y descripción del metamodelo. MDE propone el uso de transformaciones de modelos con el fin de transformar un modelo en otro y también para producir el producto final.

MDE es aceptado por diversas organizaciones y empresas que incluyen a la OMG, IBM y Microsoft.

Existen cinco cosas que una infraestructura de soporte MDE debe definir:

- Conceptos disponibles para la creación de modelos y reglas claras que rigen su uso.
- La notación a utilizar para describir los modelos.
- Tiene que ser claro, cómo los elementos del modelo representan los elementos del mundo real y los artefactos de software.
- Conceptos para facilitar las extensiones de usuarios dinámicos para modelar conceptos, modelos de notación y los modelos creados a partir de ellos.
- Conceptos para facilitar el intercambio de conceptos de modelos y notación, y los modelos creados a partir de conceptos definidos por el usuario para facilitar las asignaciones de los modelos a otros artefactos.

2.6.1. Desarrollo de software dirigido por modelos (DSDM)

El Desarrollo de Software Dirigido por Modelos (DSDM) es un enfoque de desarrollo de software basado en modelos, donde un modelo puede definir la funcionalidad, estructura o comportamiento de un sistema. Los modelos permiten trabajar con un nivel de abstracción más cerca a los conceptos de dominio, en lugar de centrarse en conceptos orientados a la plataforma como el desarrollo del software tradicional. El objetivo de esta propuesta es maximizar la productividad e incrementar la interoperabilidad entre sistemas para facilitar la reutilización y adaptación del cambio tecnológico.

Además, cualquier artefacto software es considerado un modelo o un elemento del modelo. Mientras que en los enfoques anteriores se utilizaban modelos para la documentación o la comunicación de ideas. DSDM es una aproximación abierta e integradora no vinculada a una norma especial, por lo tanto, existen diferentes implementaciones.

2.6.1.1. Modelos

Un modelo es una simplificación (o una descripción abstracta) de una parte del mundo llamado sistema, construido con un objetivo previsto en la mente; debería ser más fácil de usar y entender que el sistema original y debe ser capaz de responder a preguntas sobre el sistema. Las respuestas proporcionadas por el modelo deben ser exactamente las mismas que las dadas por el sistema en el que se basa el modelo [48], [49].

Los modelos pueden consistir en un conjunto de elementos con una representación gráfica o textual. Mientras que esto sirve como un punto de partida, Kleppe et al. [50] da una definición aún más dirigida al DSDM. Un modelo es una descripción de un (parte de) sistema escrito en un lenguaje bien definido. Un lenguaje bien definido es un lenguaje con una forma bien definida (sintaxis) y significado (semántica), lo que es conveniente para la interpretación automatizada por un computador.

La idea de MDE es la creación de diferentes modelos de un sistema en diferentes niveles de abstracción. Cada modelo representa un aspecto determinado del sistema.

De acuerdo con estas definiciones, el código fuente también es un modelo. Siendo el código fuente una representación simplificada de la estructura interior de la máquina y las operaciones que son necesarias para automatizar las tareas en el mundo real.

Tipos de modelos

El estándar MDA [51], que representa una forma específica de MDE, define cuatro tipos de modelos en el ciclo de vida del desarrollo de software:

Modelo Independiente de Cómputo (CIM)

Un modelo independiente de cómputo es una vista del punto de vista independiente de cómputo, que se centra en el entorno y en los requisitos del sistema. Los detalles estructurales o de procesamiento del sistema son ocultados o todavía no son determinados. Un CIM algunas veces es llamado un modelo de dominio y un vocabulario que es familiar para los profesionales del dominio en cuestión y utilizan en su especificación [51]. Se centra en el entorno del sistema y en los requisitos que el usuario tiene en el sistema, la descripción proporciona lo que se espera que el sistema debe hacer.

Modelo Independiente de Plataforma (PIM)

El modelo independiente de plataforma se centra en el funcionamiento de un sistema al mismo tiempo ocultando los detalles necesarios para una determinada plataforma. Un PIM exhibe un determinado grado de independencia de la plataforma con el fin de ser aptos para su uso con un número de diferentes plataformas de tipo similar. También es una representación de la funcionalidad y comportamiento del negocio, sin distorsión por los detalles de la tecnología. Además, la especificación no cambia entre plataformas.

El objetivo es posponer el proceso de desarrollo de creación de modelos tanto como sea posible, para ello el modelo generado no deberá tener en cuenta los aspectos tecnológicos de la plataforma. La principal ventaja de realizar esto es la de ser capaz de reaccionar eficientemente y con bajos costos a los cambios de tecnología.

Modelo Específico de Plataforma (PSM)

Un modelo específico de plataforma es una combinación de un PIM con el detalle adicional de la plataforma específica del sistema.

Modelo de Plataforma

Finalmente, los modelos de plataforma son la representación de conceptos técnicos de partes de la plataforma, los servicios prestados por esa plataforma y para su uso en un posterior PSM, los conceptos que modela el uso de la plataforma por las aplicaciones.

2.6.1.2. Metamodelos

La palabra "meta" es griega y significa "arriba", por lo tanto, el término metamodelo puede interpretarse como un modelo que describe otro modelo. Un lenguaje consiste en palabras cuya combinación es restringida por una gramática. Si una sentencia en un lenguaje es vista como un posible modelo, la definición de su estructura, la gramática puede ser vista como su metamodelo. En el DSDM un metamodelo define como un modelo puede ser visto, esto puede ser formulado de manera más precisa como: un metamodelo define los constructores y las reglas que se utilizan para crear una clase de modelos.

Esto es consistente con la siguiente definición:

"*Un metamodelo es un modelo de un conjunto de modelos* [52]."

"*Un metamodelo es un modelo que define el lenguaje para expresar un modelo* [53]."

Un metamodelo es un modelo de especificación que describe sus modelos en un cierto lenguaje de modelado. Un metamodelo dice lo que se puede expresar en un modelo válido del lenguaje de modelado.

La interpretación de un metamodelo es el mapeo de elementos del metamodelo a elementos del lenguaje de modelado. El valor de verdad de las declaraciones en el metamodelo puede determinarse por cualquier modelo expresado en el lenguaje de modelado. Puesto que el metamodelo es la especificación de modelos, un modelo en el lenguaje de modelado es válido sólo si ninguna de estas afirmaciones es falsa. Los metamodelos precisos son un requisito previo para la realización de transformación de modelos automáticas y para definir modelos precisos.

Puesto que un metamodelo es también un modelo, podría ser expresado en algunos lenguajes de modelado. Un metamodelo para un lenguaje de modelado podría utilizar el mismo lenguaje de modelado. Las afirmaciones en el metamodelo se expresan en el mismo lenguaje que está siendo descrito por el metamodelo. Esto es llamado metamodelo reflexivo.

2.6.1.3. Meta-Object Facility (MOF)

Meta-Object Facility (MOF) es un estándar propuesto y definido por la OMG [53] para soportar a MDA. Esta norma propone cuatro niveles de arquitectura metamodelos con cuatro meta capas como se muestra en la **Figura 5**. Cada meta capa es el metamodelo de los constructores en las capas anteriores.

Las capas se describen a continuación:

1. **Capa M3 Metametamodelo**: En esta capa reside el metametamodelo, un lenguaje para definir los metamodelos del nivel M2. En el estándar de OMG, MOF es el lenguaje definido en esta capa.
2. **Capa M2 Metamodelo**: Los metamodelos M2 se utilizan para describir los modelos M1. El metamodelo de UML de la OMG describirá los constructores de UML.
3. **Capa M1 Modelo**: En este nivel es donde se definen los modelos. Un modelo es una instancia de un metamodelo M2. Es decir, si en la capa M2 reside el metamodelo de UML, en M1 podríamos tener uno de sus modelos: diagrama de clases, diagrama de actividad, diagrama de secuencia y así sucesivamente.
4. **Capa M0 Instancia**: En esta capa se definen objetos del mundo real.

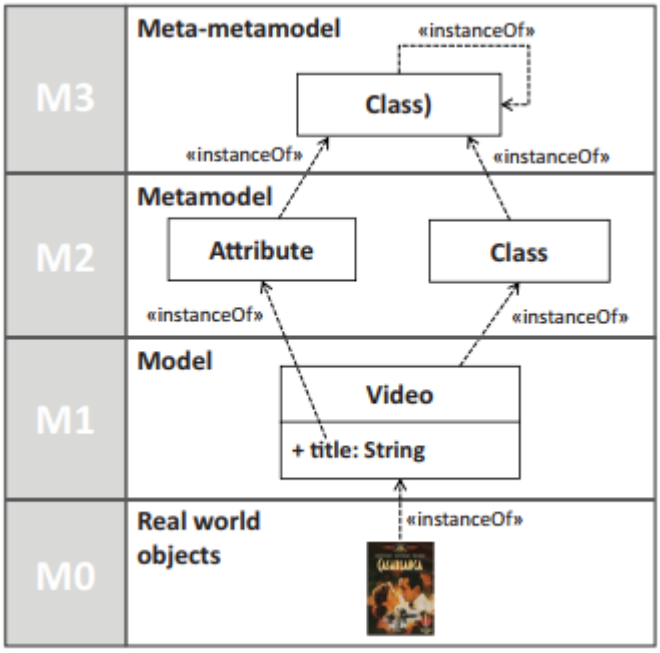


Figura 5*:* Capas de Arquitectura MOF [46].

MOF es una arquitectura de metamodelado cerrada. Esto significa que el nivel M3 podría definirse con instancias de elementos de M3. Esto implica que podemos definir MOF.

Además, MOF proporciona conceptos para definir un lenguaje:

- Clases, que son metaobjetos del modelo MOF.
- Tipos de datos (propiedad), que son necesarios para el modelo de datos descriptivos (es decir, los tipos primitivos).
- Asociaciones (propiedad), que son las relaciones binarias entre los metaobjectos del modelo.
- Paquetes, que son un conjunto de modelos.

2.6.1.4. Transformación de Modelos

Transformación de modelos: es el proceso de conversión de un modelo a otro modelo del mismo sistema [51]. Un modelo puede ser transformado a varios modelos alternativos que pueden mantener la semántica, pero con diferente sintaxis (Ver **Figura *6***).

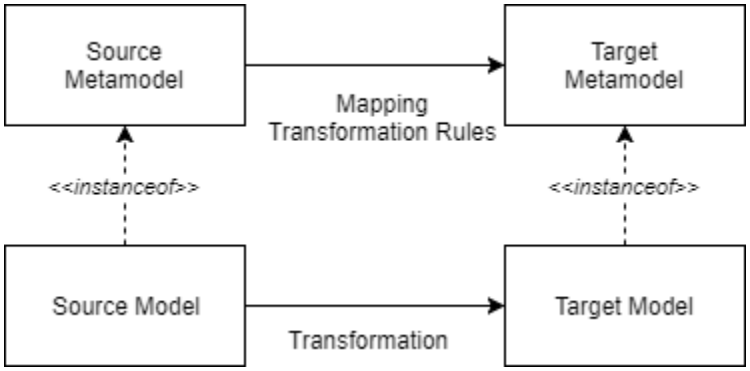


Figura 6: *Definición de una transformación de modelos.*

Una definición de transformación consiste en una colección de reglas de transformación que son especificaciones inequívocas de la manera que (parte de) un modelo puede utilizarse para crear una pieza de otro modelo. Las transformaciones son definidas en términos de los metamodelos involucrados en el proceso de transformación.

Una transformación, una definición de transformación y sus reglas de transformación pueden definirse de la siguiente manera [50]:

- Una transformación es la generación automática de un modelo destino desde un modelo de fuente, según una definición de transformación.
- Una definición de transformación es un conjunto de reglas de transformación que juntas describen cómo un modelo origen puede ser transformado en un modelo destino.

- Una regla de transformación es una descripción de cómo una o más constructores en el lenguaje origen pueden ser transformados en uno o más constructores de un lenguaje destino.

La característica más importante de una transformación es el hecho de que una transformación de modelos debe mantener el significado entre el modelo origen y el modelo destino. En este punto hay que decir que el significado del modelo sólo puede ser preservado y expresado en el modelo origen y destino. Parte de la información pueden perderse si el lenguaje de destino es menos expresivo que el lenguaje de origen.

Un mapeo se define como una transformación unidireccional en contraste a una relación que define una transformación bidireccional.

2.6.1.5. Lenguaje de restricción de objetos (OCL)

Lenguaje de restricciones de objetos (OCL) [54] es un lenguaje formal usado para describir las expresiones de modelos UML. Estas expresiones suelen especificar condiciones invariables que se deben mantener en el modelado del sistema o consultas sobre los objetos descritos en un modelo. Tenga en cuenta que cuando OCL evalúa las expresiones, que no tienen efectos secundarios; es decir, su evaluación no puede alterar el estado de ejecución del correspondiente sistema.

Las expresiones OCL pueden ser usadas para especificar operaciones/acciones que, cuando son ejecutadas, no alteraran el estado del sistema. Los modeladores UML pueden utilizar OCL para especificar las restricciones específicas de la aplicación en sus modelos. Los modeladores UML pueden también utilizar OCL para especificar las preguntas sobre el modelo de UML, que son totalmente lenguaje de programación independiente.

OCL no sólo se puede aplicar en modelos UML, pero también puede ser aplicado en UML o metamodelos MOF ya que están expresados en UML o un subconjunto de UML.

Por lo tanto, OCL puede ser utilizado para restringir la semántica del metamodelo, por ejemplo, por medio de estereotipos o lenguajes de dominio específicos (LDE).

En el contexto MDA, OCL se puede utilizar de tres formas:

- Modelos precisos en M1 a nivel MOF.
- Definición de lenguajes de modelado.
- Definición de transformaciones.

2.6.1.6. ATL (Atlas Transformation Language)

Es un lenguaje de dominio específico para la especificación de transformaciones de un modelo a otro [55], [55], [56]. ATL está inspirado en los requerimientos del estándar QVT [57] de la OMG y se basa en el formalismo de OCL [54].

La decisión de utilizar OCL está motivada por su amplia adopción en MDE y el hecho de que es un lenguaje estándar apoyado por OMG y de las principales herramientas de los proveedores.

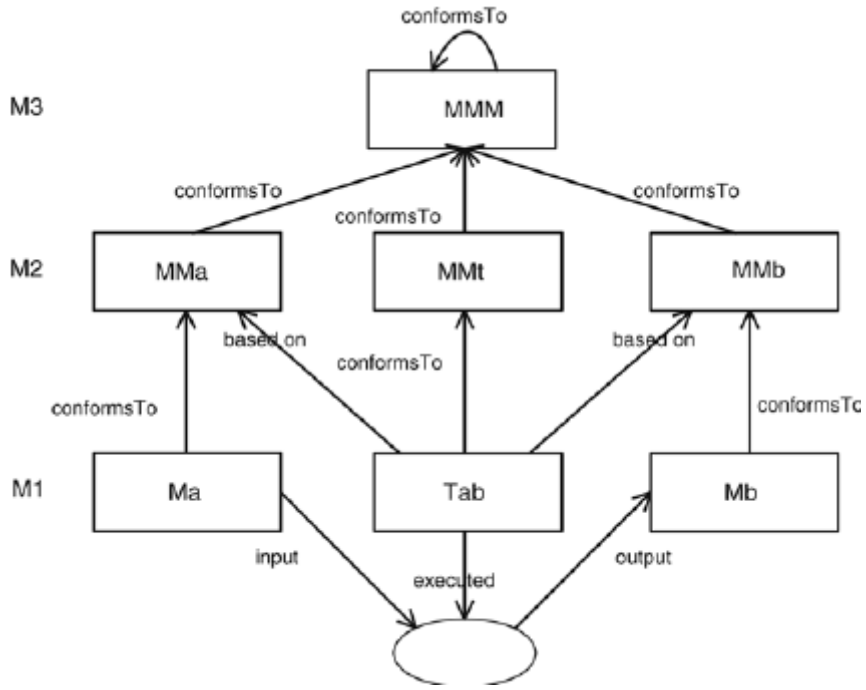


Figura 7: *Patrón de transformación de modelos.*

ATL es aplicado en el contexto del patrón de transformación que se muestra en la **Figura 7**. Este patrón es un modelo origen *Ma* que se transforma en un modelo destino *Mb* según una definición de transformación *mma2mmb.atl* escrita en el lenguaje ATL. La definición de transformación es un modelo conforme al metamodelo de ATL [55]. Todos los metamodelos son conformes a MOF.

ATL es un lenguaje híbrido, es decir, que ofrece una mezcla de construcciones declarativas e imperativas. El estilo declarativo de especificación de transformación tiene un número de ventajas. Generalmente está basado en la especificación de las relaciones entre los patrones origen y destino y por lo tanto tiende a estar más cerca de la forma en que los desarrolladores perciben intuitivamente una transformación [55].

Las transformaciones ATL son unidireccionales, es decir, operan en modelos origen sólo para lectura y en modelos de destino para escritura. Durante la ejecución de una transformación, el modelo origen puede ser navegado, pero los cambios en él no están permitidos, y en un modelo destino es imposible ser navegado [55] . Una transformación bidireccional se implementa como un par de transformaciones: una para cada dirección. Los modelos origen y destino de ATL pueden expresarse en el formato de serialización XMI de la OMG. Los metamodelos origen y destino también pueden expresarse en XMI o en la notación más conveniente [55].

Las definiciones de transformación en ATL se realizan en módulos (*module*). Un módulo contiene una sección obligatoria de cabecera (*header*), una sección de importación (*import*), y un número de ayudantes (*helpers*), y reglas de transformación (transformation rules). La sección de cabecera da el nombre al módulo de transformación y declara los modelos origen y destino. A continuación, se detalla un ejemplo de la sección de cabecera:

```
module Class2Relational;
create OUT : Relational from IN : Class;
```

Código 1: *Ejemplo de definición de transformación en ATL.*

La sección ***header*** comienza con la palabra clave *module* seguida por el nombre del módulo. Luego, los modelos origen y destino se declaran como variables introducidas por sus metamodelos [55]. La palabra clave *create* indica los modelos de destino. La palabra clave *from* indica los modelos origen. En este ejemplo, el modelo destino junto a la variable *OUT* es creado del modelo origen *IN*. Los modelos origen y destino se ajustan a los metamodelos *Class* y *Relational* respectivamente. En general, más de un modelo origen y destino pueden ser enumerados en la sección de cabecera [55].

Un ***helper*** define las operaciones en el contexto de los elementos de un modelo o en el contexto de un módulo. Ellos pueden tener parámetros de entrada y pueden utilizar la recursividad. Un *helper attribute* se utiliza para asociar valores de sólo lectura nombrados en los elementos de un modelo. De igual forma las operaciones tienen un nombre, un contexto y un tipo. La diferencia es que ellas no pueden tener parámetros de entrada. Sus valores se especifican mediante una expresión OCL. Para ilustrar la sintaxis de los *helpers* se presenta el siguiente ejemplo [55]:

```
helper context String def: firstToLower() : String =
self.substring(1, 1).toLower() + self.substring(2, self.size());
```

Código 2: *Ejemplo de definición de un helper en ATL.*

Las reglas de transformación es la construcción básica en ATL que se utiliza para expresar la lógica de transformación. Las reglas ATL pueden especificarse en un estilo declarativo o en un estilo imperativo [55].

Matched Rules: Las *matched rules* son reglas ATL declarativas. Una *matched rule* se compone de un patrón de origen y un patrón de destino. La regla del patrón de origen regla especifica un conjunto de tipos de destino (procedentes de los metamodelos de origen y desde el conjunto de tipos disponibles de colecciones en OCL) y un *guard* (una expresión OCL booleana). Un patrón de origen se evalúa a un conjunto de coincidencias en el modelo origen [55].

El patrón de destino se compone de un conjunto de elementos. Cada elemento especifica un tipo de destino (del metamodelo destino) y un conjunto de enlaces. Un enlace se refiere a una característica del tipo de destino (es decir, un atributo, una referencia, o un extremo de la asociación) y especifica una expresión de inicialización para el valor de característica [55]. El siguiente fragmento de código muestra una simple *matched rule*. Esta regla implementa la lógica para la transformación de las clases a las tablas.

```
rule Class2Table
   {
      from
         c : Class!Class
      to
         out : Relational !Table
            (
               name <- c.name,
               col <- Sequence {key}->union(c.attr->select(e | not
e .multiValued)),
               key <- Set {key}
            ),
         key : Relational!Column
            (
               name <- 'objectId',
               type <- thisModule.objectIdType
            )
   }
```

Código 3: *Ejemplo de transformación de una matched rule de una transformación de clase a tabla.*

Tipos de *mathed rules*: Hay varios tipos de *matched rules* que difieren en la forma en que se desencadenan [55].

- ***Standard rule*** son aplicadas una vez para todas las combinaciones que se puedan encontrar en los modelos origen [55].
- ***Lazy rule*** son provocados por otras reglas. Se aplican en una sola combinación tantas veces como es referida por otras reglas, cada vez que produzca un nuevo conjunto de elementos de destino [55].

- ***Unique lazy rule*** también son provocados por otras reglas. Se aplican una sola vez por una combinación determinada. Si una *unique lazy rule* se activa más tarde en la misma combinación, esta utiliza los elementos de destino ya creados [55].
- **Herencia de reglas**. En ATL, la herencia de reglas se puede utilizar como un mecanismo de reutilización de código, y también como un mecanismo para especificar reglas polimórficas [55].

Una regla (llamada subregla) puede heredar de otra regla (regla padre). Una subregla coincide con un subconjunto de las combinaciones de la regla padre. Los tipos patrón de origen de la regla padre podrán ser sustituidos por sus subtipos en el patrón de origen de la subregla [55].

ATL tiene una parte **imperativa** en base a dos conceptos principales:

- ***Called rules***: Una *called rule* es básicamente un procedimiento: se invoca por su nombre y puede tomar argumentos [55].
- ***Action block***: Un *action block* es una secuencia de sentencias imperativas, y se puede utilizar en lugar de o en una combinación con un patrón de destino en combinación o en *called rules* [55].

Las declaraciones imperativas disponibles en ATL son las construcciones conocidas para la especificación de un flujo de control, tales como las condiciones, bucles, asignaciones, etc. [55].

3. Trabajos relacionados

El trabajo de mayor importancia relacionado con este trabajo de fin de máster es [4] donde se presenta por primera vez el método de especificación de valor para la derivación y priorización de procesos de negocio. En él, se presenta una primera aproximación al método donde se centra en la primera actividad, por lo que deja el resto de actividades como propuestas sin desarrollarlas en mucha profundidad, cosa que se pretende solventar en este trabajo.

Pese a que el valor tiene definiciones vagas ello no ha impedido que sea utilizado en el desarrollo de software, llegando incluso a crear una nueva rama dentro de la misma denominada "*Value Based Software Engineering*" (Ingeniería de Software basada en valor) presentada por Biffl et al (2006) en [58] donde se recopila el estado de arte sobre cómo se utiliza el valor en la ingeniería de software, para ayudar a la toma de decisiones.

Como resultado al uso de valor en la ingeniería de software, comenzaron a aparecer múltiples lenguajes de modelado, los cuales ofrecen la posibilidad de modelar el valor. Uno de los lenguajes más conocidos es el de e³-value propuesto por Gordijn et al. en [59]. Lenguaje de modelado gráfico que permite representar las proposiciones de valor ofrecidas por los distintos implicados en el sistema y las relaciones establecidas entre ellos.

El trabajo [60] resulta sumamente interesante en relación a los lenguajes de modelado de objetivos que podrían utilizarse para modelar valor debido a que realiza una comparativa entre los mismos, y que ayuda a que este método seleccionara GRL como base para crear Value@GRL.

Otro de los trabajos más importantes es el propuesto por Amyot et al. [61]. En este trabajo se propone un método de propagación para calcular la satisfacción de los elementos intencionales. Utiliza un lenguaje de modelado extensión de GRL, de forma semejante a la nuestra, y de igual forma propaga la importancia para calcular el valor o satisfacción de los elementos intencionales. Las principales diferencias que hay con este trabajo son que la propagación de la importancia la hacemos de forma distinta, ya que en el trabajo de Amyot, la importancia se propaga de los elementos hijos a los padres, lo cual pensamos que puede ser una amenaza ya que el padre que tenga más hijos debido a que se ha especificado más sus acciones será aquel que tendrá más importancia, cosa que puede ser problemática debido a la libertad del lenguaje. Por otro lado, la propuesta se utiliza para generar UCM (*Use Case Model*) modelos de caso de uso, mientras que en nuestra propuesta generamos BPMN (*Bussines Process Model*) modelos de proceso de negocio, los cuales luego utilizamos no solo para seleccionar las actividades a priorizar sino también para ayuda a los desarrolladores a tener una idea más clara del dominio en el que están trabajando.

Una de las técnicas más conocidas para la propagación y priorización de elementos es la de AHP (*Analytical Hierarchy Process*) [16], técnica que sobre una lista de requisitos valora las dimensiones de coste y valor de cada requisito. Esta técnica realiza una comparación por pares, comparando un requisito con otro, teniendo en cuenta la importancia/coste. En esta técnica no se ha profundizado en el significado del valor para el cliente, pero podría utilizarse en este contexto concreto.

Otra de las técnicas que podría utilizarse para propagar el valor sería la técnica de VOP (*Value Oriented Prioritization*) propuesta en [62] donde se propone un mecanismo para cuantificar y ordenar requisitos para incrementos de una aplicación. Para ello se genera una matriz, donde se ponderan los requisitos candidatos según la contribución que realizan a las distintas áreas de negocio, después se realiza la suma total y se ordenan según el valor calculado.

Una aproximación al uso de valor, en el campo del desarrollo de software dirigido por modelos es el realizado por De Castro et al. [63][64] donde propone resolver el problema de obtener servicios y especificaciones de procesos de negocio mediante el uso de transformaciones de modelos. En este trabajo se propone el modelado de valor utilizando e³-value para luego transformarlo en un modelo de procesos de negocio, en este caso BPMN, donde se representa el software desde el punto de vista del negocio.

Por último, existen diversas herramientas que permiten la utilización de modelos de procesos como los de BPMN para definir historias de usuario, las cuales son utilizadas en métodos ágiles como scrum para seleccionar los elementos que aparecerán en el backlog, algunas de estas herramientas son la de visual-paradigm [65] y IBM BPM [66]

4. Método de especificación de valor para la derivación y priorización de procesos de negocio

4.1. Objetivo del método

El objetivo de este método es el de proveer de un método para ayudar a los desarrolladores a la hora de generar nuevo software, desde las primeras fases, como la elicitación de requisitos, utilizando requisitos basados en el valor que el cliente les otorga, para así poder ordenar los requisitos de mayor a menor importancia, para saber qué requisitos le interesan más al cliente, así como en qué orden deberían implementarse.

Cabe destacar que este método **no es ágil**, pero que sus características pueden aprovecharse para desarrollar aplicaciones debido a que sus características permiten priorizar los elementos a integrar en un incremento.

4.2. Descripción del método

El método está compuesto principalmente por cuatro actividades (modelado de objetivos, definición de modelado de valor, definición de procesos de negocio de alto nivel y priorización de actividades del proceso de negocio), las cuales mostramos en la **Figura *8***.

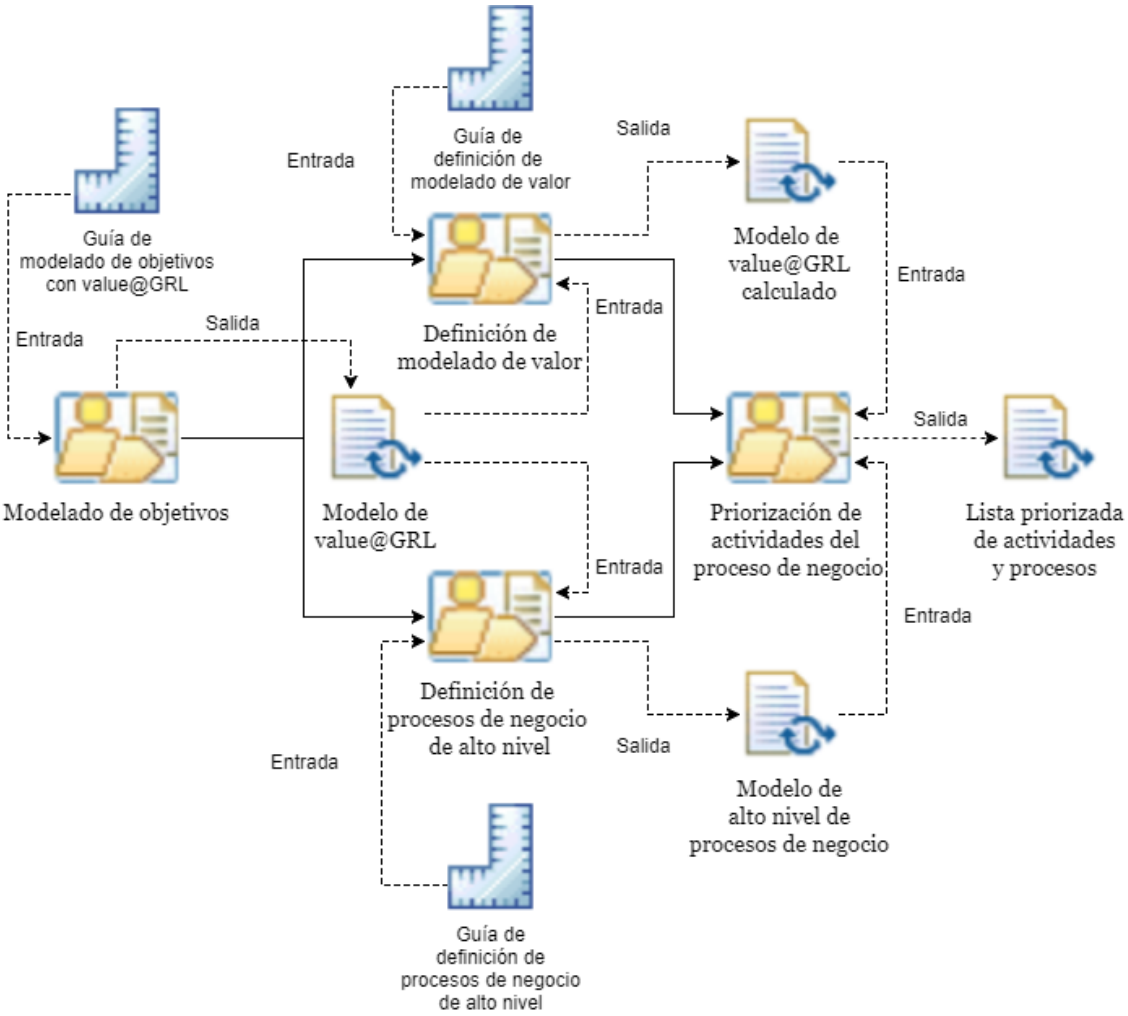


Figura 8: *Actividades del método de especificación de valor para la derivación y priorización de procesos de negocio.*

4.2.1. Modelado de objetivos

La primera actividad de este método es la de modelado de objetivos donde se realiza un modelo de objetivos para representar las necesidades que tiene cada uno de los distintos actores del sistema.

Los participantes en esta actividad son el experto en el domino, los *stakeholders* y los analistas.

La salida de esta actividad es un modelo de objetivos, en nuestro caso un modelo de Value@GRL (tipo de modelo que hemos desarrollado explícitamente para este método y que explicaremos más adelante) donde se representan los objetivos y tareas de cada uno de los actores del sistema.

4.2.2. Definición de modelado de valor

Esta actividad está compuesta por dos subactividades, en la primera se le asigna la **importancia** a cada uno de los elementos intencionales del modelo de objetivos. En la segunda se propaga la importancia asignada para calcular el **valor** de cada elemento de acuerdo con las relaciones que posea.

Los participantes en esta actividad a la hora de asignar la importancia a cada uno de los elementos intencionales del modelo son los mismos que los de la actividad anterior, es decir, el experto en el dominio, los *stakeholders* y los analistas. Cabe destacar que cada uno de los actores participantes asignará la importancia únicamente a los elementos intencionales de su actor correspondiente en el modelo.

La entrada de esta actividad es un modelo de objetivos, el cual se genera en la primera actividad del método. La salida es un modelo de valor, donde se ha propagado la importancia del modelo de objetivos para poder así calcular el valor de cada uno de los elementos intencionales del modelo.

4.2.3. Definición de procesos de negocio de alto nivel

Esta actividad está compuesta por dos actividades más pequeñas, siendo una la transformación del modelo de objetivos de la primera actividad del método a un modelo de procesos de negocio (en este caso BPMN), donde se representan las actividades que hay que realizar. La otra actividad es la de modificar el modelo generado en la transformación para añadir el flujo de control y realizar los cambios que vean oportunos, actividad que debe ser realizada por un experto en el dominio.

La salida de esta actividad es un modelo de procesos de negocio, en nuestro caso hemos seleccionado BPMN, donde se representa a un nivel alto las actividades que se quieren realizar en el proyecto.

4.2.4. Priorización de actividades del proceso de negocio

Esta actividad tiene como entrada el modelo de valor calculado y el modelo de procesos de negocio correspondientes con las salidas de las actividades anteriores de propagación de modelado de valor y definición de procesos de negocio de alto nivel. El objetivo de esta actividad es el de juntar ambas salidas para poder proveer de una lista de actividades y procesos ordenada según el valor que tenga cada uno de los elementos de la lista.

Al finalizar esta actividad, los desarrolladores cuentan con un modelo de procesos de negocio, así como con la lista priorizada de actividades de acuerdo con su valor calculado. Donde los desarrolladores deberán basarse en estos dos

artefactos para seleccionar que actividades deberán incluir en el siguiente incremento.

4.3. Extensión y mejora respecto a la versión preliminar

Como hemos dicho anteriormente, uno de los principales objetivos de este trabajo de final de máster, es el de extender y mejorar el método que se había propuesto. En la propuesta original del método se había profundizado bastante en la actividad de modelado de objetivos, mientras que el resto de actividades meramente se habían propuesto ideas de cómo deberían realizarse. En este trabajo se ha profundizado más en el resto de actividades así como también en el método general tratando de mejorarlo.

Se ha mejorado la actividad de definición de modelado de valor (ver sección 6), donde se ha **modificado la fórmula propuesta originalmente**, así como también se ha modificado el **orden de aplicación de la misma** y se ha razonado sobre el porqué de ello. La fórmula y cómo emplearla se explica más adelante en la sección correspondiente.

Se ha extendido la actividad de definición de procesos de negocio de alto nivel (ver sección 7), donde se ha **incluido el mapeo y las transformaciones entre el modelo de objetivos de Value@GRL y un modelo de procesos de negocio**. En este caso hemos seleccionado BPMN como modelo, debido a que es un muy utilizado hoy en día en la industria.

La fórmula que se había propuesto originalmente para este método es la mostrada a continuación:

$$Valor(E) = E_{importancia} + \sum_{i=0}^{Ec} (\frac{Ec_i}{100} * Valor(Eco_i)) + \frac{1}{E_h} * Ep_{importancia}$$

Mientras que la nueva fórmula propuesta es la siguiente:

$$Valor = IEimp + \sum_{i=0}^{n} \left(\frac{IEcontrib_i}{100} * Valor(IEimpDest_i) \right) + \sum_{i=0}^{n} (Valor(IEimpDest_i)) + \left(\frac{IEPadre}{\#IEHijos} \right)$$

Como puede observarse comparando ambas fórmulas, se han conservado algunas partes de la fórmula anterior, como la importancia propia de cada elemento, la propagación de la importancia de la contribución en base tanto de la importancia de la contribución como del valor del elemento que contribuye y la propagación de la importancia de los elementos padres a los elementos hijos. La parte que se ha incluido ha sido la de la propagación del valor de la dependencia, en la cual se ha considerado que si algo es dependiente contribuye en su totalidad, ya que es necesario. Todas las partes de la fórmula se explican con más detalle en la **sección 6.2** de este trabajo.

Además de todo esto, también se han creado guías para cada una de las actividades del método, donde se explica cómo debería modelarse con el lenguaje de modelado desarrollado Value@GRL, cómo debería asignarse la importancia a cada uno de los elementos intencionales, y cómo se propaga ésta para calcular el valor. Finalmente, se definen los mapeos de correspondencia entre los elementos de un modelo de valor y los elementos de un modelo de procesos de negocio.

5. Modelado de objetivos con Value@GRL

El principal objetivo del modelado del valor es explicar los principales objetivos que contribuyen a la organización, cómo estos objetivos interactúan entre sí, qué tareas en los límites del sistema de información deben realizarse y qué dependencias de servicios externos serán necesarias para cumplir los objetivos. De manera que sea posible realizar la priorización de los requisitos a entregar utilizando el criterio de valor, identificando los elementos más valiosos (de mayor contribución al cliente) y dividir el sistema para su entrega en iteraciones.

El diagrama de Value@GRL ha sido una modificación del diagrama de las primeras versiones del diagrama de GRL, pero delimitando los elementos existentes, así como añadiendo la posibilidad de añadir el valor a cada uno de los elementos.

Para mostrar los distintos elementos que pueden utilizarse en Value@GRL vamos a hacerlo mediante un ejemplo de cómo se debería utilizar e iremos explicando los distintos elementos que aparecen.

5.1. Identificación de actores

La primera actividad que hay que realizar a la hora de hacer un modelo de objetivos, es la de identificar los actores e indicar de qué tipo son.

Un actor es un elemento que representa una entidad activa (un *stakeholder*, sistema u otro) que tiene intenciones y lleva a cabo acciones para conseguir sus objetivos ejerciendo su know-how. Un actor, posee un conjunto de valores que representan a su organización. Estos valores pueden descomponerse como un conjunto de objetivos cuyo cumplimiento representa el grado de eficacia de la organización. La frontera de un actor modela aquellos elementos intencionales que se encuentra, deben cumplirse o realizarse dentro de la organización. Existen tres tipos de actores:

- Actor principal: Es aquel actor que representa el *stakeholder* en el que nos enfocamos (puede haber más de uno).
- Actor sistema: Es aquel actor que representa el sistema que queremos desarrollar.
- Actor externo: Es aquel actor que se relaciona con el actor sistema, pero que no es un actor principal.

Como ejemplo para mostrar los distintos tipos de actores, vamos a poner una tienda de ordenadores (**PcComponentes**) que será el sistema que deseamos desarrollar, por lo que será nuestro actor sistema. Un **cliente** que será el que comprará en nuestra tienda y es el *stakeholder* en el que nos vamos a enfocar, por lo que será el actor principal. Y por último un sistema de envío de paquetes (**Seur**) que permitirá a nuestro actor sistema enviar los ordenadores que vende. La representación de estos actores y su tipo pueden verse en la **Figura *9***.

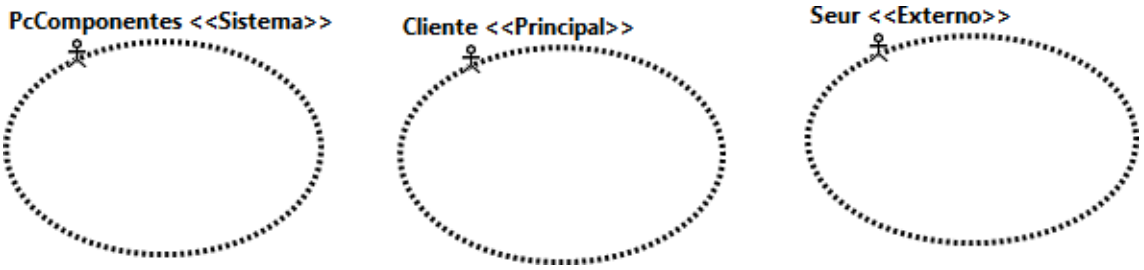


Figura 9: *Representación de los actores en el ejemplo.*

5.2. Modelado de los elementos intencionales

La segunda actividad para la realización de un modelo de objetivos es la de modelar los elementos intencionales de los actores principal y externo.

Los elementos intencionales describen una intención. Su uso en el modelo del valor permite explicitar los elementos que pueden contribuir al valor de la organización (tanto de forma positiva como negativa). Son aquellos objetivos o tareas relevantes desde el punto de vista del valor de la organización, los actores externos y de la intencionalidad del sistema. Los distintos tipos de elementos intencionales que existen pueden verse en la **Figura *10***.

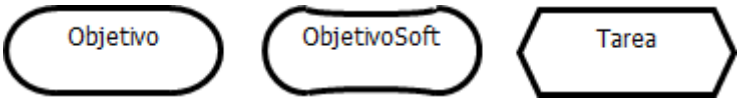


Figura 10: *Representación de los distintos tipos de elementos intencionales.*

5.2.1. Objetivos

Un objetivo es una condición o estado del mundo que un actor querría conseguir. Cómo se consigue el objetivo no está especificado por el objetivo mismo. Un objetivo puede ser tanto un objetivo de negocio como uno del sistema o una aserción deseable para el actor, siempre que estos sean relevantes para el valor del actor. Un objetivo de negocio expresa objetivos de acuerdo con el estado de los asuntos de negocios que un individuo u organización querrían conseguir. Un objetivo de sistema expresa una meta que el sistema objetivo debe conseguir y generalmente describe objetivos funcionales del sistema de información

objetivo (cuyos requisitos queremos capturar). Un objetivo conseguido contribuye de forma positiva al valor de su actor mientras que uno no conseguido puede reducir el valor del mismo.

Siguiendo con el ejemplo que hemos propuesto, nuestro actor principal desea (tiene como objetivo) **comprar un ordenador**, mientras que nuestro actor externo ofrece un servicio de **transporte de productos**, en la **Figura *11*** mostramos cómo debería modelarse en Value@GRL.

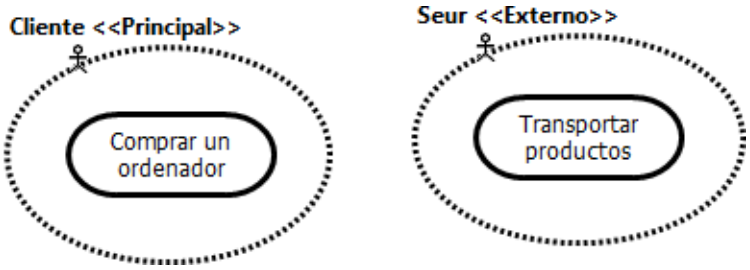


Figura 11: *Representación de los objetivos en el ejemplo.*

5.2.2. ObjetivosSoft (soft-goals)

Un objetivoSoft es una condición o estado del mundo que un actor querría conseguir pero que, a diferencia del concepto de objetivo, no existe un criterio que determine de forma exacta si la aserción del objetivo se ha cumplido y es responsabilidad de un juicio subjetivo y la interpretación del modelador juzgar si un particular estado de las cosas en efecto consigue suficientemente el objetivoSoft definido. Este concepto se suele aplicar a cualidades y aspectos no funcionales como la seguridad, la robustez, el rendimiento o la usabilidad (entre muchos otros) que poseen una escala cuantitativa de cumplimiento a la que es necesario establecer límites para valorar su cumplimiento o no. De igual manera, el cumplimiento de un objetivoSoft incrementa el valor de un actor.

En nuestro caso, el actor principal desea que cuando pida un ordenador le **llegue rápidamente**, así como también desea que el sistema sea **seguro**, ya que va a realizar compras con él, esto puede modelarse como se muestra a continuación en la **Figura *12***.

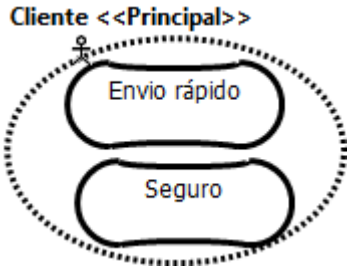


Figura 12: *Representación de los objetivosSoft en el ejemplo.*

5.2.3. Tareas (Tasks)

Una tarea específica una forma particular de hacer algo. Cuando una tarea es parte de una descomposición de otra de mayor nivel, restringe esa tarea de nivel más alto a un curso de acción particular. Las tareas representan las formas de cumplimiento de los objetivos. Una tarea debe contribuir con al menos un objetivo (hard o soft). Una tarea está relacionada en un mayor nivel de abstracción, con una actividad o conjunto de actividades de proceso de negocio que estarán soportadas por el sistema de información.

Las acciones (tareas) que nuestro actor principal deberá realizar para poder comprar su ordenador son la de **proveer la información** de lo que desea comprar, así como la dirección de envío y también la de **pagar** el producto, en la **Figura *13*** mostramos como debería modelarse.

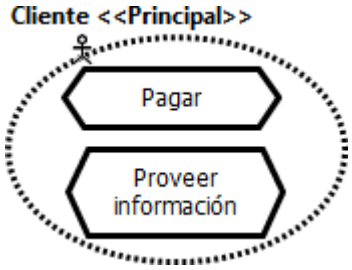


Figura 13: *Representación de las tareas en el ejemplo.*

Como resultado general del modelado de los elementos intencionales para los actores principal y externo es el que mostramos en la **Figura 14**.

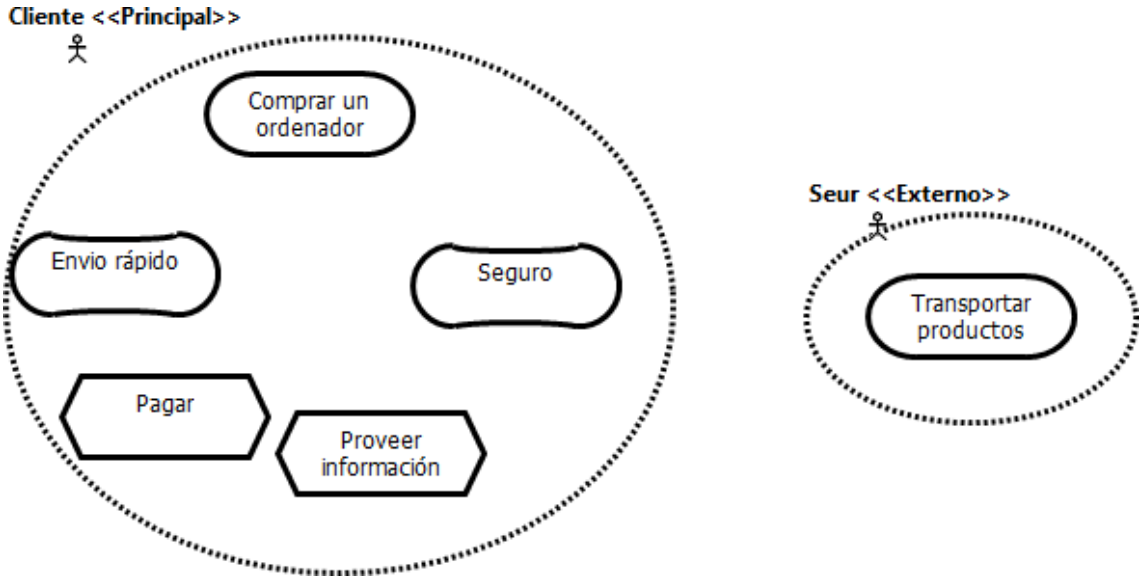


Figura 14: *Representación del modelado de elementos intencionales del ejemplo.*

5.3. Modelado de los enlaces internos

La tercera actividad para realizar un modelado de valor es la de modelar las relaciones existentes entre los elementos intencionales de un mismo actor.

Existen dos tipos de enlaces internos: descomposición y contribución.

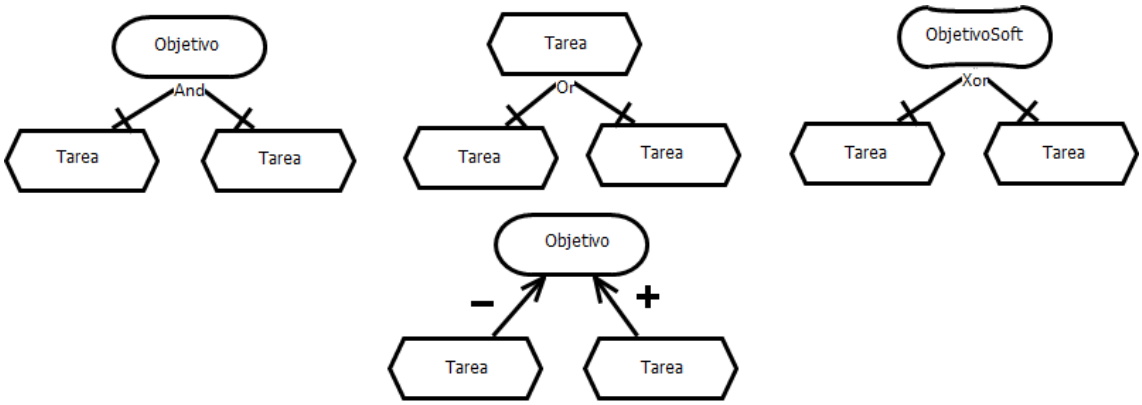


Figura 15: *Representación de los distintos tipos de enlaces.*

En la **Figura 15** se muestran los distintos tipos de enlaces internos existentes, siendo los tres enlaces de la parte superior **descomposición** de tipo **And**, **Or**, **Xor** y el enlace de la parte inferior un enlace de **contribución**.

5.3.1. Descomposición

En ocasiones un elemento intencional (tareas, objetivos y objetivos flexibles) puede ser descompuesto en unidades más pequeñas, para detallar en más profundidad un elemento relevante para el cliente.

Las descomposiciones pueden ser de tres tipos distintos, donde cada uno representa la cantidad de elementos intencionales que pueden seleccionarse, los posibles tipos son los siguientes:

- AND: Tipo de descomposición donde todos los elementos hijos necesitan son necesarios para la realización de la tarea padre.
- OR: Tipo de descomposición donde se muestran las distintas alternativas para la realización de la tarea padre. Cabe mencionar que estas alternativas **no son excluyentes**, y pueden seleccionarse, tantas como se desee, así como también ninguna de ellas.
- XOR: Tipo de descomposición donde únicamente hay que seleccionar una de las posibles alternativas, es decir, de todas las posibles alternativas solo pueden seleccionarse una, debido a que es **excluyente**.

5.3.2. Contribución (interna)

Un enlace de contribución une un elemento intencional con otro sobre el que tiene un efecto (positivo o negativo). Un enlace de contribución puede unir objetivos, objetivosSoft y tareas. El uso de los enlaces de contribución debe tratar de representar cualquier efecto que los elementos intencionales tengan entre sí que pueda condicionar la selección de requisitos del sistema o que afecten de alguna manera al valor del cliente. Estos efectos tendrán un papel decisivo a la hora de priorizar requisitos.

Para nuestro ejemplo, existe una **descomposición** de tipo AND con las tareas del mismo actor, representando que para poder realizar el objetivo (comprar un ordenador) es necesario previamente proveer la información del pedido y pagarlo, así como también existen dos **contribuciones** positivas, que contribuyen al usuario a la hora de comprar un ordenador, dando como resultado lo mostrado en la **Figura 16**.

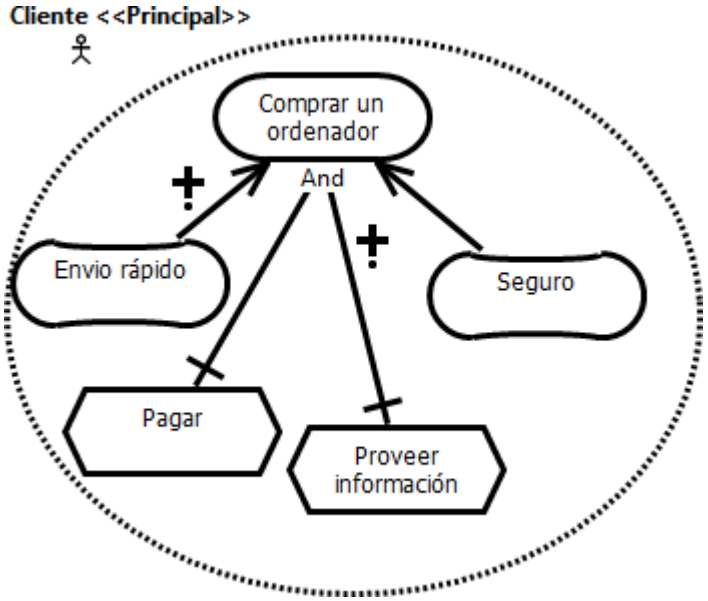


Figura 16: *Representación de los enlaces entre los distintos elementos intencionales del actor principal.*

5.4. Modelado del actor sistema y sus enlaces internos

La cuarta actividad es la de modelar los elementos intencionales del actor sistema y de las relaciones existentes entre los mismos. Continuando con nuestro ejemplo, nuestro actor sistema (PcComponentes) desea **vender ordenadores**, para lo cual necesita por un lado **obtener la información de la venta** que realiza, **cobrar** al cliente **y enviar** el pedido, lo cual además de modelarse como tareas, debería modelarse como una descomposición del objetivo, ya que estas tareas son necesarias para lograrlo. Además, para nuestro sistema es muy importante la **satisfacción del cliente**, la cual puede contribuir a que se vendan más ordenadores. En la **Figura 17** mostramos el resultado de este modelado.

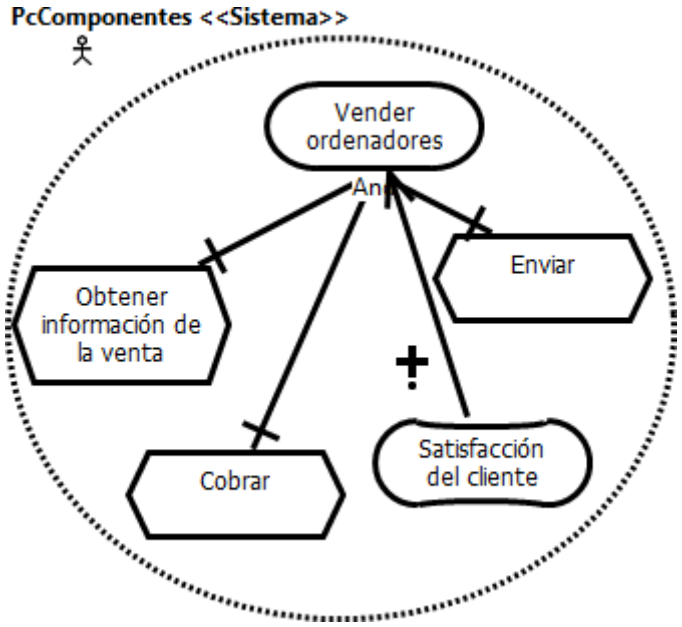


Figura 17: *Elementos intencionales del actor sistema y sus enlaces.*

5.5. Modelado de los enlaces externos

La última actividad para realizar el modelado de valor consiste en modelar las relaciones existentes entre los distintos actores con el actor sistema.

Para poder modelar las relaciones entre los distintos actores, se utilizan los enlaces de dependencia y los enlaces de contribución (que hemos explicado anteriormente).

Existen dos tipos de enlaces externos: la contribución, la cual hemos explicado previamente y la dependencia.

5.5.1. Dependencia

Una dependencia conecta dos actores a un elemento intencional. Describe cómo un actor depende en otro para cumplir el elemento intencional que los une. En el caso del dominio de software en la nube esta relación puede modelar el consumo de servicios de un tercero. Un ejemplo de esto es cuando una tarea necesita de una información que provee otra tarea, siendo así una dependencia de una tarea a la otra.

Como ejemplo, podemos modelar una dependencia, entre el actor sistema (PcComponentes) y el actor externo (Seur), donde para poder enviar una venta, el actor sistema depende del servicio de transporte de productos que ofrece el actor externo. El modelo resultante de esto puede verse en la **Figura 18**.

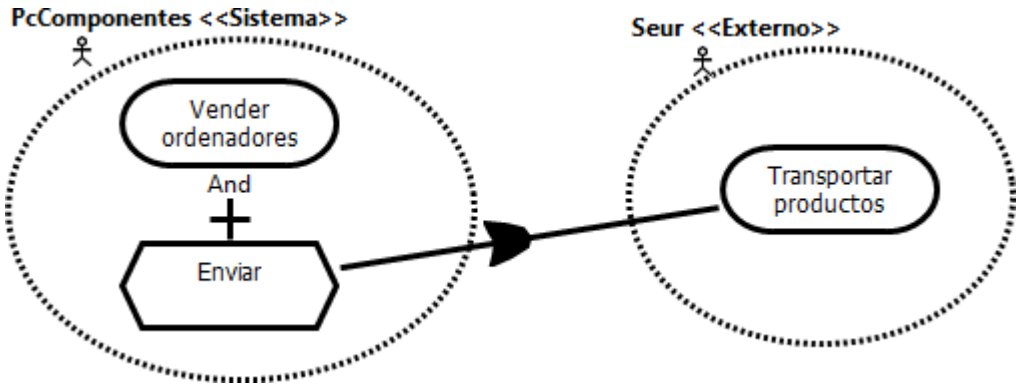


Figura 18: *Representación de una dependencia entre el actor sistema y el actor externo.*

5.5.2. Contribución (externa)

Como hemos dicho anteriormente, la **contribución** puede ser utilizada tanto como un enlace interno como uno externo ya que una tarea de un actor puede contribuir a otra tarea de otro actor. Un ejemplo de esto podría ser que el **envío rápido** que le interesa al cliente contribuye a lograr la **satisfacción del cliente** que desea la empresa, esto puede modelarse como puede verse en la **Figura 19**.

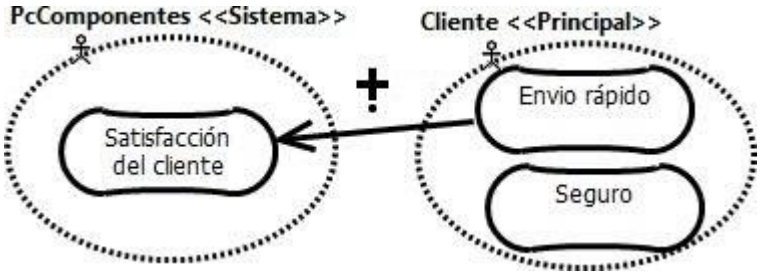


Figura 19: *Representación de una contribución entre el actor principal y el actor sistema.*

El resultado del modelado de objetivos de todo el ejemplo propuesto en esta sección puede verse en la **Figura *20***, donde aparecen todos los actores, tareas y relaciones que hemos nombrado durante toda la sección.

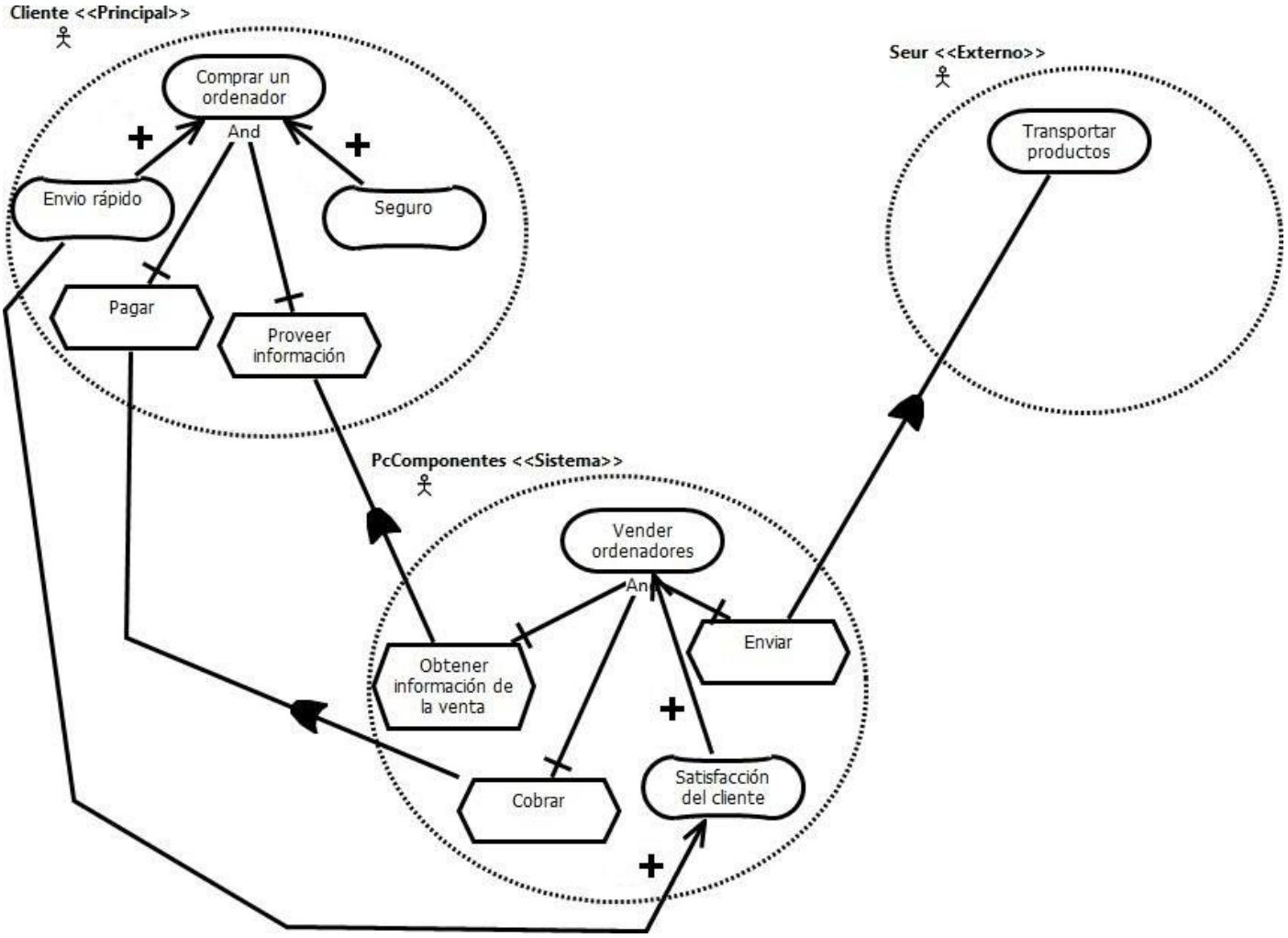


Figura 20: *Modelo de Value@GRL con todos los elementos y relaciones.*

6. Definición del modelo de valor

6.1. Asignación de importancia

Antes de poder comenzar a propagar la importancia de los elementos para poder calcular su valor es necesario que previamente se valore la importancia de cada uno de los elementos del modelo de objetivos. Para ello se debería realizar una reunión con los distintos actores para que dieran la importancia según su criterio a los elementos intencionales correspondientes a su actor.

A la hora de asignar la importancia a los distintos elementos, hemos creado una escala intervalo con cuatros posibles valores:

- 0: Insignificante
- 25: Poco importante
- 50: Importante
- 75: Muy importante
- 100: Imprescindible

En esta escala, los diferentes intervalos se han dividido de forma que haya la misma distancia entre todos los elementos, además de la posibilidad de asignar la importancia de algo como imprescindible o insignificante (todo o nada). También se ha elegido un número impar de intervalos para que sea más sencillo a la hora de elegir la importancia ya que existe un intervalo "medio" entre poco importante y muy importante.

En un principio la escala tenía muchos más intervalos, lo que causaba que fuera difícil asignar la importancia a los distintos elementos por lo que tuvimos que reducirla dando como resultado la mostrada anteriormente.

A parte de asignar la importancia a los distintos elementos intencionales del modelo también es necesario valorar la importancia de las relaciones de contribución que aparecen en el modelo ya que así se indica cuanto contribuye un elemento intencional en el otro elemento intencional, se utiliza la misma escala que para los elementos intencionales.

El resultado de asignar la importancia a todos los elementos intencionales del ejemplo es el mostrado en la ***Figura 21***.

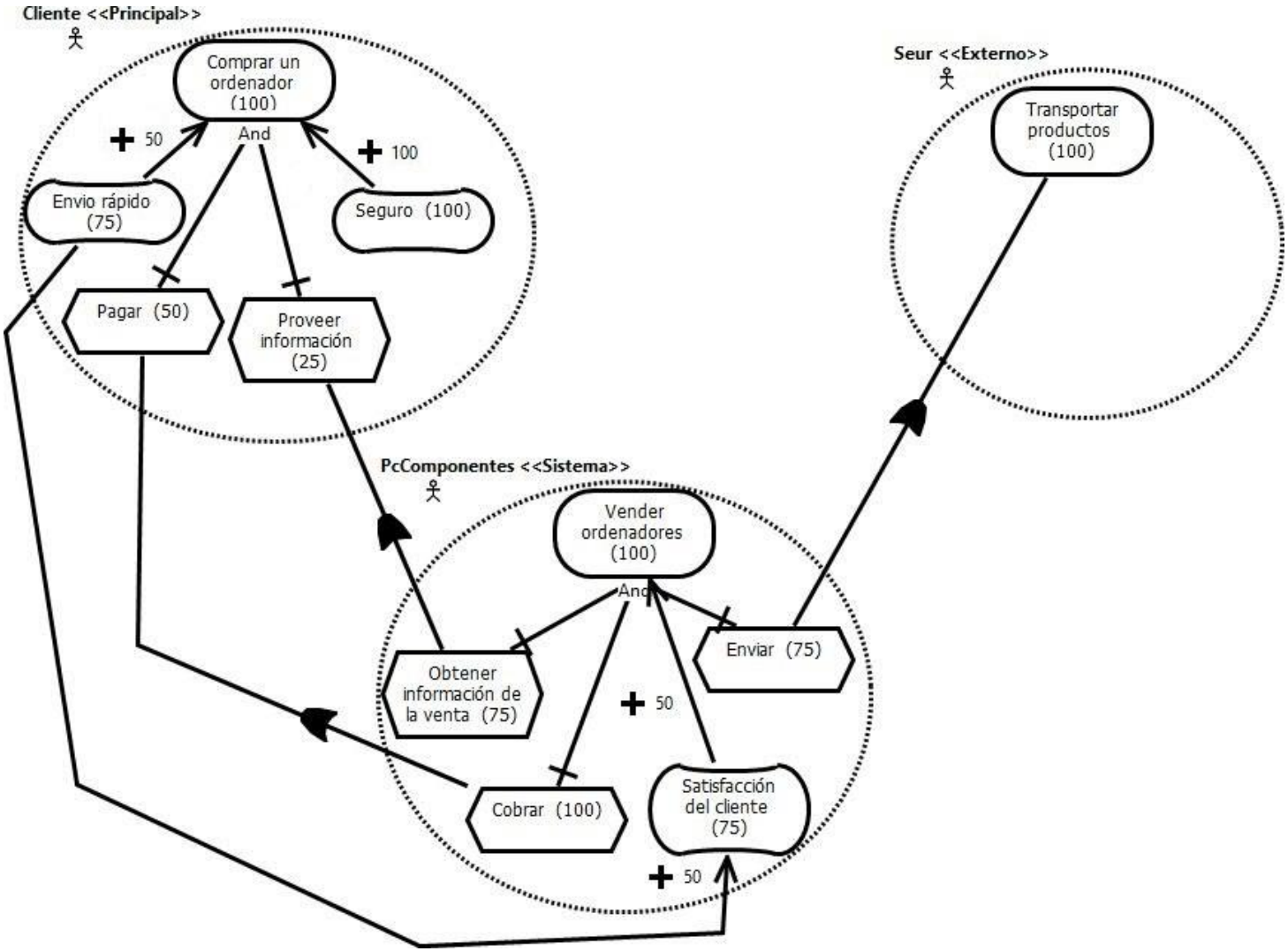


Figura 21: *Modelo de objetivos con la importancia asignada.*

6.2. Propagación de importancia

Uno de los principales elementos que hemos cambiado para mejorar la priorización de este método ha sido la fórmula que se aplica para la realización de la propagación del valor, siendo esta nueva fórmula la que puede verse a continuación.

$$Valor = IEimp + \sum_{i=0}^{n}\left(\frac{IEcontrib_i}{100} * Valor(IEimpDest_i)\right) + \sum_{i=0}^{n}(Valor(IEimpDest_i)) + \left(\frac{IEPadre}{\#IEHijos}\right)$$

Siendo:

- **IEimp**: La importancia o valor propio de cada elemento.
- **IEcontrib**: El valor del enlace de contribución, pudiendo ser este 0, 25, 50, 75 o 100.
- **IEimpDest**: El valor del destino, es decir, es el valor que tiene el elemento con el cual está relacionado.
- **IEPadre**: Es el valor o importancia que tiene el elemento padre en una descomposición
- **#IEHijos**: Cantidad de elementos hijos que tiene una descomposición.

Una vez vista la fórmula vamos a explicarla paso por paso y porqué está hecha de esta manera.

$$IEimp$$

El primer elemento que encontramos en la fórmula es el IEimp, que es la importancia que tiene un elemento. En un primer intento de la fórmula habíamos hecho que los elementos del actor sistema no tuvieran importancia y que toda la importancia que tuvieran viniese de las relaciones con el resto de actores, pero tras varias pruebas, nos dimos cuenta de que podía darse el caso de que, si un elemento carecía de importancia, este podría llegar a tener un valor de 0 al aplicar la fórmula, así que añadimos EImp para ponerle un valor.

$$\sum_{i=0}^{n}\left(\frac{IEcontrib_i}{100} * Valor(IEimpDest_i)\right)$$

El siguiente elemento que podemos encontrar en la fórmula es el sumatorio de todas las contribuciones que recibe un elemento (siendo n el número de contribuciones que tiene ese elemento), teniendo en cuenta tanto el valor del elemento (EImpDest) como el valor que tenga la contribución (IEcontrib). En cuanto a la división que hay de IEcontrib/100 es para calcular su peso, dando los posibles valores de 1, ¾, ½ y ¼ correspondiendo con el valor del enlace de la contribución de 100, 75, 50 o 25. Una vez calculado el peso de la contribución se multiplica por el valor del elemento que contribuye. Cabe

destacar que para hacer el cálculo de las contribuciones no tenemos en cuenta la direccionalidad, esto es debido a que la dirección es muy subjetiva y puede depender del modelador. Así como por ejemplo podríamos decir que una buena usabilidad contribuye a la visualización de los datos, o que una buena visualización contribuye a la usabilidad.

$$\sum_{i=0}^{n}\left(Valor(IEimpDest_i)\right)$$

Este elemento de la fórmula tiene que ver con las dependencias, y básicamente lo que hace es sumar el valor de todas las dependencias que tienen. Siendo n el número total de dependencias e IEimpDest el valor que tiene el elemento con el que se relaciona. Mencionar que aquí tampoco tenemos en cuenta la direccionalidad de la dependencia.

$$\left(\frac{IEPadre}{\#IEHijos}\right)$$

El último elemento de la fórmula está relacionado con la descomposición que ofrece GRL, en nuestro caso hemos decidido que el valor que tiene el padre se reparta entre la cantidad de hijos que este tenga, siendo por ejemplo un padre con un valor de 100 y 4 hijos, cada uno de los hijos obtendrá 25 de valor del padre. Hemos realizado esta forma de calcular la descomposición debido a que estuvimos haciendo pruebas y nos dimos cuenta de que si el valor era calculado en base a los elementos hijos de la descomposición sucedía que el padre que tenía más hijos era el que mayor valor tenía, y esto podía darse, cosa que podía ocasionar problemas debido a la libertad que ofrece Value@GRL, donde un elemento puede descomponerse en tantos elementos hijos como se desee, pudiendo dar lugar a que un elemento tenga muchos elementos hijos y otro no tenga ninguno, dando como resultado que el elemento con muchos hijos tuviera un mayor valor, pese a que no sea importante.

En cuanto al orden de aplicación de la fórmula es muy semejante al que se había propuesto originalmente en la metodología. En un principio se aplica esta fórmula para todos los elementos de los actores que no sean el actor sistema, empezando siempre por los elementos padres para poder propagar adecuadamente su valor a sus elementos hijos, y luego de aplicar la fórmula a todos los elementos que no sean del actor sistema, se aplica la fórmula al actor sistema para propagar adecuadamente el valor que aportan el resto de actores. Cabe mencionar que lo hacemos de esta forma porque en nuestro modelo solo se ponen las relaciones entre el actor sistema y el resto de actores, es decir no existen relaciones entre varios tipos de actores, solo con el sistema.

El resultado de aplicar el cálculo del valor, sobre el ejemplo, da como resultado lo mostrado en la **Figura 22**.

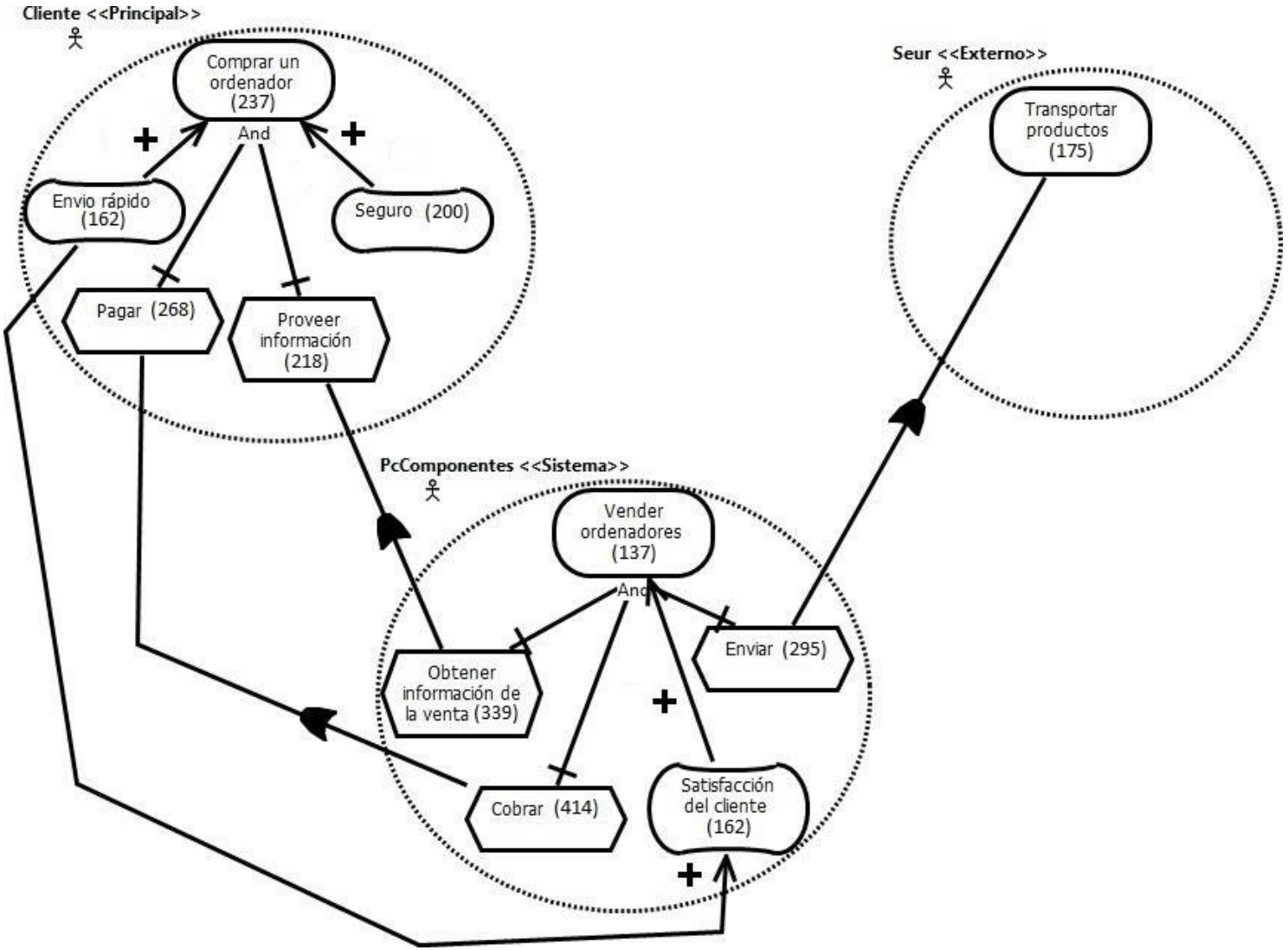


Figura 22: *Modelo de Value@GRL con el valor.*

7. Definición de procesos de negocio de alto nivel

7.1. Mapeo entre elementos de los modelos

Esta sección está orientada a un **mapeo conceptual** entre los distintos elementos de los metamodelos, utilizando ejemplos con modelos, y sin hacer referencia técnica a los metamodelos; en la sección siguiente explicamos las transformaciones definidas usando los modelos de una forma más técnica.

Antes que nada, mencionar que el lenguaje de modelado de Value@GRL ofrece la posibilidad de **modelar de múltiples formas distintas** las mismas cosas, y que es muy **subjetivo**, dependiendo por tanto de quien modele, por lo que en esta memoria presentamos los posibles mapeos y cuál de ellos hemos seleccionado y por qué.

En esta sección únicamente vamos a presentar los mapeos de los elementos más importantes. Para la explicación de los diferentes mapeos, lo que hemos hecho ha sido clasificar los elementos intencionales entre si tienen o no una descomposición y de que tipo es.

El mapeo de un **actor** corresponde con una calle en BPMN, como se muestra en la **Figura 23**.

Figura 23: *Mapeo de un actor a una calle.*

El mapeo de un **objetivo** de Value@GRL corresponde a una **tarea** en BPMN o a un **subproceso**, dependiendo de lo que se haya querido modelar con ello. Debido a que en el modelo de Value@GRL se carece de la expresividad necesaria para seleccionar en qué debería transformarse, hemos decidido que si el **objetivo** se encuentra en **un actor externo** este se transformará en una **tarea**, mientras que si el **objetivo** se encuentra en el **actor sistema o** en el **actor principal** en un **subproceso**.

Hemos decidido esto porque normalmente los actores externos se utilizan para realizar pequeñas acciones dando o recibiendo información, mientras que en el actor principal o actor sistema es lo que se va a implementar y es necesario aumentar su semántica a posteriori mediante la utilización de BPMN. Además, mencionar que la herramienta que utilizamos permite muy fácilmente

transformar una tarea en un subproceso en el caso de que le interese al modelador del dominio. En la **Figura 24** se muestran las posibles transformaciones de un objetivo.

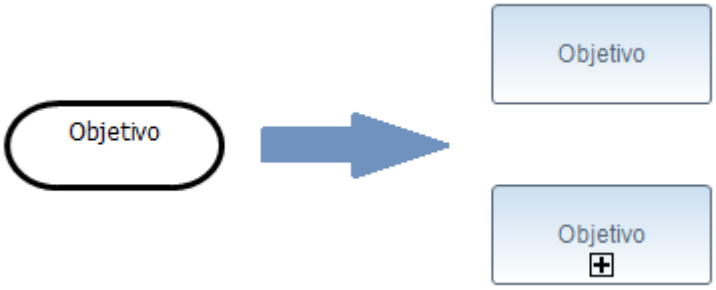


Figura 24: *Posibles mapeos de un objetivo.*

El mapeo de una **tarea** de Value@GRL puede corresponder con una **tarea** en BPMN o con un **subproceso**, dependiendo de la abstracción con la que se haya modelado la tarea, o si esta necesita de más especificación. Para este mapeo hemos decidido que la **tarea** se transforme en una **tarea** y que el modelador si así lo desea transforme la tarea en un subproceso, hemos decidido hacerlo de esta manera porque existe la descomposición en Value@GRL si se desea añadir información sobre qué otras tareas componen otra tarea. En la **Figura 25** se muestran las posibles transformaciones que tiene una tarea.

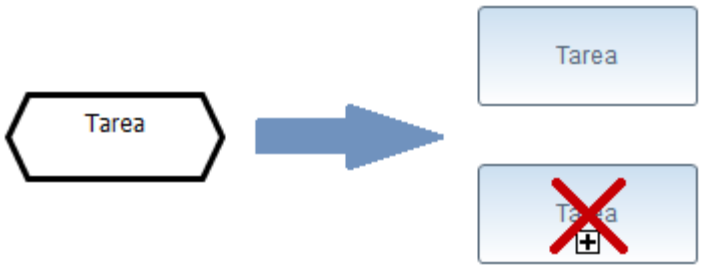


Figura 25: *Posibles mapeos de una tarea.*

El mapeo de un **objetivo con descomposición** de Value@GRL puede corresponder con una **tarea** o con un **subproceso**, atendiendo a la abstracción que tenga el modelo, dependiendo de la subjetividad del modelador. Debido a que no podemos valorar la abstracción de los objetivos, hemos decidido que un **objetivo** se mapee en un **subproceso** (Ver **Figura 26**).

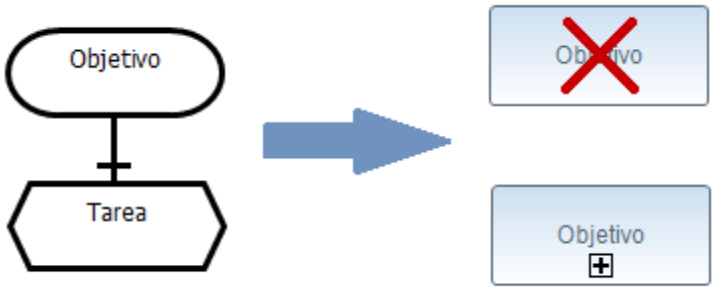


Figura 26: *Posibles mapeos de un objetivo con descomposición.*

El mapeo de una **tarea con descomposición** de Value@GRL corresponde a un **subproceso**, esto es debido a que las tareas generalmente se modelan a un nivel alto, y si tienen elementos hijos es para especificar la tarea con más detalle. En la **Figura 27** se muestra este mapeo.

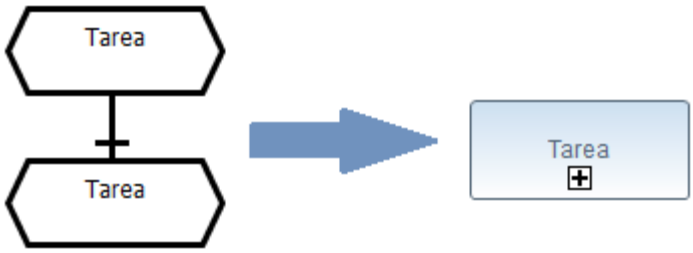


Figura 27: *Mapeo de una tarea con descomposición.*

Para el mapeo entre los distintos elementos no solo hemos tenido en cuenta si los elementos intencionales tienen o no descomposición, sino también el tipo de descomposición que tiene, ya que cada tipo de descomposición tienen un significado distinto y debería representarse de forma distinta en BPMN. Antes que nada, mencionar que si hay una descomposición no tiene porqué corresponder

Así pues, si la **descomposición** es de tipo **AND** y el mapeo del elemento padre es a un subproceso de acuerdo con los mapeos anteriormente propuestos, los elementos hijos de este se transformarán de forma recursiva aplicando los mapeos anteriores, pudiendo dar lugar así a que un subproceso esté compuesto por tareas y por subprocesos. Cabe destacar que, debido a la carencia de orden de precedencia en el modelo de objetivos, no es posible que este mapeo genere un flujo donde las tareas se realicen en un orden concreto.

En la **Figura 28** se muestra un ejemplo de una transformación de una descomposición de tipo **AND**, donde puede observarse la carencia de flujo.

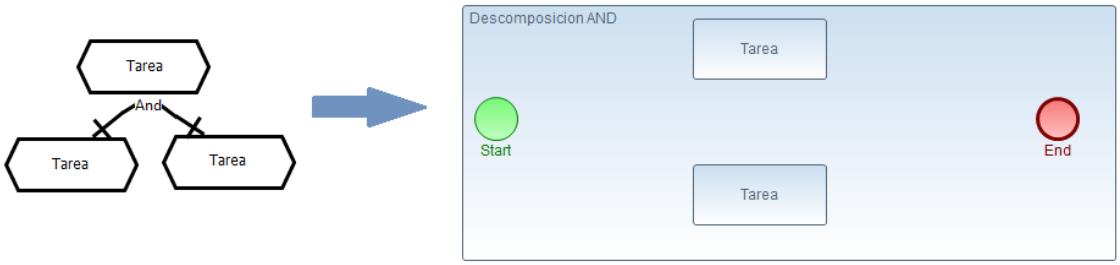


Figura 28: *Mapeo de una descomposición de tipo **AND**.*

Una **descomposición** de tipo **OR** tiene un mapeo distinto a la de tipo AND, debido a la semántica que tiene, ya que una descomposición de tipo OR representa posibles alternativas para realizar algo incluyendo no implementar ninguna alternativa. Ya que este tipo de descomposición aporta una semántica de alternativas es posible generar un flujo, dando lugar así al mapeo de la imagen de a continuación.

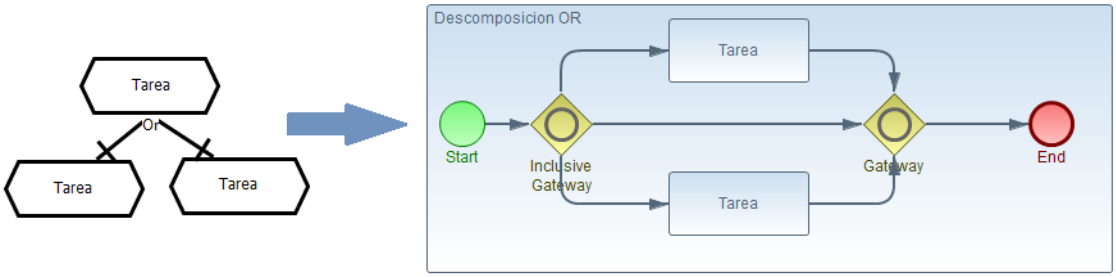


Figura 29: *Mapeo de una descomposición de tipo **OR**.*

Como puede observarse en la **Figura 29**, la diferencia con la descomposición de tipo AND es la de incluir el flujo, así como también puertas lógicas que permiten la selección de ninguna, una, varias o todas las alternativas.

Una **descomposición** de tipo **XOR** al igual que la de tipo **OR** tiene una semántica propia donde se representan las posibles alternativas para realizar algo, pero donde solo puede seleccionarse una de todas las posibles alternativas. Ya que este tipo de descomposición aporta una semántica de alternativas es posible generar un flujo, dando lugar así al mapeo de la **Figura 30**.

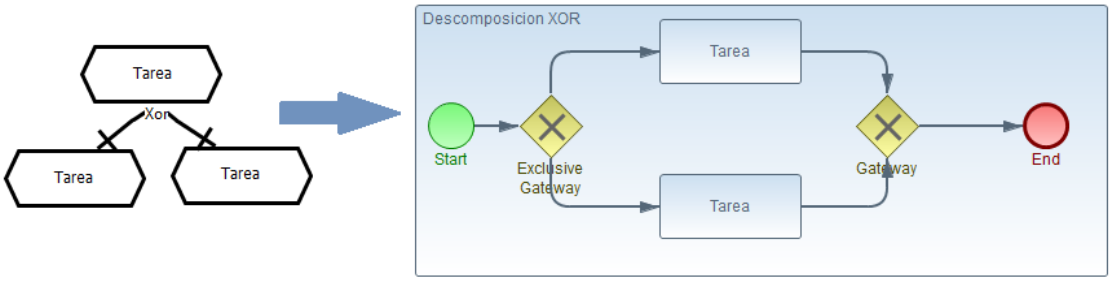


Figura 30: *Mapeo de una descomposición de tipo **XOR**.*

Al igual que la descomposición de tipo OR, con la de tipo XOR es posible añadir un flujo. La diferencia entre estos dos tipos de descomposiciones son las puertas utilizadas, mientras la de tipo OR se mapea a una puerta inclusiva, la de tipo XOR se mapea a una puerta exclusiva. Y cómo funciona cada una de estas puertas, ya que la inclusiva permite seleccionar todas, ninguna o solo las que interese, mientras que la puerta exclusiva obliga a seleccionar una de todas las posibles opciones.

El mapeo de una **dependencia** entre dos elementos intencionales corresponde con una comunicación entre actores mediante la utilización de mensajes. Ya que la dependencia habla de la necesidad de una tarea sobre la otra, es posible modelar el flujo y el orden en el cual deberían hacerse las tareas. En la **Figura 31** mostramos cómo debería ser esta transformación, las calles modeladas no pertenecen a esta transformación, pero la hemos puesto porque es importante para el contexto, para mostrar la comunicación entre actores.

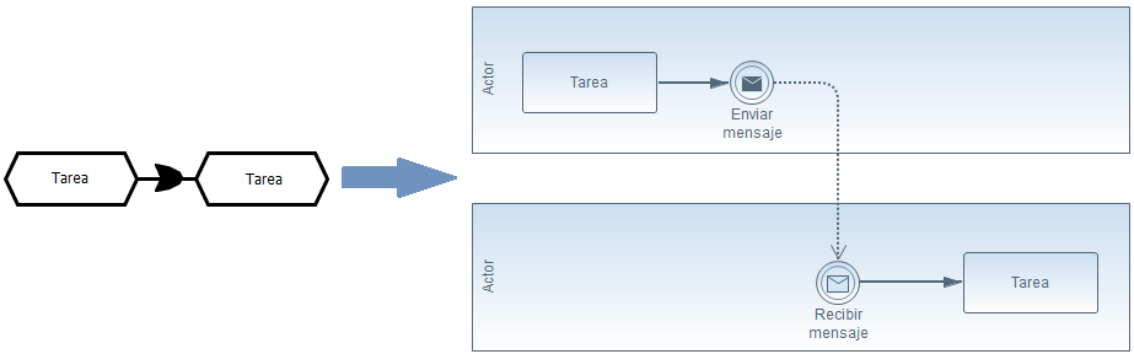


Figura 31: *Mapeo de una dependencia.*

En cuanto al mapeo de los objetivoSoft, no se ha realizado ninguno al modelo de procesos de negocio, ya que un objetivoSoft normalmente está relacionado con un requisito no funcional y por tanto no puede representarse su funcionalidad. La única opción que podría realizarse sería la de mapear el objetivoSoft a una anotación en el modelo de procesos de negocio, cosa que no es necesaria, ya que está en el modelo de objetivos, y por lo tanto aparecerá más tarde en la lista priorizada.

Con el mapeo de una contribución sucede de forma semejante, una contribución puede utilizarse para enlazar tanto un par de tareas, donde una tarea contribuye a la otra como una tarea que ayuda a obtener un objetivoSoft. Debido a lo cual, no se ha definido un mapeo concreto para la contribución.

En la **Figura *32*** mostramos cual sería el resultado de aplicar este mapeo conceptual al ejemplo que estamos utilizando a lo largo de esta memoria.

Una vez realizado este primer mapeo, un experto en el dominio deberá modificarlo, para añadirle información sobre precedencia, y las actividades faltantes, si fuese necesario. Un ejemplo de esto puede verse en la **Figura 33**.

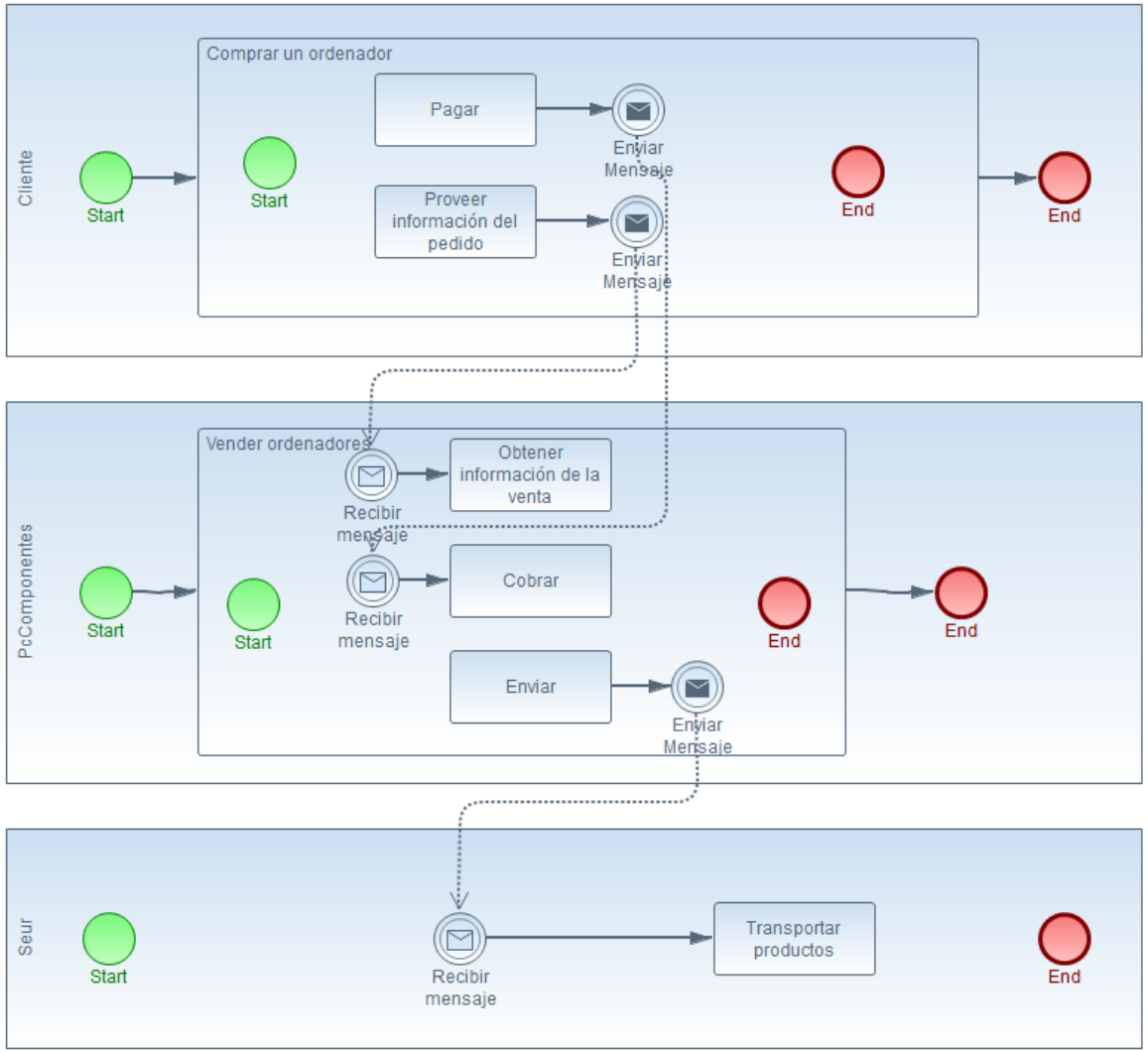


Figura 32: *Modelo BPMN generado realizando el mapeo.*

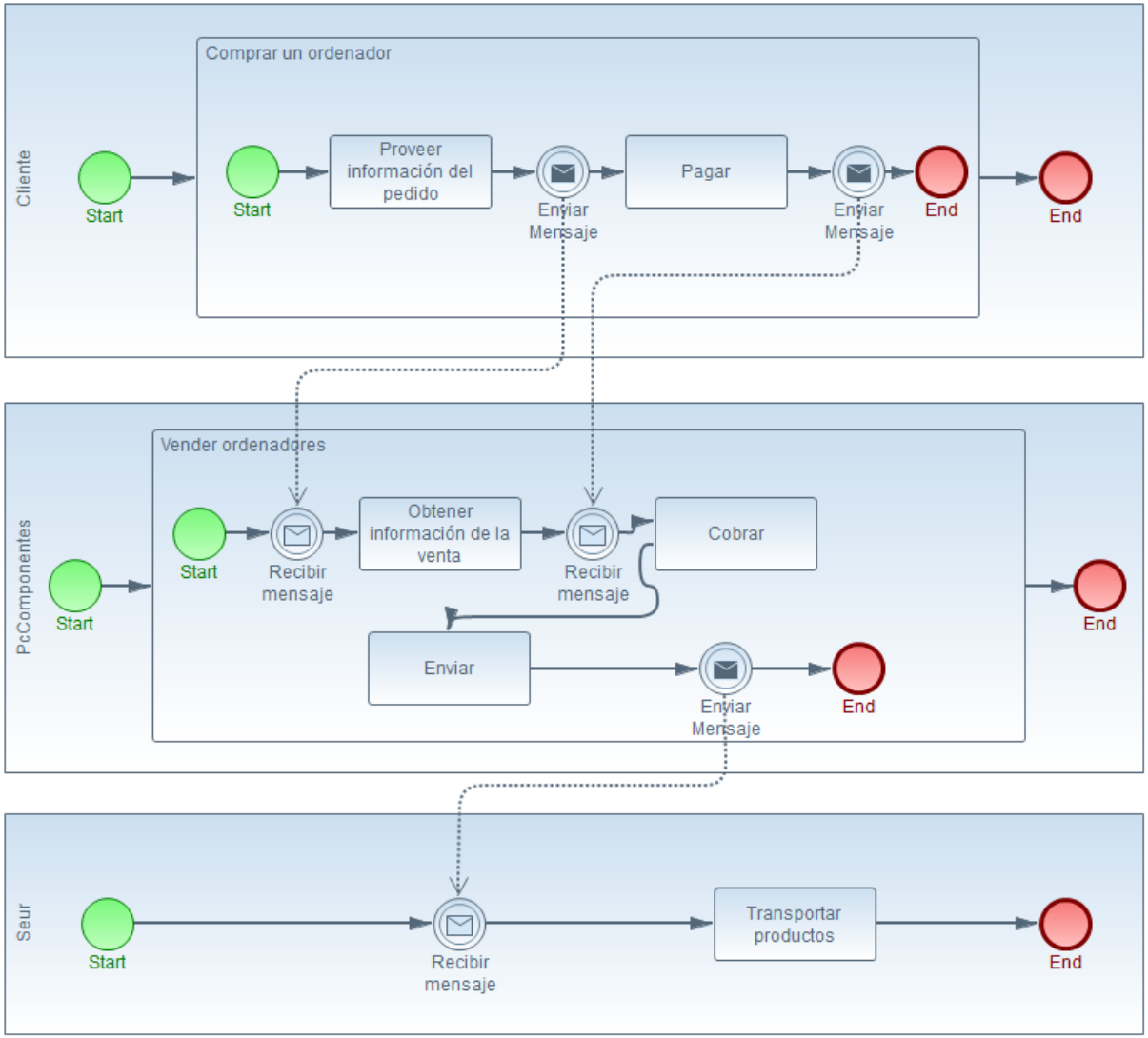


Figura 33: *Modelo BPMN modificado*

7.2. Metamodelo Value@GRL

Basado en el *Core Overview Metamodel* de la especificación de GRL en el manual de URN (2011) [28]. Para ser específicos, se está utilizando un metamodelo basado en el estándar de GRL, pero al que se le ha incluido la posibilidad de añadir valor a los distintos elementos intencionales.

Los elementos **conceptuales** de este metamodelo (**Figura 34**) han sido explicados previamente.

- **GRLspec**: Elemento que representa el diagrama de valor en general
- **Actor**: Elemento que representa una entidad activa (un *stakeholder*, sistema u otro) que tiene intenciones y lleva a cabo acciones para conseguir sus objetivos ejerciendo su know-how. Representan a *stakeholders* del sistema. Un actor puede contener elementos intencionales.
- **ActorType** (Tipo de actor): Tipos de contribución:
  - **Main:** Actor principal, sobre el que se enfoca el problema.
  - **System:** Actor que representa el sistema que se desea desarrollar.
  - **External:** Resto de actores o sistemas que interactúan con el sistema.
- **Intentional Element** (Elemento Intencional): Un elemento del modelo GRL que describe una intención, puede ser incluido en la definición de un actor, relacionarse con otros elementos intencionales con diferentes tipos de enlaces (del tipo *ElementLink*) y poseer un valor de importancia. El atributo indica la importancia de este elemento con respecto al actor que le contiene, por defecto está inicializada a 50.
  - **Goal** (Objetivo)**:** Un objetivo es una condición o estado del mundo que un actor querría conseguir. Cómo se consigue el objetivo no está especificado por el objetivo mismo.
  - **Sof goal** (ObjetivoSoft): Un objetivoSoft es una condición o estado del mundo que un actor querría conseguir pero que, a diferencia del concepto de objetivo, no existe un criterio que determine de forma exacta si la aserción del objetivo se ha cumplido y es responsabilidad de un juicio subjetivo y la interpretación del modelador juzgar si un particular estado de las cosas en efecto consigue suficientemente el objetivoSoft definido.
  - **Task** (Tarea): Una tarea específica una forma particular de hacer algo.

- **GRLLinkableElement:** Elemento que indica que elementos pueden enlazarse, mediante *ElementLink*.
- **ElementLink** (Enlace de elementos): Conecta dos elementos intencionales y representa la relación intencional que existe entre ellos. *ElementLink* abstrae el comportamiento común del enlace de descomposición, el enlace de contribución y el enlace de dependencia. Un *ElementLink* tiene un elemento intencional fuente (src) y un elemento intencional destino (dest), deben ser elementos intencionales distintos.
- **Dependency** (Dependencia): Una dependencia es un enlace que conecta a dos elementos intencionales de distintos actores o dos actores a un elemento intencional. Describe cómo un actor depende en otro para cumplir el elemento intencional que los une. Al menos uno de los elementos unidos por un enlace de dependencia debe estar contenido en la definición de un actor.
- **Decomposition** (Descomposición): Los enlaces de descomposición proveen la habilidad de definir qué elementos intencionales fuente necesitan ser satisfechos o estar disponibles para que el elemento intencional objetivo sea satisfecho. El tipo de descomposición viene indicado en el atributo *decompositionType*.
- **DecompositionType** (Tipo de descomposición): Un elemento intencional puede ser descompuesto en tres formas AND; XOR, IOR.
    - **AND:** La satisfacción de todos los elementos sub-intencionales es necesaria para cumplir el objetivo.
    - **XOR:** La satisfacción de uno y solo uno de los elementos sub-intencionales es necesario para conseguir el objetivo. Indican alternativas.
    - **OR:** La satisfacción de uno de los elementos sub-intencionales es suficiente para lograr el objetivo, pero varios elementos sub-intencionales pueden ser satisfechos.
- **Contribution** (Contribución): Un enlace de contribución une un elemento intencional con otro sobre el que tiene un efecto (positivo o negativo). El atributo *contribution* es el nivel cualitativo de contribución. Mientras que el atributo *quantitativeContribution* indica el nivel cuantitativo de la contribución, por defecto 0.
- **ContributionType** (Tipo de contribución): Tipos de contribución:
    - **Positive:** La contribución es positiva.
    - **Negative:** La contribución es negativa.
- **Importance:** Tipo de dato, donde se representan los distintos valores que pueden asignarse al atributo *importance* de los elementos y relaciones.

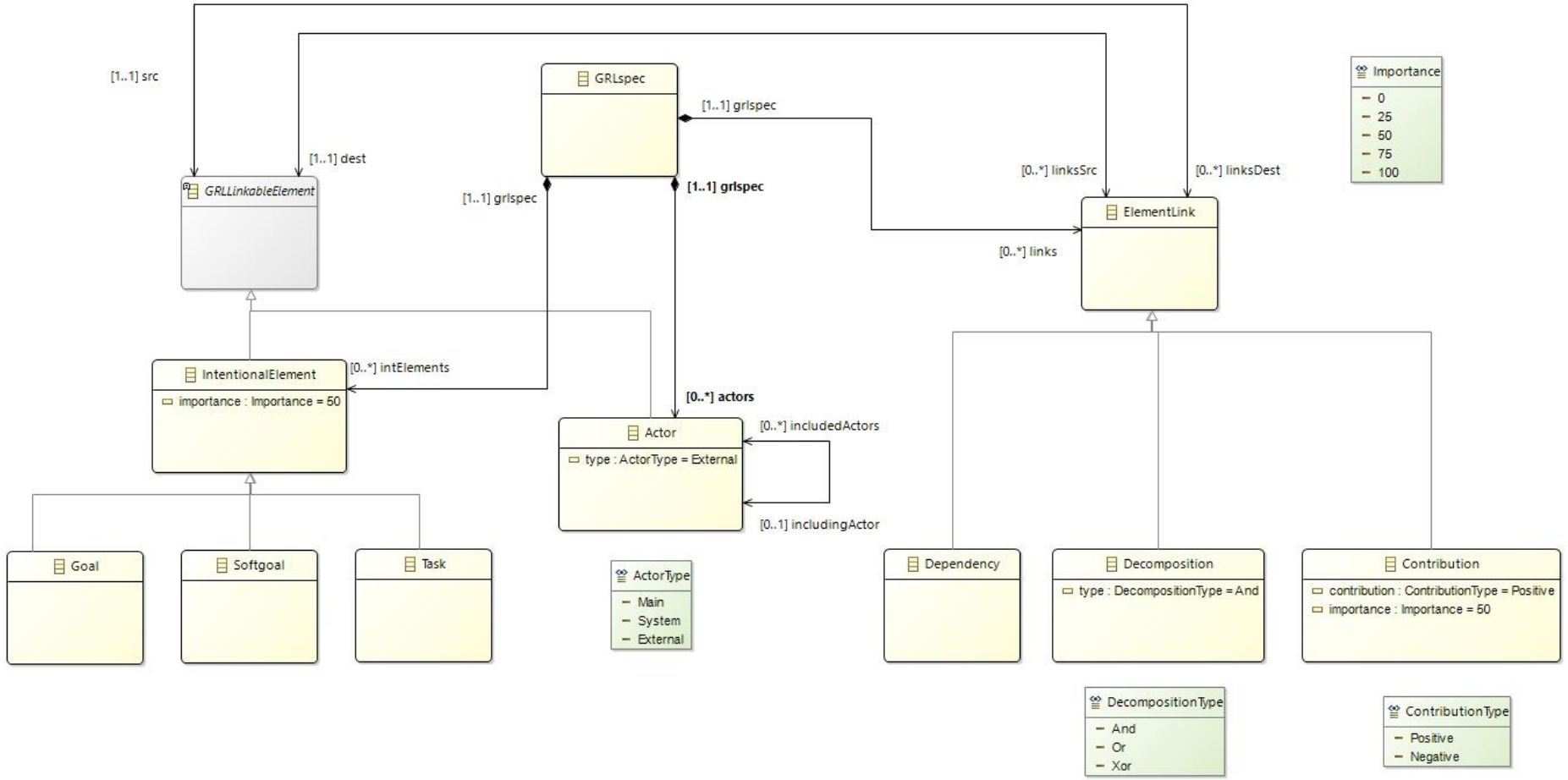


Figura 34: *Extracto del metamodelo GRL de URN.*

7.3. Metamodelo BPMN

Adaptado del modelo BPMN 2.0 implementado en Eclipse Modeling Framework por SAP y puesto a disposición de la comunidad en [67]. Por razones de espacio el tamaño del modelo ha sido reducido, suprimiendo los elementos más específicos y dejando aquellos básicos para un diagrama BPMN. No se han realizado cambios estructurales ni en las propiedades de los elementos que aparecen en el metamodelo.

Los distintos elementos de este metamodelo (**Figura 35**) son:

- **Base Element** (Elemento base): El elemento base es la súper clase abstracta de la mayoría de los elementos BPMN. Este elemento aporta el atributo id como identificación.
- **Definitions** (Definiciones): Este elemento se utiliza para dar información sobre el diagrama, como por ejemplo la versión del mismo.
- **Lane Set** (Piscina): Una piscina es un contenedor de calles, donde se almacenan las distintas calles de un mismo sistema.
- **Lane** (Calle): Una calle representa un actor, sistema o usuario concreto, donde todos sus elementos internos son realizados por este.
- **Flow Node:** Clase abstracta para todos los elementos que unen dos objetos en una relación de flujo.
- **Sequence Flow** (Flujo de secuencia): Se utiliza para indicar el orden de los objetos en un proceso. Cada flujo de secuencia tiene una fuente y un objetivo.
- **Event** (Evento): Algo que ocurre en el transcurso de un proceso. Los eventos afectan al flujo del proceso y normalmente tienen una causa o un impacto y en general requieren o permiten una reacción (el comienzo de una actividad, el final de una actividad, el cambio de estado en un documento, la llegada de un mensaje, etc.).
- **Throw Event** (Evento de lanzamiento)**:** Tipo de evento que al ser ejecutado lanza una reacción. Una vez activado los datos en su entrada se asignan automáticamente al mensaje, escala, error o señal referenciada por la definición del evento descrita en el atributo *EventDefinitinion*.

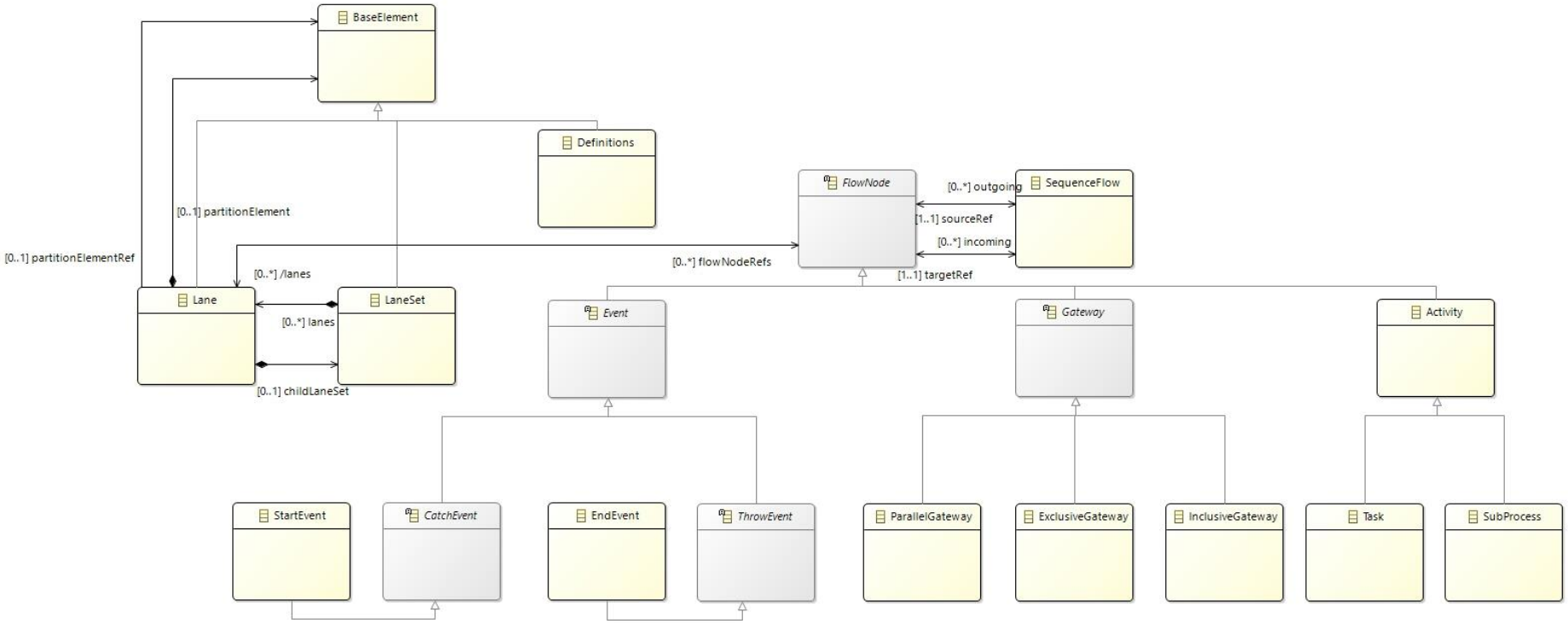


Figura 35: *Extracto del metamodelo de BPMN.*

- **Catch Event** (Evento de captura): Eventos que tienen un disparador definido. Cuando el disparador del evento ocurre, por ejemplo, un mensaje se recibe los datos se asignan de forma automática a la salida de datos que corresponde a la definición del evento que describe este disparador y que está contenida en el atributo EventDefinition.
- **Start Event** (Evento de inicio)**:** El evento de inicio indica cuando un proceso en particular comienza. El evento de inicio comienza el flujo del proceso por lo que no tiene ningún enlace de secuencia de entrada.
- **End Event** (Evento de fin): El evento que indica cuando un proceso en particular funciona. Es el último elemento del flujo.
- **Gateway** (Puerta): Una puerta se usa para controlar cómo los flujos de secuencia interaccionan conforme convergen y se divergen en el interior de un proceso. La puerta controla los datos, implicando que existe un mecanismo que o bien permite o deniega el paso a través de la puerta.
- **Exclusive Gateway** (Puerta exclusiva): Tipo de puerta que al evaluar las expresiones asociadas a sus flujos de salida, la primera expresión evaluada a True determinará el flujo de secuencia por el que continuará el proceso.
- **Inclusive Gateway** (Puerta inclusiva)**:** Tipo de puerta que al evaluar las expresiones asociadas a sus flujos de salida, seleccionará todas aquellas que sean evaluadas a true para que el proceso continúe por ellas.
- **Parallel Gateway** (Puerta paralela)**:** Tipo de puerta usada para combinar y crear flujos paralelos. Es decir, combina flujos que se están siguiendo de forma paralela en uno solo (el flujo de salida de la puerta) o de forma inversa, transforma un flujo de entrada en varios flujos de salida que se sigan por el proceso de forma paralela.
- **Activity** (Actividad): Es un punto del flujo de proceso en el que se realiza trabajo. Una actividad puede ser atómica o compuesta. Una actividad compuesta puede estarlo de otras actividades atómicas o compuestas (procesos).
- **Task** (Tarea): Es un trabajo atómico que se tiene que realizar, es decir es la unidad de trabajo más pequeña que existe.
- **Subprocess** (Subproceso): Una actividad cuyos detalles internos han sido modelados utilizando actividades, puertas, eventos y flujos. Un sub-proceso aparece en el interior de un proceso, pero puede ser abierto para enseñar un proceso de nivel más bajo. Un sub-proceso define un alcance contextual que puede ser utilizado para dar visibilidad a atributos, como alcance transaccional, para gestionar excepciones o eventos. Puede estar contraído o expandido, escondiendo o mostrando sus detalles.

7.4. Transformación entre modelos

Una vez presentado el mapeo conceptual entre los distintos elementos de los metamodelos, pasamos a realizar este mapeo de una forma más técnica, teniendo en cuenta los distintos elementos de los metamodelos.

Value@GRL	BPMN
GRLspec	Definitions, Process, Laneset
Actor	Lane
Goal	Task, subProcess
Softgoal	-
Task	Task
Goal con descomposición	subProcess
Softgoal con descomposición	-
Task con descomposición	subProcess
Descomposición AND	startEvent, endEvent
Descomposición OR	startEvent, inclusiveGateway, sequenceFlow, endEvent
Descomposición XOR	startEvent, exclusiveGateway, sequenceFlow endEvent
Dependencia	sequenceFlow, intermediateThrowEvent, intermediateCatchEvent
Contribución	-

Tabla 1: *Transformaciones entre el metamodelo de Value@GRL y BPMN.*

Como hemos mencionado en el apartado anterior, los objetivos (Goal), objetivosSoft (softGoal) y tareas (Task) se transforman de forma distinta dependiendo de si tienen o no descomposición, además cada una de las descomposiciones tiene una transformación distinta.

7.5. Prototipo de la transformación

Se ha implementado un pequeño prototipo de las transformaciones mostradas en la sección anterior, utilizando el lenguaje de transformación de modelo a modelo (M2M) de ATL (Atlas Transformation Language).

```
rule root2root
  {
    from
      grlspec : grl!GRLspec
    to
      definitions : bpmn2!Definitions (
        rootElements <- process,
        targetNamespace <- 'http://www.jboss.org/drools'
      ),
      process : bpmn2!Process (
        laneSets <- laneset,
        flowElements <- grlspec.intElements
      ),
      laneset : bpmn2!LaneSet (
        lanes <- grlspec.actors
      )
  }
```

Código 4: *Prototipo de transformación de GRLspec a Definitions, Process y Laneset.*

```
rule actor2lane
   {
      from
        actor : grl!Actor
      to
        lane : bpmn2!Lane (
           name <- actor.name
        )

   }
```

Código 5: *Prototipo de transformación de un actor a una lane.*

```
rule task2task
   {
      from
        ielem : grl!IntentionalElement (ielem.type = #Task and
not ielem.linksDest->exists(x |
x.oclIsTypeOf(grl!Decomposition)))
      to
        bpmn2 : bpmn2!Task (
           name <- ielem.name,
           lanes <- ielem.refs->collect(a | a.contRef.contDef)

        )
   }
```

Código 6: *Prototipo de transformación de una tarea a una tarea.*

```
rule taskdescompositionAND2subprocess
   {
      from
        ielem : grl!IntentionalElement (ielem.type = #Task and
ielem.linksDest->exists(x | x.oclIsTypeOf(grl!Decomposition))
and ielem.decompositionType = #And)
      to
        bpmn2 : bpmn2!SubProcess (
           name <- ielem.name,
           lanes <- ielem.refs->collect(a | a.contRef.contDef),
           flowElements <- ielem.linksDest->collect(s | s.src)
        )
   }
```

Código 7: *Prototipo de transformación de una tarea con descomposición de tipo AND.*

En la **Figura *36*** que se muestra a continuación, se muestra un ejemplo de transformación del modelo de objetivos que estamos presentando a lo largo de la memoria a un modelo de BPMN mediante la utilización de los prototipos propuestos. El color Rojo corresponde con el **Código *4***, el azul con el **Código *5***, mientras que el color verde corresponde con **Código *6***.

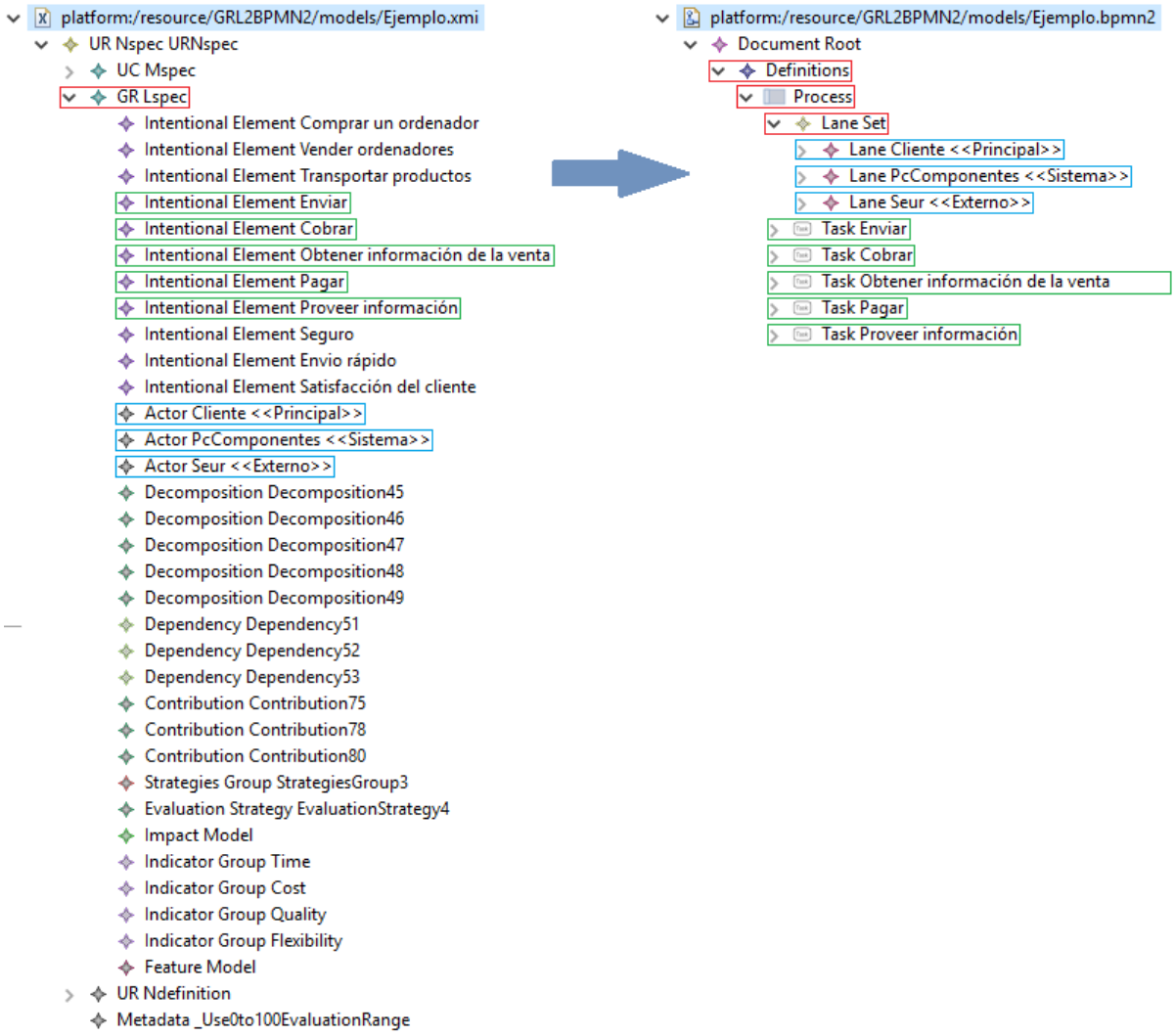


Figura 36: *Utilización del prototipo para transformar el modelo de objetivos al modelo de BPMN.*

8. Priorización de actividades del proceso de negocio

La priorización de actividades del proceso de negocio consiste en juntar las salidas de las actividades definición de modelo de valor y la definición de modelos de proceso de alto nivel, las cuales proveen del modelo de valor, y del modelo de procesos de negocio respectivamente, para así poder generar una lista ordenada (de mayor a menor valor) de las actividades del proceso que hay que implementar.

La **Tabla *2*** que mostramos a continuación representa la lista ordenada de las actividades del modelo del proceso de negocio, las cuales han sido valoradas por un experto para añadirles el esfuerzo (cosa habitualmente utilizada en el desarrollo incremental. En cuanto a lo que aporta nuestro método son las dos columnas siguientes, el valor calculado que proviene del modelo de valor, y la prioridad, donde hemos utilizado el valor calculado, junto con el esfuerzo para priorizar las distintas actividades del modelo de procesos de negocio. Las actividades de la tabla siguiente están ordenadas en base a la prioridad, calculada como el valor entre el esfuerzo. Hay que entender que en este ejemplo hemos utilizado el esfuerzo junto con el valor calculado para priorizar, pero sería posible utilizar otras variables.

Actor	Actividades del modelo de procesos de negocio	Esfuerzo	Valor calculado	Prioridad (Valor/Estuerzo)
Sistema	Obtener información de la venta	50	339	6,78
	Cobrar	100	414	4,14
	Satisfacción del cliente	50	162	3,24
	Enviar	100	295	2,95
	Vender ordenadores	75	137	1,82
Cliente	Envío rápido	25	162	6,48
	Comprar un ordenador	50	237	4,74
	Proveer información	50	218	4,36
	Pagar	75	268	3,57
	Seguro	100	200	2

Tabla 2: *Elementos intencionales con su importancia y valor calculado.*

Una vez que los desarrolladores han obtenido una lista ordenada de los elementos ordenados por valor semejante a la mostrada en la **Tabla 2**, deberán hacer uso del modelo de procesos de negocio generado anteriormente y de la lista para seleccionar aquellos elementos a incluir en el próximo incremento de la aplicación que están desarrollando.

Así pues, los desarrolladores podrán hacer que los incrementos que implementen tengan el mayor valor posible y que gracias al modelo de procesos de negocio generado previamente, tengan un conocimiento general de cómo funcionará el sistema que van a desarrollar, así como también les ayudará a seleccionar que elementos incluir en el incremento, de acuerdo a cómo se relaciona con el resto elementos.

9. Ejemplo de uso del método

Como caso de estudio de este trabajo de fin de máster, vamos a presentar un sistema de reserva de habitaciones de hoteles.

Un hotel desea desarrollar una aplicación que permita gestionar la reserva de las habitaciones del mismo incluyendo el sistema de cobros. Para ello, los clientes deberán primero **buscar una habitación** de acuerdo a sus criterios (tipo de habitación, fecha…) y **reservarla**. Además, los clientes, cuando finalicen su estancia deberán **hacer *checkout*** a la salida, para que el sistema sepa cuando debe cobrarles. Los clientes desean que el sistema sea **seguro**, debido a que van a realizar cobres mediante el mismo.

En cuanto al sistema, este deberá **comprobar las habitaciones disponibles** y devolvérselas al cliente de acuerdo con la búsqueda realizada por el usuario; **almacenar las reservas** y **cobrar**. Además, se desea que el sistema esté **disponible**, para no perder posibles clientes y que sea **fácil de utilizar**.

Para realizar cobros, el sistema usará el sistema de SCB (Sistema de Cobros Bancarios), el cual ofrece un **sistema para realizar cobros**.

9.1. Modelado de objetivos con Value@GRL

De acuerdo con la guía de modelado de objetivos con Value@GRL primeramente se debe identificar los actores, y el tipo que son. En este caso, podemos identificar los actores de **usuario**, el cual es de tipo actor principal, ya que es en el que nos enfocamos, **el sistema de reservas** (el cual vamos a llamar **RH**), que es el actor sistema, ya que es el sistema que deseamos desarrollar, y el **SCB**, como el actor externo. En la **Figura 37** mostramos la representación de los distintos actores del caso de estudio.

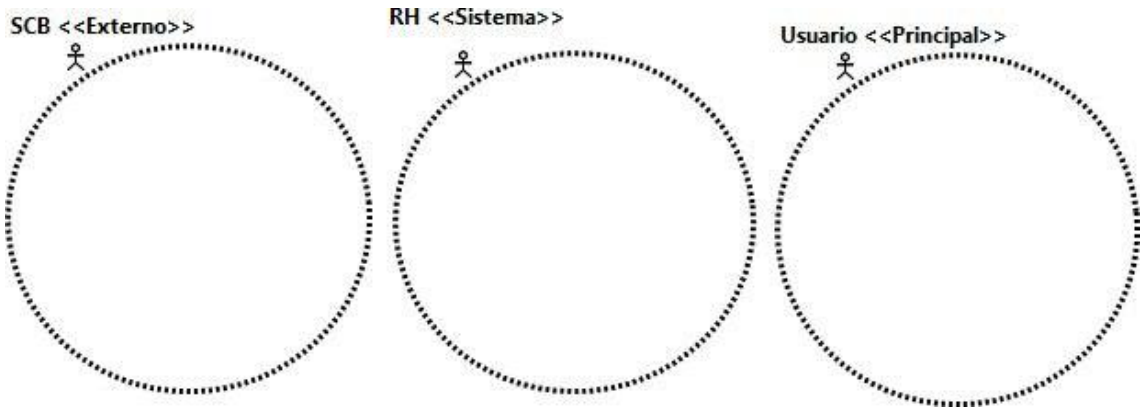


Figura 37: *Actores identificados en el caso de estudio.*

Una vez identificados los actores y el tipo que son, hay que modelar los elementos intencionales de estos. En este caso, podemos identificar como objetivo del usuario el de "reservar habitación de hotel", para lo cual deberá realizar las tareas de "buscar habitación" y "reservar habitación", además que deberá realizar la acción de "hacer *checkout*" a la salida. A parte, desea que el sistema sea seguro, que corresponde con un objetivoSoft. En cuanto al SCB únicamente tiene como objetivo ofrecer un sistema de cobros. El resultado de modelar esto, es el mostrado en la **Figura 38**.

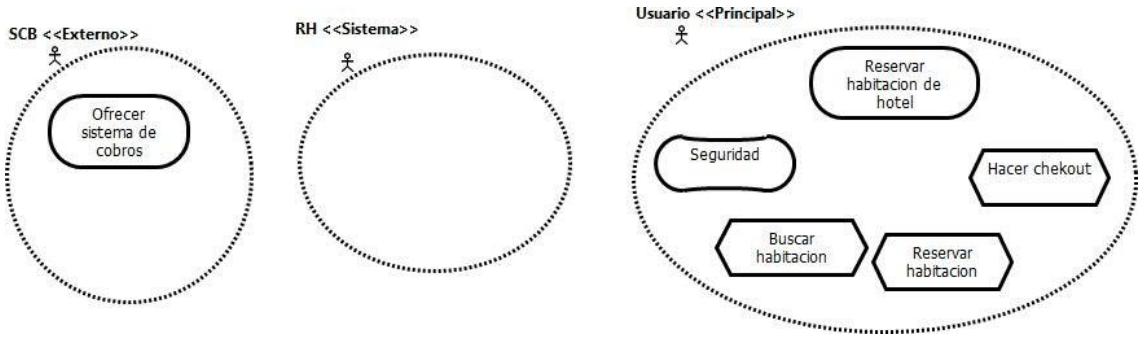


Figura 38: *Actores del caso de estudio con los elementos intencionales.*

El siguiente paso según la guía de modelado, es el de poner los enlaces internos entre los distintos elementos intencionales modelados anteriormente. Para poder "reservar una habitación de hotel", es necesario que el usuario realice las tareas de "buscar habitación" y "reservar una habitación", lo cual correspondería con una descomposición. Además de esto hay una contribución, entre "seguridad" y "reservar habitación de hotel", ya que si el sistema es seguro sería más posible que el usuario utilizara el sistema.

En la **Figura 39** únicamente mostramos el actor principal, ya que en el actor externo no podemos modelar ninguna relación interna.

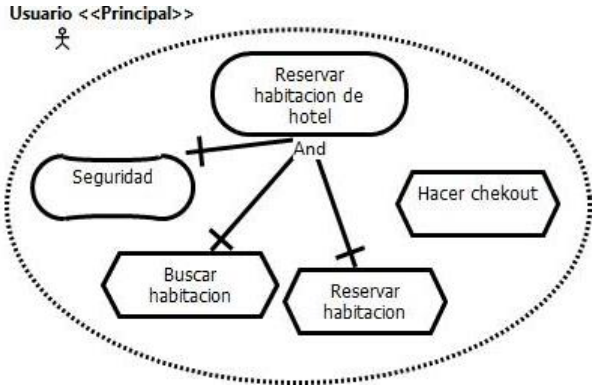


Figura 39: *Actor principal del caso de estudio con los elementos intencionales y sus enlaces.*

Cuando ya se ha modelado todos los elementos intencionales y los enlaces internos de los actores externo y principal, se procede a modelar los del actor sistema. Nuestro actor sistema tiene como tareas a realizar la de "comprobar las habitaciones disponibles", para poder realizar las reservas, "almacenar la reserva" hecha por el usuario y "cobrar". Además, se desea que el sistema sea "fácil de usar" y que tenga una buena "disponibilidad", para evitar la pérdida de clientes. En la **Figura 40** se muestra el resultado de modelar el actor sistema con todos lo citado.

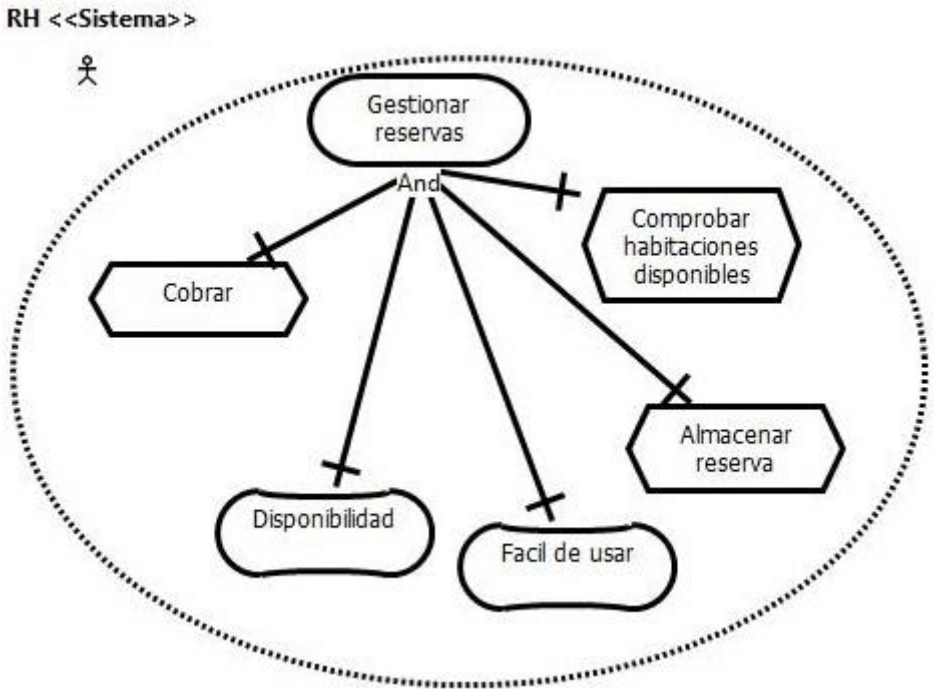


Figura 40: *Actor sistema con sus elementos intencionales y enlaces.*

Una vez que se han modelado todos los actores con sus respectivos elementos intencionales y relaciones internas, se procede a crear los enlaces externos entre los distintos actores. En este caso, los enlaces que se han modelado son entre "buscar habitación" y "comprobar habitaciones disponibles", ya que el cliente debe realizar una búsqueda con los criterios que le interese, y el sistema deberá comprobar que habitaciones hay disponibles de acuerdo a los criterios proporcionados. Luego el cliente, deberá seleccionar una habitación de las que le ha ofrecido el sistema y reservarla, la cual el sistema deberá reservar. A parte de estos enlaces externos, también existe uno más que es que cuando el sistema quiere cobrar utiliza el sistema de cobros que utiliza el SCB. En la Figura 43 se muestra el resultado de modelar todo esto.

9.2. Definición del modelo de valor

Una vez se ha realizado el modelo de valor, se procede a asignar la importancia de los elementos del modelo, dando como resultado lo que puede observarse en la **Figura 41**. Y una vez asignada la importancia, aplicar la fórmula para propagarla y calcular el valor resultante, como puede verse en la **Figura 42**.

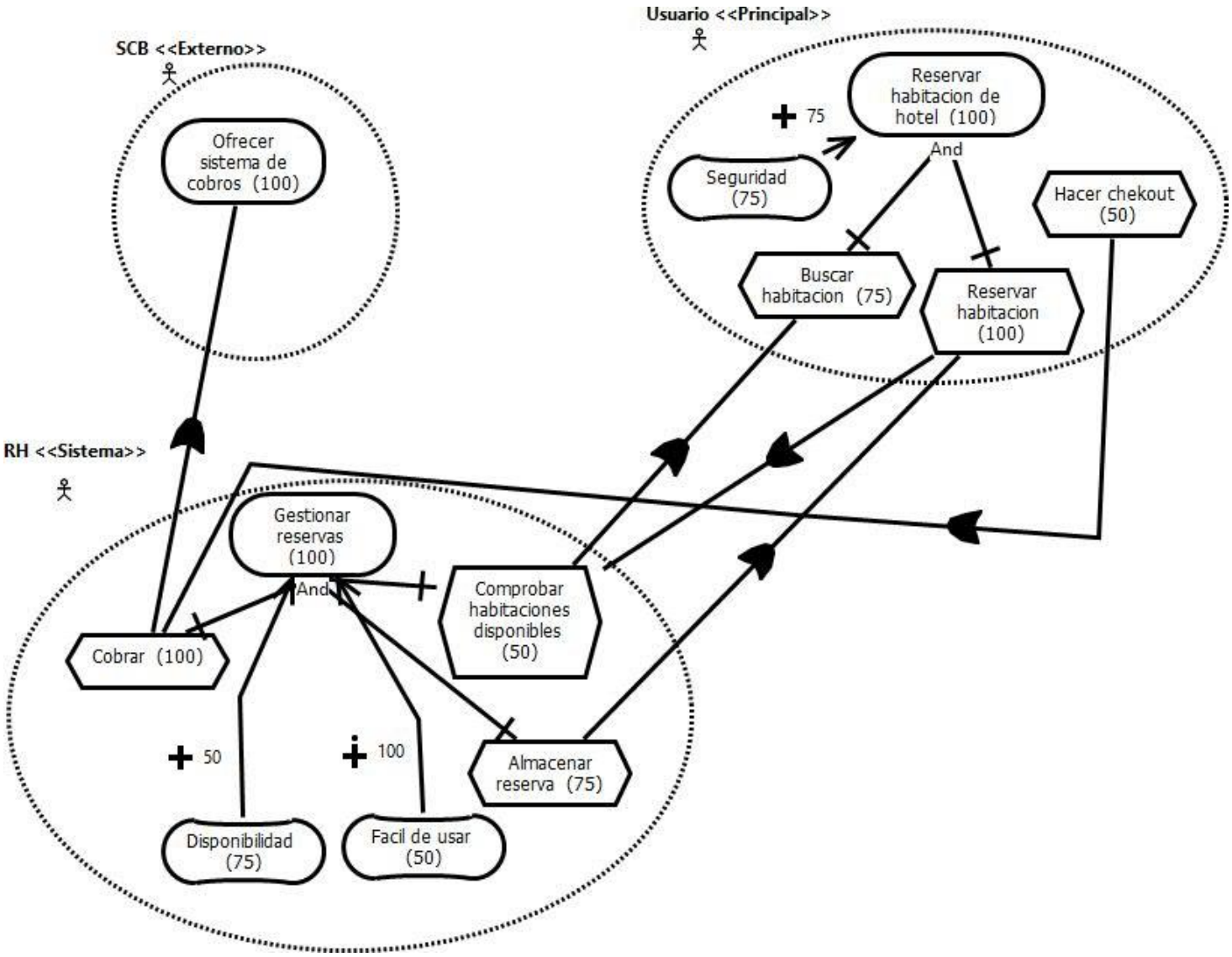


Figura 41: *Modelo de objetivos con la importancia asignada.*

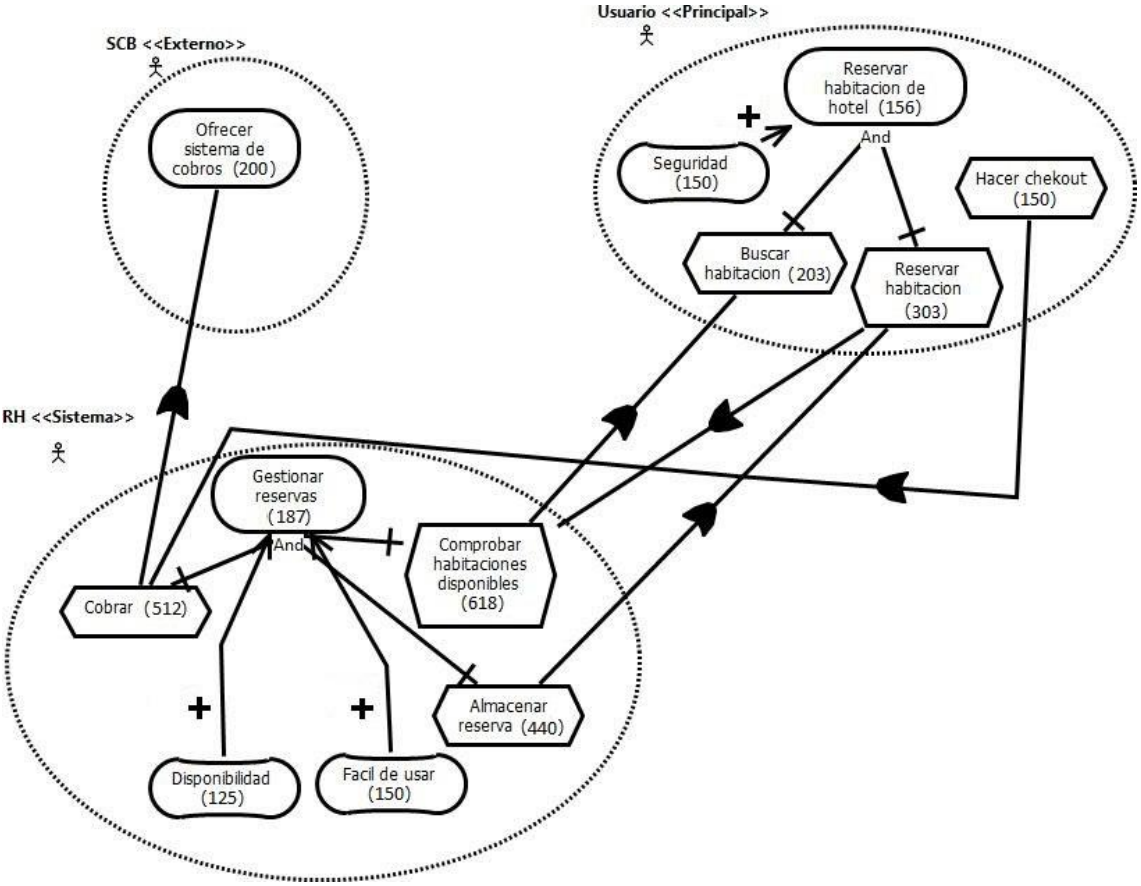


Figura 42: Modelo de valor con el valor calculado.

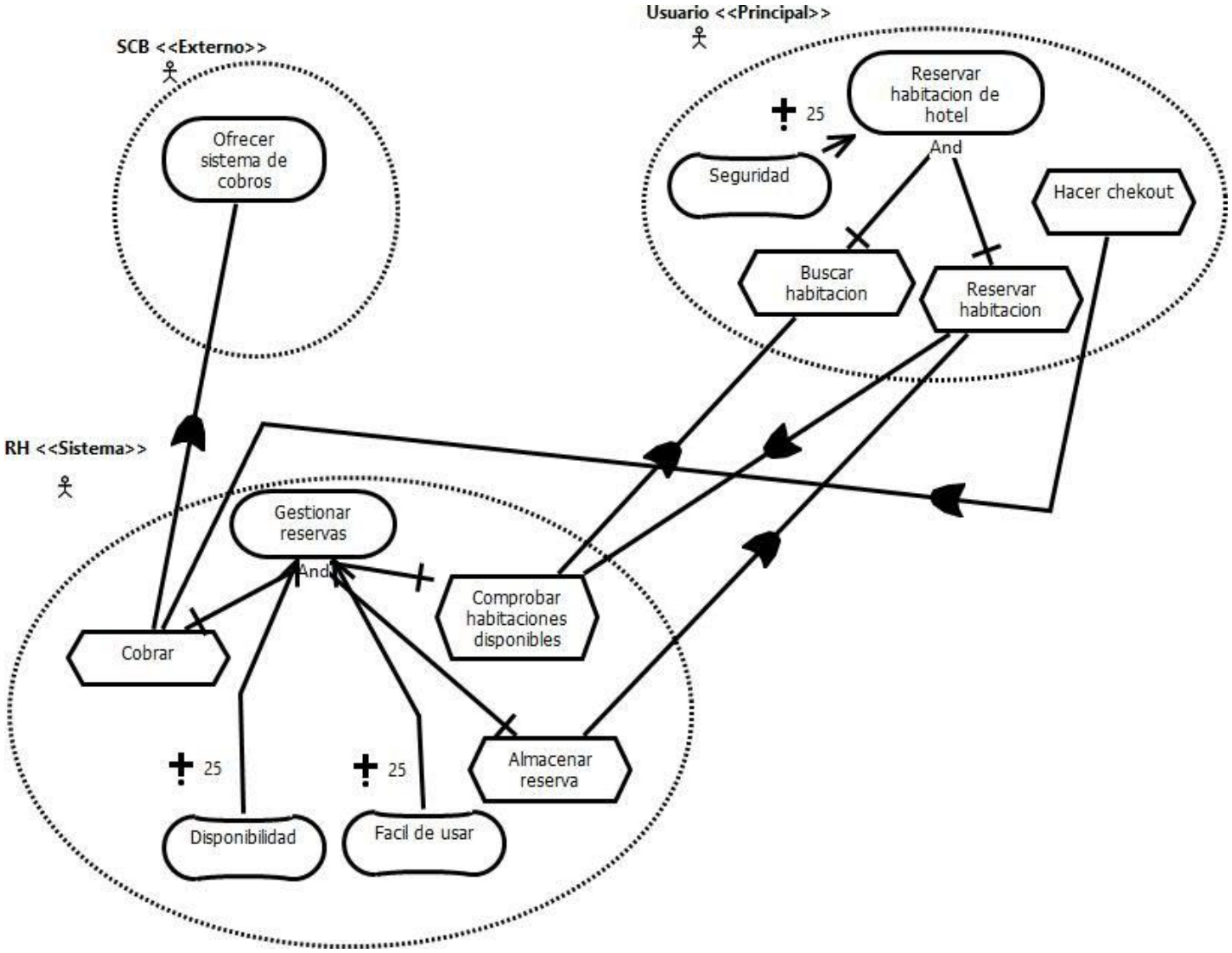


Figura 43: *Modelo de valor con todos los elementos intencionales y enlaces tanto internos como externos.*

9.3. Definición de procesos de negocio de alto nivel

Una vez realizado el modelo de objetivos es posible realizar la transformación a un modelo de procesos de negocio siguiendo las guías de modelado, como se muestra en la **Figura 44**.

Una vez realizada la transformación, es necesario que un experto la modifique añadiendo las actividades que falten, así como también el orden de precedencia, lo cual mostramos en la **Figura 45**.

En este caso en el modelo de procesos de negocio no hemos incluido ninguna tarea nueva, pero dependiendo del contexto del sistema podrían añadirse, por ejemplo, si se ofertan múltiples opciones a la hora de buscar la habitación, o si permite que el usuario pague en efectivo, depende bastante del nivel de abstracción que utilice el modelador del modelo de objetivos.

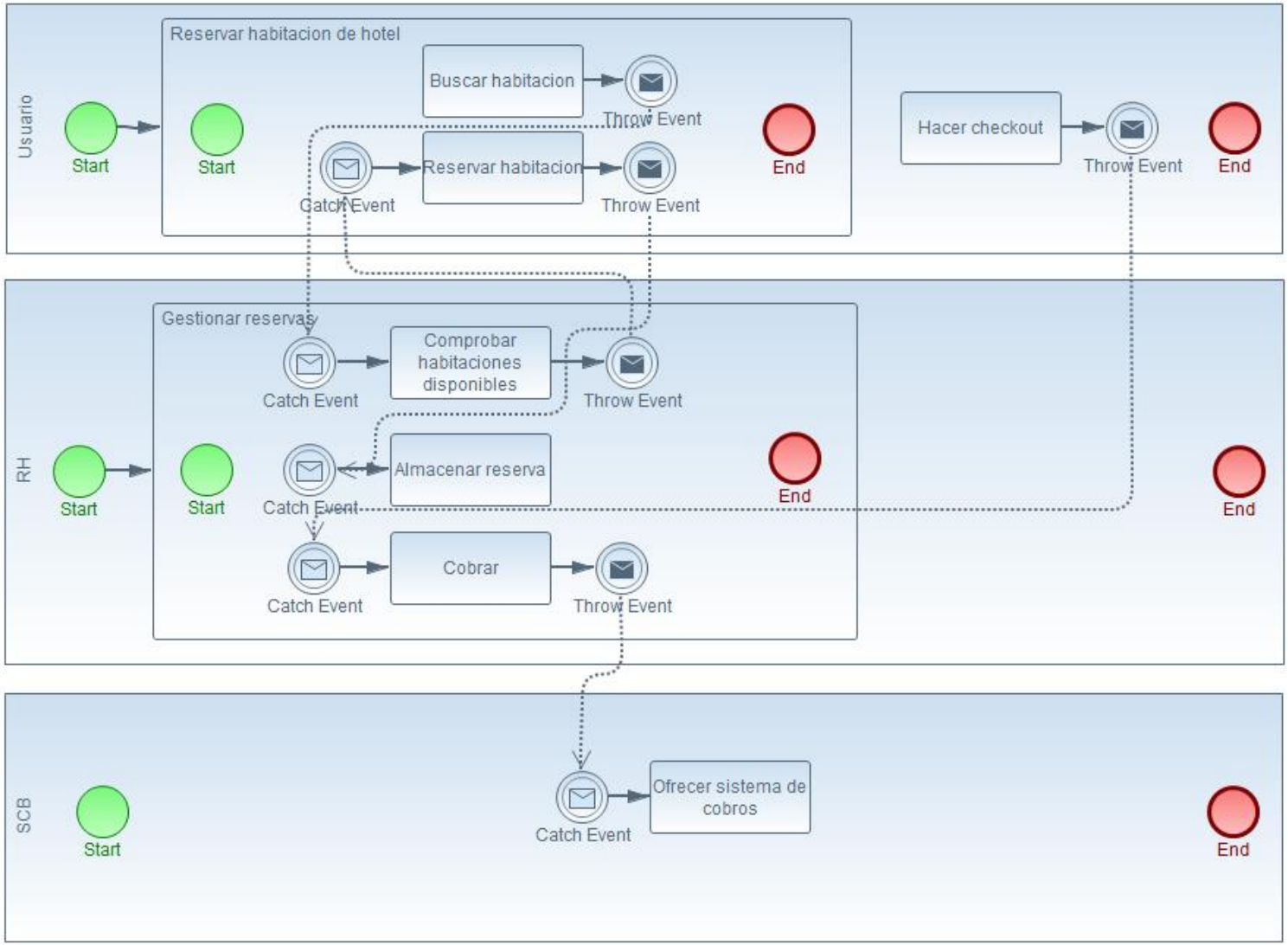


Figura 44: *Transformación del modelo de valor a modelo de procesos de negocio.*

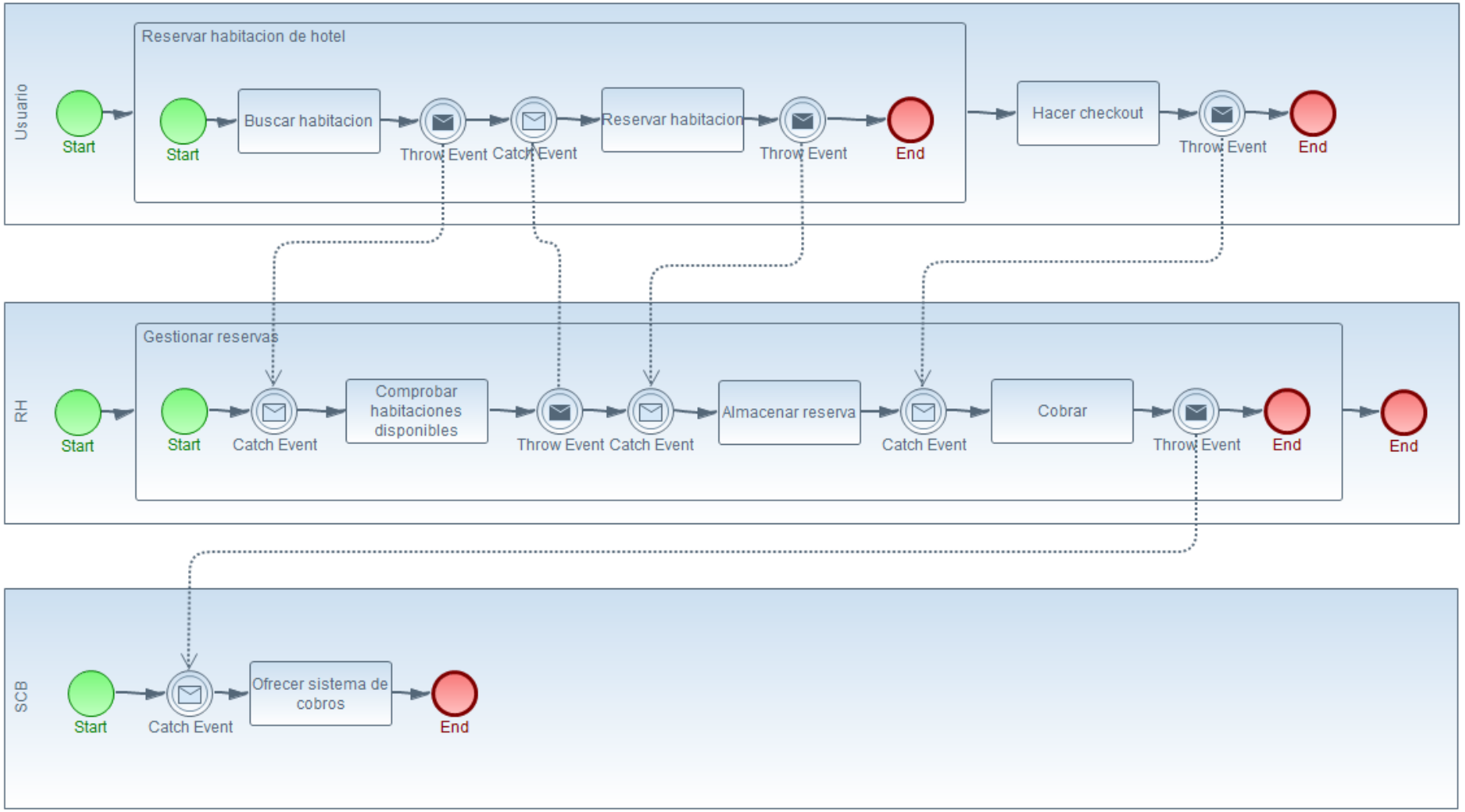


Figura 45: *Modificación del modelo BPMN generado.*

9.4. Priorización de actividades del proceso de negocio

La última actividad del método es la de priorizar los elementos intencionales del modelo de acuerdo con el valor calculado. El resultado de esta priorización puede verse en la **Tabla 3**. Donde, previamente un experto ha valorado el esfuerzo necesario para realizar las actividades del modelo de procesos de negocio, y junto con el valor, se ha ordenado las distintas actividades mediante el resultado de dividir el valor entre el esfuerzo.

Actor	Elemento intencional	Esfuerzo	Valor calculado	Prioridad (Valor/Estuerzo)
Sistema	Comprobar habitaciones disponibles	50	618	12,36
	Almacenar reserva	50	440	8,8
	Cobrar	100	512	5,12
	Fácil de usar	50	150	3
	Disponibilidad	50	125	2,5
	Gestionar reservas	75	187	2,5
Usuario	Buscar habitación	50	203	4,06
	Reservar habitación	75	303	4,04
	Reservar habitación de hotel	50	156	3,12
	Seguridad	75	150	2
	Hacer checkout	75	150	2

Tabla 3*: Lista priorizada por valor de los elementos intencionales.*

Una vez que se ha obtenido los elementos intencionales del modelo, y generado el modelo de procesos de negocio. Los desarrolladores deberán utilizar la lista priorizada de elementos intencionales, junto con el modelo de procesos de negocio para seleccionar que elementos incluir en cada uno de los incrementos.

Con todo esto, los desarrolladores podrán realizar incrementos aportando el mayor valor posible en cada uno de ellos, y gracias al modelo de procesos de negocio sabrán que elementos están relacionados entre sí, y el orden de precedencia de los mismos. Además, el modelo de procesos de negocio les proveerá de una idea de cómo funcionará el sistema que van a desarrollar.

10. Evaluación empírica de Value@GRL

Esta sección presenta la evaluación empírica de Value@GRL mediante un experimento controlado y su replicación en los que se compara este lenguaje con respecto a uno alternativo (i*). A continuación, se presenta los lenguajes de modelado a comparar, así como el diseño, ejecución y resultados de los experimentos.

10.1. Lenguajes de modelado a comparar

10.1.1. Value@GRL

Este lenguaje de modelado lo hemos presentado previamente en la **sección 5** de este trabajo, básicamente este lenguaje está basado en el lenguaje GRL el cual se ha delimitado en cuanto a la cantidad de elementos existentes, así como también se ha añadido la posibilidad de añadir *valor* tanto a los elementos intencionales, como a las contribuciones. Aunque **para este experimento no se ha tenido en cuenta la parte de valor**, y se han comparado los lenguajes como si fueran lenguajes de modelado de objetivos.

10.1.2. i* (i-star)

Este lenguaje de modelado ha sido desarrollado por Eric Yu y presentado por primera vez en [68]. i-Star es un lenguaje de modelado de objetivos que permite razonar sobre un entorno organizativo y sus sistemas de información que están compuestos por actores heterogéneos con objetivos diferentes e incluso competitivos. El método permite modelar los actores que participan en el sistema y las relaciones existentes entre ellos, ayudando así a entender los requisitos del sistema.

De forma semejante a Value@GRL, este lenguaje de modelado está compuesto por distintos elementos intencionales, algunos de los cuales tiene en común con GRL y otros propios.

10.1.2.1. Elementos intencionales

Los elementos intencionales más representativos que posee i* son los siguientes:

- **Objetivo:** objetivo de alto nivel del negocio, de la organización, del sistema o, incluso, de los *stakeholders*. Capturan la finalidad o propósito del sistema, y guían el proceso de decisión entre las alternativas para su realización durante el proceso de desarrollo del software. Es un objetivo funcional del sistema. Se caracteriza porque es posible verificar, de manera objetiva, si se ha cumplido el objetivo.
- **ObjetivoSoft:** objetivo cuya satisfacción o cumplimiento no puede ser establecido de manera absoluta, es decir, que en algunos casos solo se podrá afirmar que se ha cumplido (o no) de manera parcial. Este tipo de objetivo suele estar relacionado con requisitos no funcionales.
- **Tarea:** manera de alcanzar un objetivo. Puede ser de bajo nivel como un procedimiento o algoritmo o de mayor nivel como un entorno o interfaz de usuario.
- **Recurso:** objeto que es necesario para realizar una tarea o para alcanzar un objetivo. Un ejemplo de un recurso podría ser un documento entre dos actores.

Como hemos mencionado en el apartado anterior, en general, las distintas propuestas de modelado orientado a objetivos utilizan un mismo conjunto de elementos intencionales por lo que los elementos del método i* son los mismos del método Value@GRL, a excepción del elemento Recurso que está presente solo en i*.

10.1.2.2. Relaciones

Los tipos de relaciones que i* posee son semejantes a los que GRL o Value@GRL tiene, pero delimitados en cuanto a qué elementos puede relacionar, y algunos cambios.

- Descomposición: Permite que una tarea, se descomponga en más tareas, siendo estas una unidad más pequeña y detallada de la tarea padre.
- Contribución: Representa un efecto, positivo o negativo de un elemento intencional sobre otro.
- Dependencia: Describe como un elemento intencional depende de otro para poder cumplirse. Esta relación solo puede realizarse entre elementos intencionales de distintos actores y necesita de un elemento intencional que indique el motivo de la dependencia.

10.1.3. Comparativa entre Value@GRL e i*

Cómo ha podido observarse en las secciones de Value@GRL e i-Star, ambos lenguajes de modelado son similares, aunque tienen ciertas diferencias. Las cuales son que i* posee un elemento intencional más que es el de **recurso**, que **delimita las relaciones** entre los elementos ya que ofrece menor libertad a la hora de realizar los enlaces, y que la **dependencia necesita de un elemento intermedio** para poder relacionar dos elementos intencionales.

10.2. Planificación del experimento

10.2.1. Objetivo

De acuerdo al paradigma GQM (*Goal Question Metric*) [3], el objetivo del experimento se puede definir de la siguiente manera:

Analizar *los modelos de objetivos creados con Value@GRL e i** ***con el propósito*** *de compararlos* ***con respecto*** *a la calidad de los modelos producidos en términos de su corrección y completitud, y del tiempo de modelado, productividad, facilidad de uso percibida, utilidad percibida e intención de uso* ***desde el punto de vista*** *de ingenieros de software y analistas de negocio noveles que utilizan estos métodos para modelar los objetivos de sistemas de información,* ***en el contexto*** *de un grupo de estudiantes de master y grado de la Universitat Politècnica de València.*

Aunque es deseable involucrar a ingenieros de software con experiencia y profesionales, nos centramos en el perfil de los modeladores noveles, ya que uno de nuestros objetivos es proporcionar un método de modelado de objetivos que ayude a los ingenieros de software y analistas de negocio con menos experiencia en la especificación de modelos de objetivos.

Las tres Preguntas de Investigación (PI) abordadas en este estudio son:

- PI1: ¿Value@GRL permite a los modeladores obtener modelos de objetivos de mayor calidad que con i *?
- PI2: ¿Value@GRL permite a los modeladores ser más productivos (obtener modelos de objetivos con mayor calidad en menos tiempo) que con i *?
- PI3: ¿Es Value@GRL percibido como más fácil de usar, útil, y es más probable que se utilice que i* para la especificación de modelos de objetivos?

10.2.2. Variables

La variable independiente es el lenguaje de modelado de objetivos, el cual tiene dos posibles valores nominales: Value@GRL e i*. Por lo tanto, hay dos posibles tratamientos: La especificación de un modelo de objetivos para dos sistemas software y la especificación del modelo de objetivos, para el mismo sistema software utilizado i*.

En cuanto a las variables de tipo dependientes, hay de dos tipos: basadas en desempeño (*performance-based*) y basadas en percepción (*perception-based*).

Las variables basadas en desempeño (es decir, calidad y productividad) evalúan como de bien los participantes han realizados las tareas del experimento.

Calidad: Indica la calidad del modelo producido con Value@GRL o i* en términos de correctitud (si el modelo se ajusta a las reglas del lenguaje de modelado de objetivos) y completitud (si el modelo contiene toda la información requerida para representar las propuestas de los diferentes *stakeholders*). Esta variable se mide mediante una aproximación de recuperación de información [69] que ha sido utilizada en otros experimentos de la ingeniería del software [70] [68] para comparar modelos creados por los participantes contra un *oráculo* (modelo correcto creado por un experto) de acuerdo con cada tipo de elementos gráfico (como por ejemplo actores, objetivos, objetivosSoft, tareas y enlaces) a través de las fórmulas siguientes:

$$precision_{elemento} = \frac{|P_{elemento} \cap O_{elemento}|}{|P_{elemento}|}$$

$$recall_{elemento} = \frac{|P_{elemento} \cap O_{elemento}|}{|O_{elemento}|}$$

Donde $P_{elemento}$ indica todo elemento gráfico particular modelado por un participante, y un $O_{elemento}$ indica el conjunto correcto de ese elemento gráfico derivado de un oráculo. En consecuencia, la precisión mide la correctitud de un elemento gráfico perteneciente a un modelo de objetivos, y el *recall* mide la completitud del modelo de objetivos con respecto a sus elementos gráficos. La *precision* y *recall* sumados cuantitativamente resumen dos dimensiones de la calidad del modelo. Por lo tanto, hemos utilizado una media armónica [69] para conseguir un equilibrio entre la correctitud y completitud de cada elemento gráfico del modelo de objetivos (fórmula siguiente):

$$F - Measure_e = 2 * \frac{precision_{elemento} * recall_{elemento}}{precision_{elemento} + recall_{elemento}} * 100$$

Donde la $F - Measure$ resume la exactitud de cada elemento gráfico del modelo de objetivos de acuerdo a sus elementos gráficos en comparación con un *oráculo*. La calidad se calcula como la media aritmética de toda la *F-Measure* para los diferentes elementos gráficos del modelo. Todas las medidas anteriores

asumen los valores entre 0% y 100%. Para cualquier medida, si su valor se aproxima al 100% significa que el participante ha hecho un modelo de objetivos muy parecido a un *oráculo*. En cambio, si el valor es cercano a 0%, indica que el modelo dista mucho de un *oráculo*.

La variable de calidad ha sido definida para dar la misma importancia tanto a la correctitud como a la completitud del modelo de objetivos, con respecto a los actores, objetivos, tareas y relaciones existentes en el modelo.

Productividad: La productividad de los participantes es calculada como la división de la calidad del modelo entre el tiempo total gastado en aplicar el método. La calidad es medida en porcentaje, y el tiempo en minutos, por lo tanto, si un participante obtiene una puntuación de 100% en calidad en 20 minutos su productividad debería de ser 5 (Productividad = 100 / 20 = 5). Hemos escogido 20 minutos, debido a que es el tiempo que toma un experto en modelar esta solución. Por tanto, cuanto más cerca de 5 sea la productividad del participante, mejor será su productividad.

Por otro lado, las variables basadas en percepción (perception-based) evalúan la percepción del participante sobre su desempeño y su intención de utilizar el método de Value@GRL o de i* en un futuro. Estas variables subjetivas se basan en el método de Aceptación de la tecnología (Technology Acceptance Model (TAM)) [71] modelo ampliamente aplicado y validado de forma empírica [72]. Las variables seleccionadas son las siguientes:

- **Percepción facilidad de uso**: El grado en el cual los modeladores creen que aprender y usar un método concreto no sirve de nada.
- **Utilidad percibida:** El grado en el cual los modeladores creen que usar un método en concreto incrementará su eficiencia en el trabajo.
- **Intención de uso:** El grado en el cual los modeladores tienen la intención de usar el método en un futuro. Esta última variable representa el juicio de la eficacia del método, es decir, si es rentable, y se utiliza comúnmente para predecir la probabilidad de aceptación de un método.

Se ha incluido la variable de tiempo de modelado, así como su hipótesis correspondiente, debido a que la productividad se calcula como una división de la calidad entre el tiempo de modelado

Tiempo de modelado: Es una variable cuantificada, la cual mide en minutos el tiempo que el participante tarda en realizar el modelo de objetivos. Así como el resto de variables cuanto más alto sea su valor, mejor resultado representa, el tiempo de modelado contra menor sea su valor mejor resultado, pues es más interesante que el método se rápido de aplicar que lento.

Resumiendo, las variables que van a tenerse en cuenta en los experimentos son las mostradas en la **Tabla 4**.

Tipo de variable	Nombre	Métrica
Independiente	Experiencia (Noveles y profesionales)	No valorada*1
	Método (Value@GRL y i*)	No valorada
Dependiente	Calidad	Precisión y completitud*2
	Tiempo de modelado	Tiempo
	Productividad	Eficiencia / Tiempo de modelado
	Facilidad de uso percibida	Encuesta
	Utilidad percibida	Encuesta
	Intención de uso	Encuesta

Tabla 4: *Variables utilizadas en la validación.*

*1: Cada uno de los experimentos ha involucrado participantes de distintos niveles de experiencia, esto se explica en la sección de selección de sujetos.

*2: Para la medición de la precisión y completitud para la variable de *calidad*, se ha utilizado múltiples soluciones posibles (oráculos), debido a que Value@GRL ofrece mucha libertad, es posible realizar el modelo de múltiples formas posibles, para lo cual, se han desarrollado múltiples soluciones y el valor seleccionado es aquel que se aproxima más a una solución, como se ha explicado anteriormente.

Además, cabe mencionar, que la encuesta utilizada para medir las variables de utilidad, facilidad de uso percibida e intención de uso puede encontrarse en el **Anexo E**.

10.2.3. Hipótesis

Estamos interesados en investigar si nuestra propuesta de modelo de objetivos (es decir, Value@GRL) puede obtener mejores resultados y percepciones que i*. Por lo tanto, nosotros primero verificamos la calidad de los modelos de valor producidos por los participantes. Después, estudiamos el efecto de los métodos sobre la productividad, sobre los participantes para crear modelos de valor empezando con un documento de requisitos. Por último, comprobamos el efecto de Value@GRL e i* sobre la percepción de los participantes.

Las hipótesis que se han planteado para los experimentos son las mostradas en la **Tabla 5**, donde hemos propuesto que para todas las variables Value@GRL obtiene mejores resultados que i*.

Nº	Variable	Hipótesis Nula H0	Hipótesis Alternativa H1
1	Calidad	Value@GRL <= i*	Value@GRL > i*
2	Tiempo de modelado	Value@GRL >= i*	Value@GRL < i*
3	Productividad	Value@GRL <= i*	Value@GRL > i*
4	Facilidad de uso percibida	Value@GRL <= i*	Value@GRL > i*
5	Utilidad percibida	Value@GRL <= i*	Value@GRL > i*
6	Intención de uso	Value@GRL <= i*	Value@GRL > i*

Tabla 5: *Hipótesis propuestas para la validación.*

La razón por la que en la hipótesis alternativa $H2_1$ (tiempo de modelado) hemos puesto que sea menor en vez de mayor es debido a que cuanto menor sea el tiempo de modelado más eficiente es el participante en el uso del método.

10.2.4. Contexto

El contexto utilizado en este conjunto de experimentos es el siguiente:

- **On-line:** Los participantes serán supervisados por los experimentadores para evitar que haya interacción entre los participantes y que estos intercambien información durante el experimento. Además, esto permite a los experimentadores solventar las distintas dudas que puedan tener los participantes.
- **Profesionales y usuarios noveles:** Como hemos mencionado anteriormente, en estos experimentos nos centraremos en modeladores noveles. En cada uno de los experimentos hay un perfil distinto de participantes, como se describe a continuación.
- **Problemas reales:** Para el diseño del experimento se han escogido un conjunto de problemas reales extraídos de diversos artículos publicados, y se han modificado para que se adecue al tiempo disponible en cada sesión experimental.

10.2.5. Selección de los participantes

Los participantes del estudio fueron seleccionados por conveniencia. Se tratan de alumnos de master o grado que asistían a cursos específicos relacionados con el modelado de software.

Perfil de los sujetos:

- **Experimento Base (UPV1):** Estudiantes de máster de la Universitat Politècnica de Valencia, tanto estudiantes de Ingeniería Informática como estudiantes de Administración y Dirección de Empresas (ADE).
  - **Ingenieros de Software noveles:** 4 estudiantes de doctorado de ingeniería informática y 16 estudiantes de máster en ingeniería informática cursando un máster en el DSIC en la Universitat Politècnica de València (UPV), España. Una encuesta previa fue proporcionada a los participantes con tal de comprender su experiencia. Ocho de los participantes reportaron tener conocimiento previo en el desarrollo software entre 2 y 7 años, con una media de 3 años, pero no tenían ningún conocimiento en Value@GRL o i*.

  - **Analistas de negocio noveles**: 12 estudiantes de un máster en Empresas, productos y gestión de servicios en la facultad de Administración y Dirección de Empresas (ADE) de la UPV. Con perfiles de negocio, economía e ingeniería industrial. Los participantes ya tenían un conocimiento previo sobre algunas áreas de modelado de procesos de negocio.

- **Replicación (UPV2):** Estudiantes del grado de Ingeniería Informática en la Universitat Politècnica de Valencia. Se tratan de 36 alumnos de la asignatura Calidad de Software de la Rama de Ingeniería de Software.
  - Todos los estudiantes fueron voluntarios, y eran conscientes de los propósitos prácticos y pedagógicos del experimento, pero las preguntas de investigación no les fueron reveladas.

10.2.6. Objetos experimentales

Para la realización de este experimento se han seleccionado dos casos de estudio distintos, siendo estos los siguientes:

- O1 – Green Route [73]: El sistema pretende proveer al usuario de una ruta por la ciudad de Valencia que se encuentre libre de contaminación y que se adecue a las preferencias del usuario. Una de las posibles soluciones de este objeto experimental utilizando Value@GRL puede verse en la **Figura 46**.
- O2 – Lattes Schoolar [74]: Este sistema tiene como objetivo listar las publicaciones y citaciones de las publicaciones de un autor. Para ello el sistema debe acceder a bases de datos externas para obtener la información necesaria. Una de las posibles soluciones de este objeto experimental utilizando Value@GRL puede verse en la **Figura 47**.

La razón por la que estos casos de estudio han sido extraídos de artículos es porque ya han sido validados previamente, y que algunos han sido incluso implementados, teniendo así tanto el caso de estudio como una de sus posibles soluciones desarrolladas por un profesional en el tema.

Como hemos mencionado ya múltiples veces, para la evaluación de la calidad, se han desarrollado múltiples soluciones posibles para cada caso de estudio.

10.2.7. Diseño de los experimentos

Para el experimento base y para su replicación hemos utilizado un diseño balanceado de tipo inter-sujetos (between-subject) donde cada parcicipante solo usa uno de los métodos con uno de los objetos experimentales. El experimento ha consistido en dos sesiones, donde en cada una de ellas a los participantes se les ha dado un tratamiento. Hemos establecido cuatro grupos (uno para cada método y objeto) y los participantes fueron asignados de forma aleatoria a cada grupo. Este diseño de experimento mitiga los posibles efectos de aprendizaje, ya que ninguno de los participantes repite ningún tratamiento o método experimental durante la ejecución. Este diseño también nos permite acomodar mejor la disponibilidad de los participantes. Además, los diseños de experimentos cruzados intra-sujetos (within-subjects) (cuando los participantes aplican todos los tratamientos) han sido criticados y/o desaconsejados en la ingeniería del software [75].

	Session 1 (i*)	Session 2 (Value@GRL)
Object 1	Group 1	Group 3
Object 2	Group 2	Group 4

Tabla 6: *Diseño del experimento.*

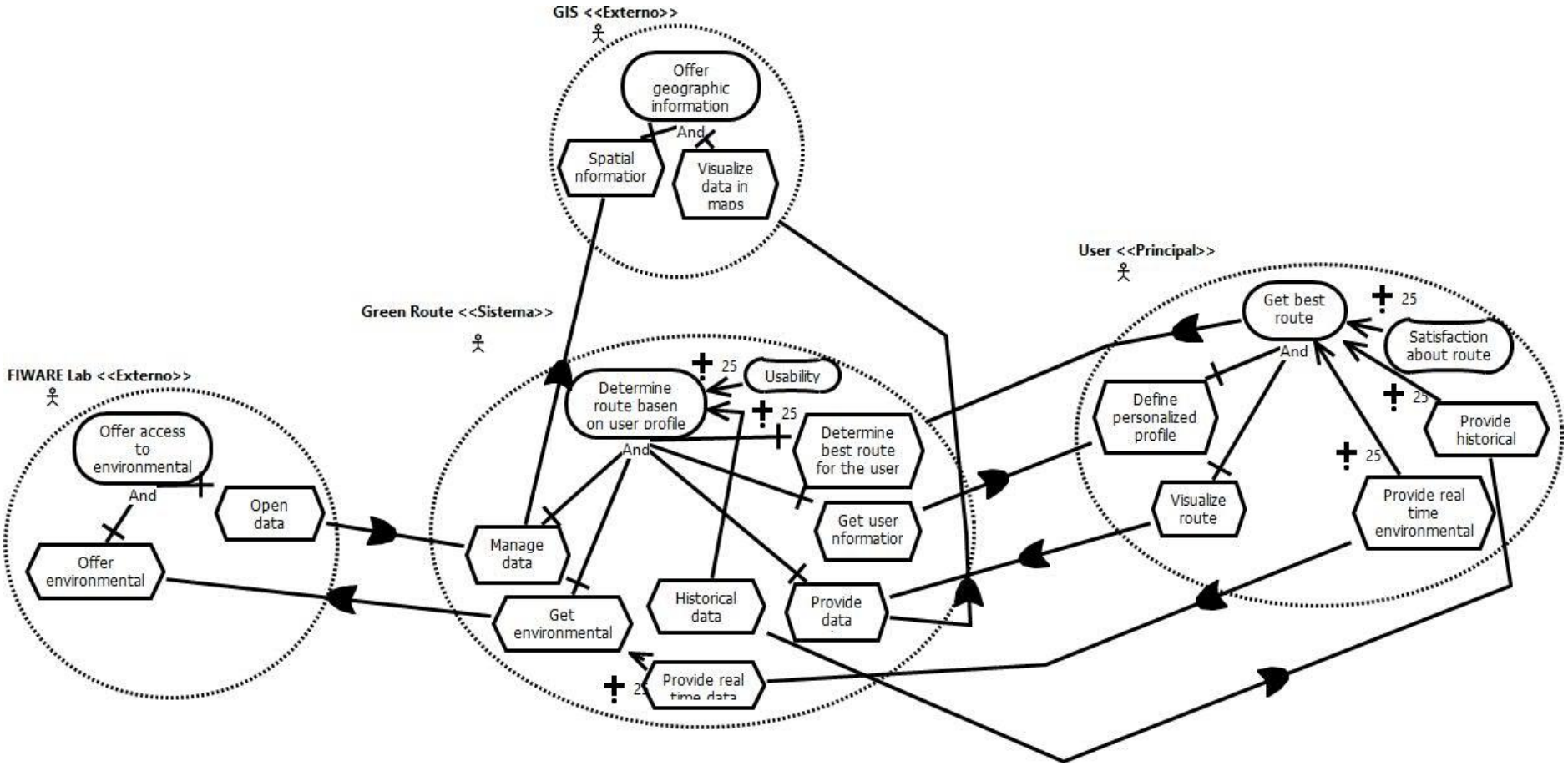


Figura 46: *Posible solución de Green Route utilizando Value@GRL.*

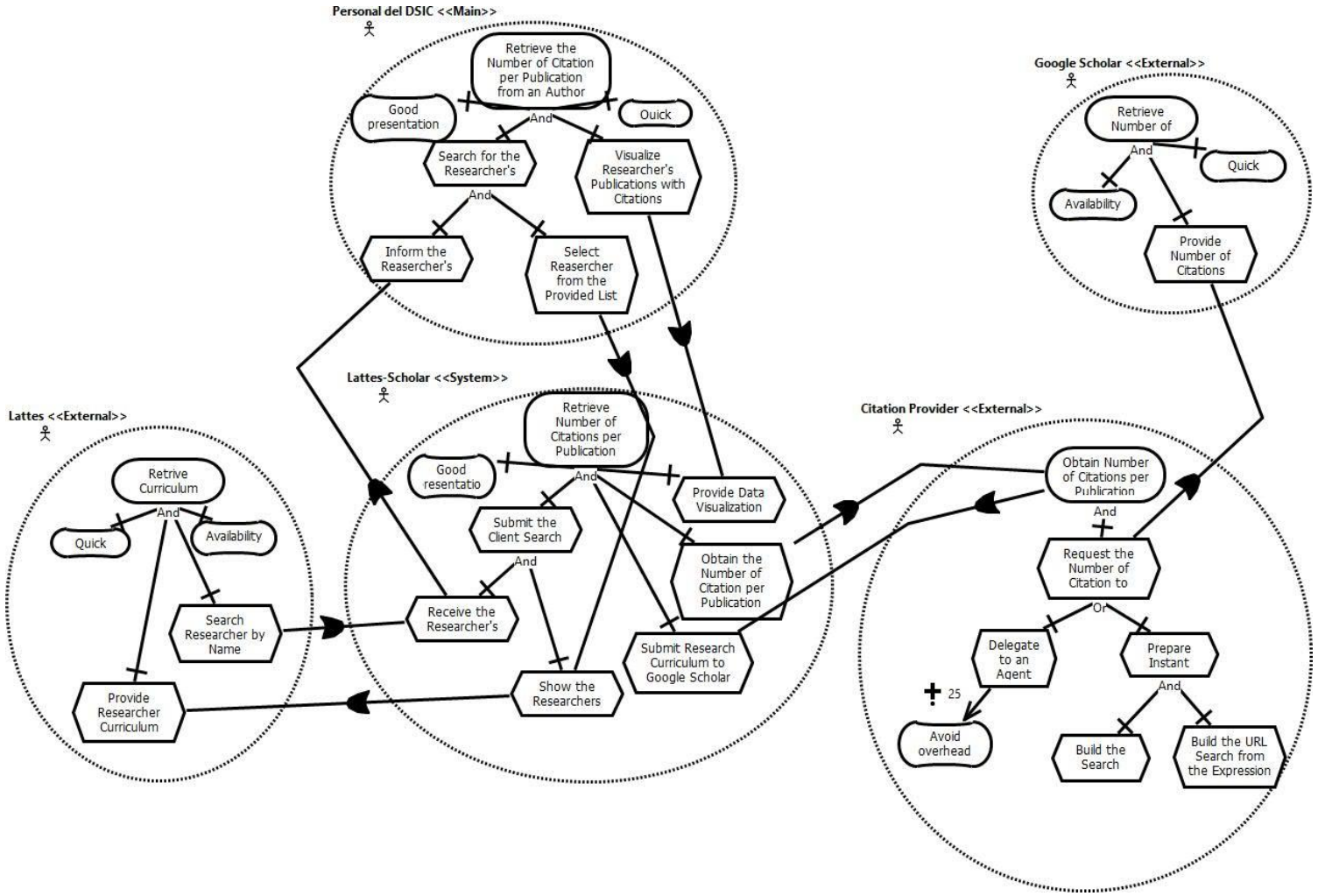


Figura 47*: Posible solución de Lattes Schoolar utilizando Value@GRL.*

10.3. Operación

Para la realización tanto del experimento como de la réplica se ha seleccionado un diseño inter-sujeto(*between-subjects***)** balanceado, de forma que cada participante use únicamente un método con un objeto experimental.

La forma en la cual se ha realizado el experimento es la que mostramos en la tabla siguiente.

Entrenamiento (120 min)	Introducción a Value@GRL	Introducción a i* (i-star)
Descanso (20 min)		
Caso de estudio (120 min)	Modelado	Modelado

Tabla 7: *Planificación del experimento.*

Una vez que los participantes hubiesen terminado de realizar el caso de estudio, se les proporcionaba una encuesta (véase el **Anexo E**), para obtener su opinión con respecto al método utilizado, así como su utilidad percibida, facilidad de uso percibida e intención de uso.

Los sujetos se han distribuido en cuatro grupos (donde cada grupo aplica un método y un objeto distinto) de forma aleatoria (ver **Tabla *8***). Este diseño también nos permite acomodar mejor la disponibilidad de los participantes. Además los diseños *within-subjects* (cuando los participantes aplican todos los tratamientos) han sido criticados y desaconsejados en SE [75].

	Sesión 1 (i*)	Sesión 2 (Value@GRL)
Objeto 1	Grupo 1	Grupo 3
Objeto 2	Grupo 2	Grupo 4

Tabla 8: *Distribución de los sujetos del experimento.*

Cabe mencionar que antes de realizar los experimento se realizó un experimento piloto con 4 estudiantes de doctorado para analizar la validez de la planificación del experimento. Dando como resultado varias mejoras sobre el material utilizado para los experimentos.

Se diseñaron múltiples documentos como instrumentación para el caso de estudio. El material de entrenamiento consiste en un conjunto de diapositivas que contiene la descripción del método, así como también un caso de estudio de entrenamiento, para enseñarles cómo utilizar el lenguaje de modelado explicado. Además, de los documentos correspondientes con los objetos experimentales. Debido a la subjetividad de ambos lenguajes de modelado, existen múltiples soluciones para el caso de estudio, por lo también se ha generado un documento para cada una de las posibles soluciones. Además, también se les proveyó de una encuesta, la cual fue hecha siguiendo una escala de tipo Likert, donde cada variable tenía un conjunto de preguntas, así como también variables abiertas en el cuestionario para poder obtener el *feedback* de los participantes en el mismo.

10.4. Recogida de datos

Una vez que los participantes en el experimento realizaran el caso de estudio, se procedió a recoger los datos del mismo.

Variables cuantitativas:

Las variables cuantitativas (calidad, tiempo de modelado y productividad), fueron recogidas por el caso de estudio. El tiempo de modelado, se medía apuntando la hora de inicio y fin de los participantes a la hora de realizar el experimento, mientras que la variable de calidad se medía comparando la solución realizada por el participante con cada una de las posibles soluciones del caso de estudio, y seleccionando aquella solución que tuviera mayor calidad. Una vez obtenidas las variables de calidad y tiempo de modelado, se procedió a calcular la productividad, como la división de la calidad entre el tiempo de modelado.

Variables cualitativas:

Las variables cualitativas (facilidad de uso percibida, utilidad percibida, e intención de uso) fueron recogidas mediante el empleo de una encuesta on-line utilizando un formulario de Google, el cual dispuso de un documento en formato Excel, con el que poder trabajar. Una vez recogidas se procedió a realizar una limpieza de los datos. Esta encuesta puede verse en el Anexo E.

10.5. Análisis de datos

Para el análisis de los datos recogidos tanto en el experimento como en la réplica se utilizaron las mismas técnicas para ambos, ya que los dos siguen un diseño **between-subjects** (inter-sujetos).

Para la realización del análisis de datos, tanto de las variables cuantitativas, como cualitativas se realizó un análisis descriptivo de las mismas obteniendo el mínimo, máximo, media y mediana para cada una de las variables. Una vez realizado el análisis descriptivo, se procedió a utilizar la técnica de ***Shapiro-Wilk***, técnica que permite **clasificar la distribución** de la variable.

Una vez se tuvo clasificada la distribución de los distintos métodos para cada una de las variables se procedió a utilizar una técnica, para determinar la existencia o no de una diferencia estadísticamente significativa entre las dos medias de los métodos de una variable. Para ello se utilizó la técnica de ***t-student*** en el caso de que ambos lenguajes e tuvieran una **distribución normal**, y la técnica de ***Mann-Whitney***, en el caso de que uno de los dos o los dos tuvieran una **distribución no normal**.

10.6. Resultados

10.6.1. Experimento

10.6.1.1. Variables basadas en desempeño

El análisis descriptivo de las distintas variables basadas en desempeño del experimento son las mostradas en la **Tabla 9**, y cuya representación gráfica corresponde con la **Figura 48**.

Variable	Método	Max	Min	Mean
Calidad	Value@GRL	83.07	42.52	58.54
	i*	54.32	19.37	36.60
Tiempo de modelado	Value@GRL	60.00	27.00	43.56
	i*	74.00	24.00	40.44
Productividad	Value@GRL	2.63	0.86	1.41
	i*	1.61	0.51	0.99

Tabla 9: *Análisis descriptivo de las variables basadas en desempeño del experimento.*

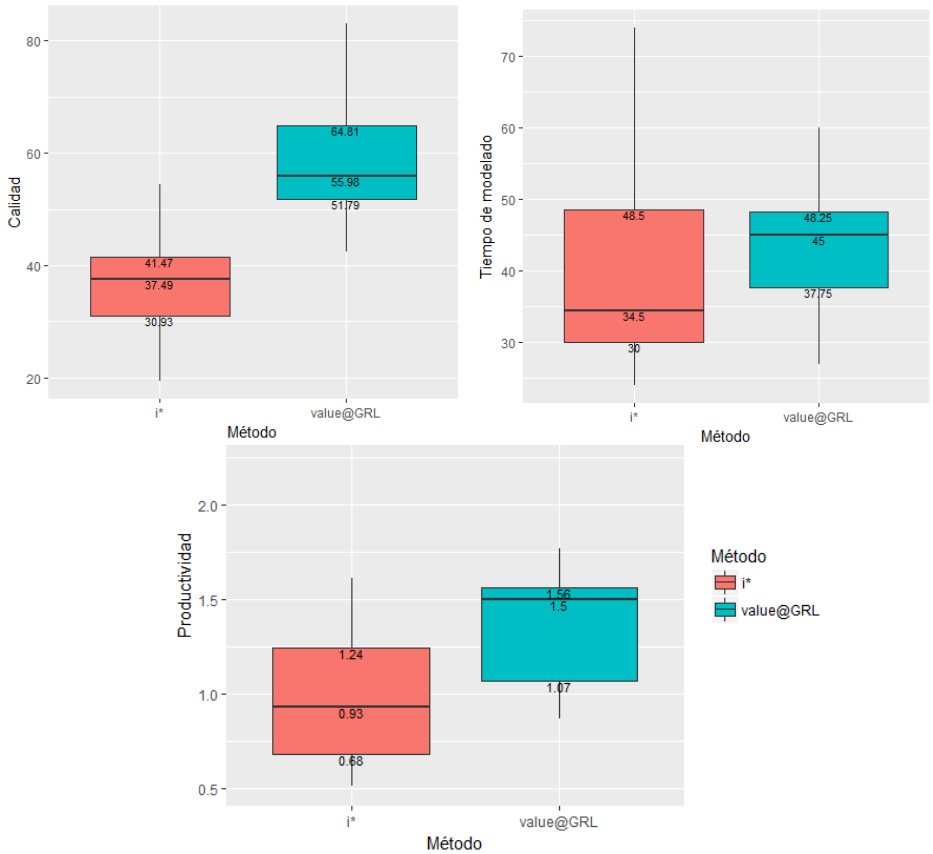


Figura 48:*Boxplot de las variables basadas en desempeño.*

El análisis descriptivo realizado sobre las variables basadas en desempeño muestra que Value@GRL tiene una media superior para todas las variables (calidad, tiempo de modelado y productividad) en comparación con los resultados obtenidos por i*. Pero no puede determinarse únicamente con este análisis descriptivo, si realmente existe una diferencia significativa entre las medias de los métodos, para ello es necesario realizar un análisis de distribución de las variables y aplicar una técnica para determinar si existe o no una diferencia significativa entre los valores. En la **Tabla 10**, se realiza este análisis.

Variable	Método	Distribución	Método	p-value
Calidad	Value@GRL	Normal	t-test	0.0005857
	i*	Normal		
Tiempo de modelado	Value@GRL	Normal	Mann-Withney	0.2902
	i*	No normal		
Productividad	Value@GRL	No normal	Mann-Withney	0.01893
	i*	No normal		

***Tabla 10**: Resumen del resultado de las pruebas para las variables basadas en desempeño.*

Los resultados obtenidos al realizar las pruebas estadísticas sobre las variables muestran que hay una diferencia significativa entre las medias de las variables de calidad y productividad, en cambio no hay una diferencia significativa para el tiempo de modelado (el p-value es mayor a 0,05).

Con estos resultados, puede concluirse que se aceptan las hipótesis $H1_1$ (los modelos generados con Value@GRL son de mayor calidad) y la hipótesis de $H3_1$ (los modelos generados con Value@GRL tienen mayor productividad), en cambio se rechaza la hipótesis$H2_1$ y se acepta la hipótesis nula $H2_0$ (los modelos generados con Value@GRL cuestan el mismo tiempo de modelar que aquellos hechos con i*). En la **Tabla 11** mostramos un resumen de los resultados de las hipótesis planteadas.

Variable	Hipótesis nula	Hipótesis alternativa
Calidad	Rechazar	Aceptar
Tiempo de modelado	Aceptar	Rechazar
Productividad	Rechazar	Aceptar

Tabla 11: *Resumen del resultado de las hipótesis de las variables de desempeño del experimento.*

10.6.1.2. Variables basadas en percepción

El análisis descriptivo de las distintas variables basadas en percepción del experimento son las mostradas en la **Tabla 12** y cuya representación gráfica se muestra en la **Figura 49**.

Variable	Método	Max	Min	Mean
Facilidad de uso percibida	Value@GRL	5.00	3	4.38
	i*	5.00	2	3.69
Utilidad percibida	Value@GRL	5.00	2	3.94
	i*	4.00	2	3.19
Intención de uso	Value@GRL	5.00	3	4.50
	i*	5.00	1	3.19

Tabla 12: *Análisis descriptivo de las variables basadas en percepción del experimento.*

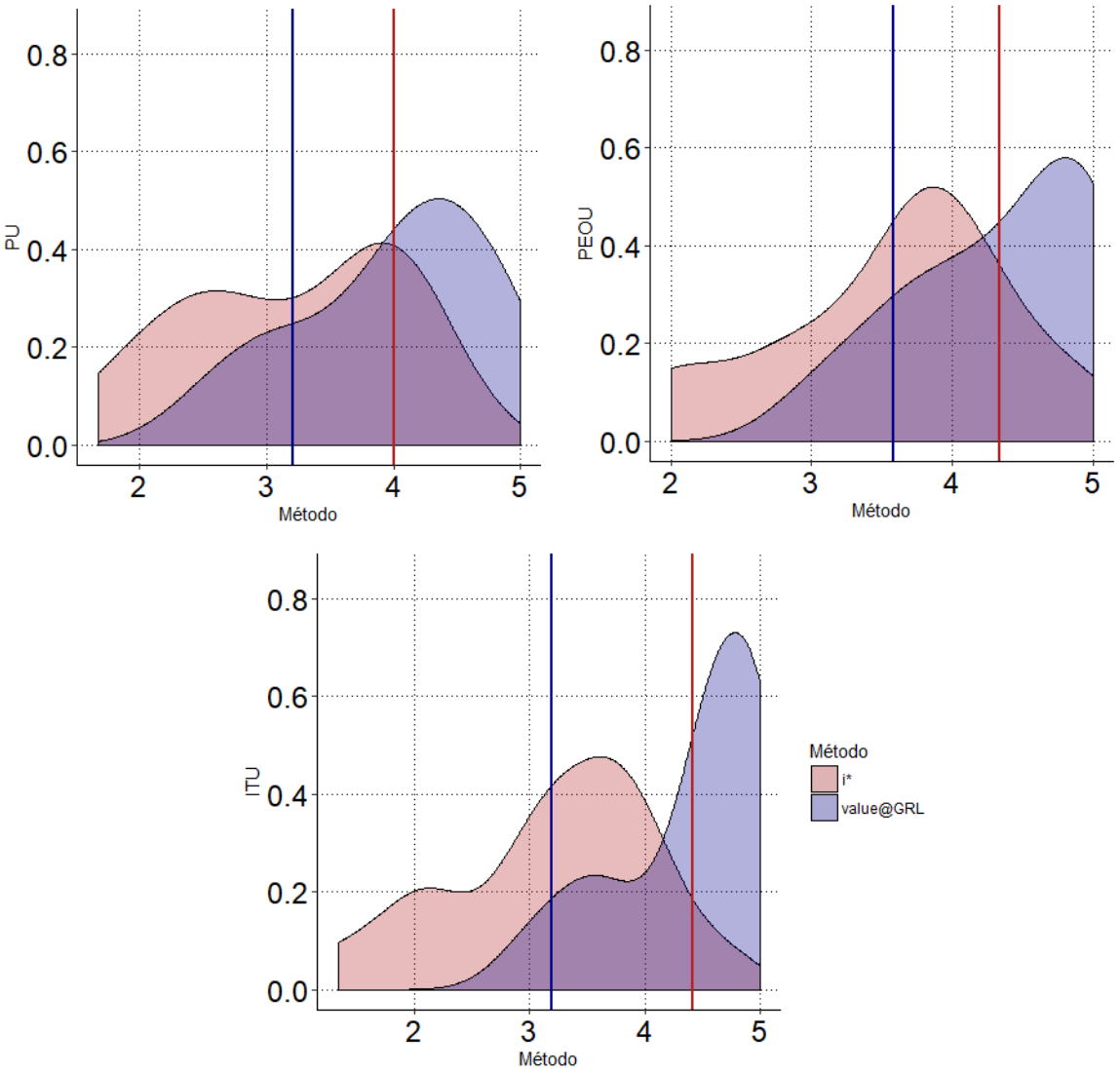


Figura 49: *Densidad de flujo de las variables basadas en percepción.*

El análisis descriptivo realizado sobre las variables basadas en percepción muestra que Value@GRL tiene una media superior para todas las variables (facilidad de uso percibido, utilidad percibida e intención de uso) en comparación con los resultados obtenidos por i*. Pero no puede determinarse únicamente con este análisis descriptivo, si realmente existe una diferencia significativa entre las medias de los métodos, para ello es necesario realizar un análisis de distribución de las variables y aplicar una técnica para determinar si existe o no una diferencia significativa entre los valores. En la **Tabla 13**, se realiza este análisis.

Variable	Método	Distribución	Método	p-value
Facilidad de uso percibida	Value@GRL	No normal	Mann-Withney	0.009549
	i*	No normal		
Utilidad percibida	Value@GRL	No normal	Mann-Withney	0.009556
	i*	No normal		
Intención de uso	Value@GRL	No normal	Mann-Withney	0.000378
	i*	Normal		

***Tabla 13**: Resumen del resultado de las pruebas para las variables basadas en percepción.*

Los resultados obtenidos al realizar las pruebas estadísticas sobre las variables muestran que hay una diferencia significativa entre las medias de las variables de facilidad de uso percibida, utilidad percibida e intención de uso, ya que todas las variables están por debajo del p-value de 0,05.

Con estos resultados, puede concluirse que se aceptan las hipótesis $H4_1$ (los modelos generados con Value@GRL tienen mayor facilidad de uso percibida), la hipótesis de $H5_1$ (los modelos generados con Value@GRL tienen mayor utilidad percibida y la hipótesis $H6_1$ (los modelos generados con Value@GRL). Un resumen de los resultados de las hipótesis de este experimento puede observarse en la **Tabla *14***.

Variable	Hipótesis nula	Hipótesis alternativa
Facilidad de uso percibida	Rechazar	Aceptar
Utilidad percibida	Rechazar	Aceptar
Intención de uso	Rechazar	Aceptar

Tabla 14: *Resumen del resultado de las hipótesis de las variables de desempeño del experimento.*

10.6.2. Replicación

10.6.2.1. Variables basadas en desempeño

El análisis descriptivo de las distintas variables basadas en desempeño del experimento son las mostradas en la **Tabla *15*** y cuya representación gráfica es la mostrada en la **Figura 50**.

Variable	Método	Max	Min	Mean
Calidad	Value@GRL	75.52	36.65	54.22
	i*	74.96	25.76	42.96
Tiempo de modelado	Value@GRL	92.00	31.00	64.31
	i*	75.00	32.00	54.56
Productividad	Value@GRL	1.64	0.48	0.88
	i*	1.48	0.36	0.86

Tabla 15: *Análisis descriptivo de las variables basadas en desempeño del experimento.*

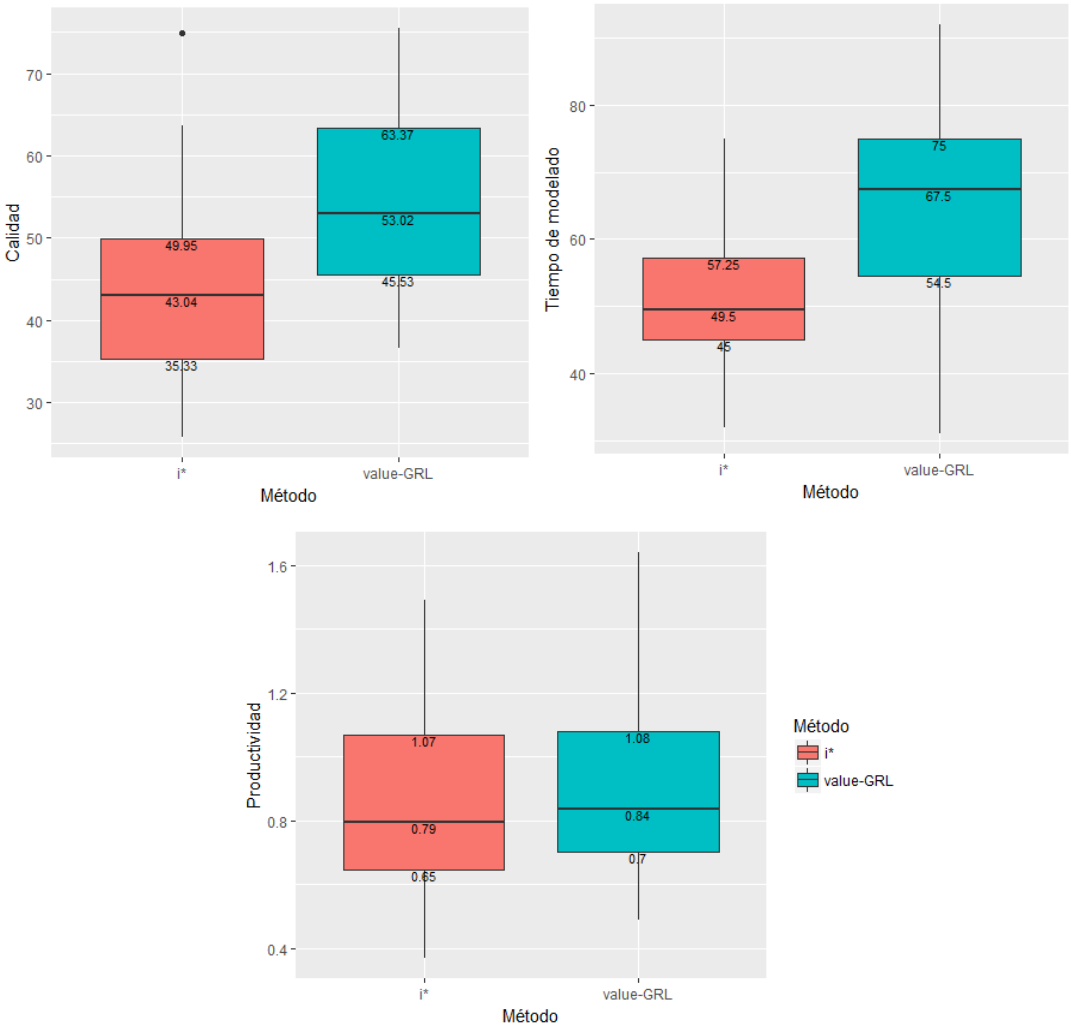


Figura 50: *Boxplot de las variables basadas en desempeño de la réplica.*

El análisis descriptivo realizado sobre las variables basadas en desempeño muestra que Value@GRL tiene una media superior para todas las variables (calidad, tiempo de modelado y productividad) en comparación con los resultados obtenidos por i*. Pero no puede determinarse únicamente con este análisis descriptivo, si realmente existe una diferencia significativa entre las medias de los métodos, para ello es necesario realizar un análisis de distribución de las variables y aplicar una técnica para determinar si existe o no una diferencia significativa entre los valores. En la **Tabla 16**, se realiza este análisis.

Variable	Método	Distribución	Método	p-value
Calidad	Value@GRL	Normal	t-test	0.0001348
	i*	Normal		
Tiempo de modelado	Value@GRL	Normal	t-test	0.000132
	i*	Normal		
Productividad	Value@GRL	No normal	Mann-Withney	0.6842
	i*	Normal		

Tabla 16: *Resumen del resultado de las pruebas para las variables basadas en desempeño.*

Los resultados obtenidos al realizar las pruebas estadísticas sobre las variables muestran que hay una diferencia significativa entre las medias de las variables de calidad y tiempo de modelado, en cambio no hay una diferencia significativa para la productividad (el p-value es mayor a 0,05).

Con estos resultados, puede concluirse que se aceptan las hipótesis $H1_1$ (los modelos generados con Value@GRL son de mayor calidad) y se rechazan las hipótesis $H2_1$ y $H3_1$ aceptando las hipótesis nulas de H20 (los modelos generados con Value@GRL tardan lo mismo o más que los generados con i*) y H30 (los modelos generados con Value@GRL tienen la misma o menos productividad que los generados con i*). Un resumen de los resultados de las hipótesis de esta réplica puede verse en la **Tabla *17***.

Variable	Hipótesis nula	Hipótesis alternativa
Calidad	Rechazar	Aceptar
Tiempo de modelado	Aceptar	Rechazar
Productividad	Aceptar	Rechazar

Tabla 17: *Resumen del resultado de las hipótesis de las variables basadas en desempeño de la réplica.*

Del análisis de los resultados de la replicación, cabe destacar que la productividad es calculada como el ratio de la calidad entre el tiempo de modelado, y que pese a que el modelo de Value@GRL tenga una buena calidad, no tiene un buen tiempo de modelado en esta réplica. Esto no es muy importante, ya que con un poco de práctica se puede disminuir bastante el tiempo de modelado necesario para realizar el modelo, pudiendo dar así lugar que haya una mayor productividad; es mucho más costoso aumentar la calidad del modelo que disminuir el tiempo necesario para realizar el modelo de objetivos.

10.6.2.1. Variables basadas en percepción

El análisis descriptivo de las distintas variables basadas en percepción del experimento son las mostradas en la **Tabla *18*** y cuya representación gráfica puede verse en la **Figura 51**.

Variable	Método	Max	Min	Mean
Facilidad de uso percibida	Value@GRL	5.00	2.00	3.86
	i*	5.00	2.00	3.55
Utilidad percibida	Value@GRL	5.00	2.16	3.90
	i*	5.00	2.16	3.52
Intención de uso	Value@GRL	5.00	2.00	3.86
	i*	5.00	1.00	3.31

Tabla 18: *Análisis descriptivo de las variables basadas en percepción de la réplica del experimento.*

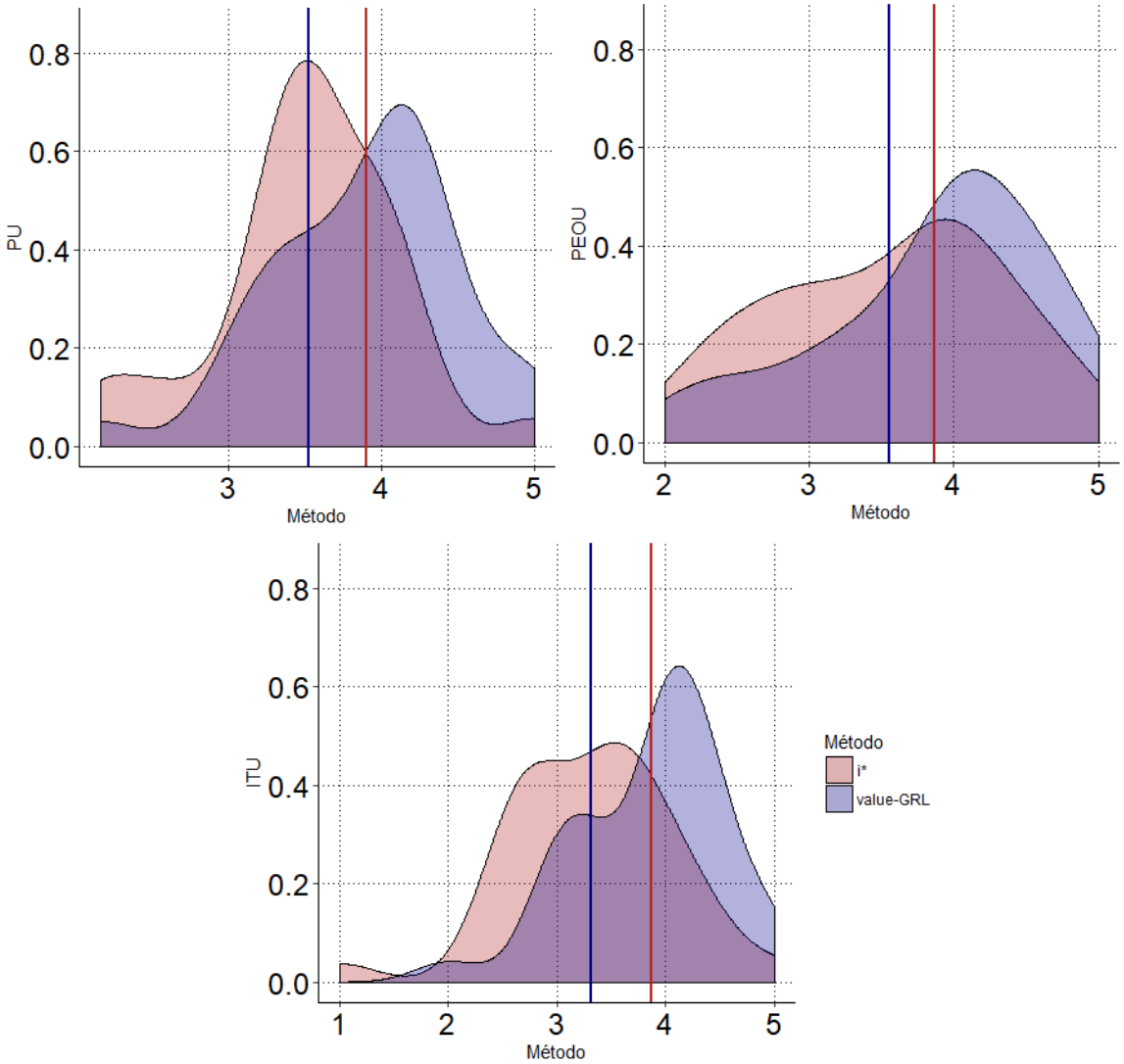


Figura 51: *Boxplot de las variables basadas en percepción de la réplica.*

El análisis descriptivo realizado sobre las variables basadas en percepción muestra que Value@GRL tiene una media superior para todas las variables (facilidad de uso percibido, utilidad percibida e intención de uso) en comparación con los resultados obtenidos por i*. Pero no puede determinarse únicamente con este análisis descriptivo, si realmente existe una diferencia significativa entre las medias de los métodos, para ello es necesario realizar un análisis de distribución de las variables y aplicar una técnica para determinar si existe o no una diferencia significativa entre los valores. En la **Tabla *19***, se realiza este análisis.

Variable	Método	Distribución	Método	p-value
Facilidad de uso percibida	Value@GRL	No normal	Mann-Withney	0.096160
	i*	Normal		
Utilidad percibida	Value@GRL	Normal	t-test	0.0167
	i*	Normal		
Intención de uso	Value@GRL	No normal	Mann-Withney	0.002166
	i*	Normal		

Tabla 19: *Resumen del resultado de las pruebas para las variables basadas en percepción de la réplica.*

Los resultados obtenidos al realizar las pruebas estadísticas sobre las variables muestran que hay una diferencia significativa entre las medias de las variables de utilidad percibida e intención de uso, ya que su p-value es menor a 0,05. En cambio, la variable de facilidad de uso percibida muestra que no hay una diferencia significativa ya que su p-value es mayor a 0,05

Con estos resultados, puede concluirse que se rechaza la hipótesis $H4_1$ aceptando su hipótesis nula $H4_0$ (los modelos generados con Value@GRL tienen menor o igual facilidad de uso percibida que los generados con i*) y se han aceptado las hipótesis alternativas $H5_1$ (los modelos generados con Value@GRL tienen mayor utilidad percibida y la hipótesis $H6_1$ (los modelos generados con Value@GRL). Un resumen de los resultados de las hipótesis de esta réplica puede observarse en la **Tabla *20***.

Variable	Hipótesis nula	Hipótesis alternativa
Facilidad de uso percibida	Aceptar	Rechazar
Utilidad percibida	Rechazar	Aceptar
Intención de uso	Rechazar	Aceptar

Tabla 20: *Resumen del resultado de las hipótesis de las variables basadas en percepción de la réplica.*

10.7. Análisis global

En la **Tabla 21** mostramos una comparación entre los distintos experimentos, donde mostramos para cada experimento la hipótesis seleccionada para cada variable.

Variable	Experimento		Réplica	
	HX0	HX1	HX0	HX1
Calidad		X		X
Tiempo de modelado	X		X	
Productividad		X	X	
Facilidad de uso percibida		X	X	
Utilidad Percibida		X		X
Intención de uso		X		X

Tabla 21: *Comparación de los resultados entre los distintos experimentos.*

Como puede observase en la tabla anterior, tanto en el experimento como en la réplica el lenguaje de Value@GRL ha obtenido mejores resultados que i-star para las variables de calidad, utilidad percibida e intención de uso.

Las variables de productividad y facilidad percibida han obtenido en el experimento que son mejores para el lenguaje de modelado Value@GRL, mientras que en la réplica se ha obtenido que ambas variables son iguales o mejores para i*.

Por último, la variable de tiempo de modelado, en el experimento ha dado que era igual para ambos métodos, mientras que en la réplica ha resultado que la media era significativamente distinta, teniendo i-star menor tiempo de modelado, esto puede haber afectado a la variable de productividad, ya que esta se calcula dividendo la calidad entre el tiempo de modelado.

En cuanto a las Preguntas de Investigación (PI) abordadas en este estudio, se ha podido llegar a las siguientes conclusiones con los resultados obtenidos del experimento y de la réplica del mismo.

- PI1: ¿Value@GRL permite a los modeladores obtener modelos de objetivos de mayor calidad que con i *?
  - Sí, para este contexto en concreto, con estos objetos experimentales y sujetos, se ha podido demostrar que estadísticamente los modelos generados con Value@GRL tienen una mayor calidad que aquellos generados con i*.

- PI2: ¿Value@GRL permite a los modeladores ser más productivos (obtener modelos de objetivos con mayor calidad en menos tiempo) que con i *?
    - Para este contexto en concreto, no puede demostrarse que, Value@GRL tenga mayor productividad que i*. Esto es debido a que en el experimento sí ha resultado que Value@GRL obtenga mayor productividad, en cambio en la réplica del mismo, no han sido los mismos resultados. Para poder comprobar con más fiabilidad si Value@GRL tiene una mayor productividad que i* sería recomendable realizar otra réplica del experimento.
- PI3: ¿Es Value@GRL percibido como más fácil de usar, útil, y es más probable que se utilice que i* para la especificación de modelos de objetivos?7
    - Para este contexto en concreto, ha podido demostrarse de forma estadística que Value@GRL es percibido de una forma más útil y es más probable que se utilice en vez que i* para la especificación de modelos de objetivos, en cambio no ha sido posible demostrar que Value@GRL tiene una mayor facilidad de uso percibida, ya que en el experimento ha dado unos resultados y en la réplica otros. Para poder comprobar con más fiabilidad si Value@GRL tiene una mayor facilidad de uso percibida que i* sería recomendable realizar otra réplica del experimento.

10.8. Amenaza a la validez

10.8.1. Amenazas internas

Las amenazas para la validación interna son el efecto de aprendizaje de los participantes, intercambio de información entre participantes y que los diseñadores del experimento influencien en él.

Con tal de evitar el efecto de aprendizaje en los participantes se han seleccionado dos objetos distintos y ambos lenguajes no han sido aplicados por los mismos participantes, los cuales no tenían conocimiento previo ni experiencia utilizando este tipo de lenguajes.

Para evitar el intercambio de información entre los participantes, se han utilizado dos objetos distintos, y monitorizando el experimento, además al tener dos sesiones de experimento al finalizar una se les pidió a los participantes que entregaran los materiales, para que así no pudieran influir en el otro grupo de participantes.

En cuanto a que los diseñadores del experimento influencien en el mismo, se ha intentado mitigar no dando información sobre la autoría de los lenguajes de modelado.

10.8.2. Amenazas externas

Las amenazas para la validación externa es la representatividad de los resultados, los cuales pueden afectados por los objetos utilizados en el experimento y el contexto de los participantes.

Para evitar esto, los objetos seleccionados pueden ser considerados pequeños proyectos, los cuales no son triviales.

En cuanto al contexto de los participantes, en el primer experimento, los participantes eran estudiantes de máster, los cuales carecían de conocimientos en el modelado de objetivos y tenían conocimientos en general en modelado, por lo cual pueden considerarse *juniors* debido a su falta de conocimiento en la industria, pero que son representativos como juniors debido a que en un futuro podrían utilizar uno de estos lenguajes de modelado; en el segundo experimento, los participantes eran estudiantes de grado los cuales carecían de conocimientos tanto en modelado de objetivos, como modelado en general, por lo que pueden considerarse como novatos, y en un futuro podrían utilizar estos lenguajes de modelado en la industria, siendo así muy representativos.

11. Conclusiones y trabajo futuro

11.1. Conclusiones

Los objetivos principales de este trabajo de fin de máster han sido los de **mejorar y extender el método** planteado en un trabajo previo, así como también el de **realizar una evaluación empírica sobre el lenguaje de modelado de valor desarrollado (Value@GRL)**. El método consiste en la utilización de un modelo de objetivos al que, en este trabajo, le hemos incluido el concepto de importancia que se utiliza para calcular valor de los elementos del modelo de objetivos. La utilización de este modelo de objetivos, enriquecido con valor, permite la generación automática de un modelo de procesos de negocio que es usado para definir una lista priorizada de las tareas que tienen que ser implementadas en el contexto de un desarrollo incremental. De esta forma, se ayuda a los desarrolladores a elegir en qué orden implementar las tareas que proporcionen el mayor valor posible para la organización.

Con respecto al primer objetivo de **mejorar y extender el método**, se han realizado **mejoras** sobre la propagación de la importancia para realizar el cálculo del valor, cambiando tanto la fórmula para propagar el valor, como el orden de aplicación de la misma, además de analizar y realizar pruebas sobre su uso, y del lenguaje de modelado desarrollado. En cuanto a la parte **de extensión del método**, se ha extendido la parte de transformación del modelo de valor al modelo de procesos de negocio, tanto definiendo una propuesta conceptual, como realizando un pequeño prototipo en ATL.

En cuanto al segundo objetivo, **realizar una evaluación empírica sobre el lenguaje de modelado de valor desarrollado**. Se han realizado un experimento controlado y dos réplicas. El experimento y una réplica en la UPV (Universitat Politècnica de Valencia) y otra réplica en la Universidad Nacional de Asunción (Paraguay), pero debido a la falta de tiempo no ha sido posible terminar de analizar los datos de la segunda réplica e incluirlos en este trabajo. El experimento ha dado como resultado que Value@GRL (lenguaje de valor desarrollado) tiene una mayor calidad, utilidad percibida e intención de uso que el lenguaje de modelado de objetivos de i*. Para la realización del experimento, no se ha utilizado la parte de valor del lenguaje de Value@GRL ya que el lenguaje i* no dispone de esta expresividad. Esto quiere decir que solo han sido comparados como lenguajes de modelado de objetivos. Tanto el lenguaje Value@GRL como i* tienen su origen en el lenguaje GRL, sin embargo, el número de elementos que pueden ser utilizados, así como las reglas que gobiernan su uso son distintas.

Como puede observarse los objetivos de este trabajo fueron abarcados en su totalidad y hay que destacar que a nivel académico se ha logrado un claro entendimiento del tema.

11.2. Trabajo Futuro

En cuanto al trabajo futuro podemos describirlo en varios ejes.

Por un lado, por parte del método, debería **implementarse la transformación del modelo de valor al modelo de procesos de negocio** y realizar distintas pruebas con varios ejemplos para determinar su completitud ya que en este trabajo solo hemos realizado un prototipo con un número limitado de ejemplos. Además, sería interesante poder añadir a estas transformaciones la posibilidad de **selección de alternativas**, ya que como se ha dicho anteriormente en la **sección 7.1**, hay múltiples alternativas de transformación para un mismo elemento, y nosotros hemos propuesto una forma, y nos gustaría que el desarrollador pudiera seleccionar qué alternativa le conviene más dependiendo de las condiciones y el contexto de uso. También sería útil incluir **trazabilidad explícita en las transformaciones** para saber de dónde viene cada uno de los elementos de las transformaciones y ayudar a los desarrolladores y expertos en procesos de negocio en la tarea de asociar procesos a objetivos. Esto último será especialmente útil al completar manualmente los procesos de negocio ya que se debería poder especificar qué objetivos se ven afectados.

Por otro lado, por parte de la evaluación empírica, habría **que terminar de realizar el análisis de los datos de la segunda réplica** del experimento (los datos de la Universidad Nacional de Asunción), y sería interesante **realizar una evaluación sobre el método en general** y no sólo sobre la parte de modelado.

11.3. Publicaciones

Durante el transcurso de este trabajo de fin de máster (TFM) se ha realizado una publicación en una conferencia internacional.

- Insfran E., Abrahão S., Pereira R., González F., Fernández M., **Cano C.**: "*Specifying Value in GRL for Guiding BPMN Activities Prioritization*", 26th International Conference on Information Systems Development. Larnaca, Cyprus (ISD 2017)
    - *Esta conferencia se encuentra catalogada en el Ranking de Conferencias CORE con la categoría A.*

Referencias

[1] T. Gorschek, C. Wohlin, P. Carre, and S. Larsson, “A Model for Technology Transfer in Practice”, *Ieee Softw.*, vol. 23, no. 6, pp. 88–95, 2006.

[2] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, "*Experimentation in software engineering: an introduction*", vol. 15. 2000.

[3] R. van Solingen, V. Basili, G. Caldiera, and H. D. Rombach, “Goal Question Metric (GQM) Approach”, in *Encyclopedia of Software Engineering*, 2002.

[4] J. Jiménez Gómez, “Método para la especificación de valor en procesos de negocio y la derivación incremental de servicios cloud”, Universitat Politècnica de València, 2016. - Trabajo Final de Máster (TFM).

[5] L. Barnett, “Agile Projects Must Measure Business Value”, *Agil. J.*, 2007.

[6] J. Patton, “Ambiguous business value harms software products”, *IEEE Softw.*, vol. 25, no. 1, pp. 50–51, 2008.

[7] R. Pettit, “Business Value Applied: Aligning The Day To Day With Business Imperative”, *Agil. J.*, 2007.

[8] D. Rawsthorne, “Managing the Work in an Agile Project”, *Haettu*, vol. 12, p. 2012, 2004.

[9] M. Poole, “Business and IT – A Marriage Made in Heaven?”, *Agile Journal*, 2007. [Online]. Available: http://www.agilejournal.com/content/view/627/76/.

[10] Object Management Group, “Value Delivery Metamodel (VDM),” 2015.

[11] J. Highsmith, “Determining Business Value”, 2013. [Online]. Available: http://jimhighsmith.com/determining-business-value/.

[12] R. S. Kaplan, “Conceptual Foundations of the Balanced Scorecard”, *Handbooks Manag. Account. Res.*, vol. 3, pp. 1253–1269, 2009.

[13] M. E. Porter, "*Competitive Advantage: Creating and sustaining superior performance.*", 1985.

[14] P. Drucker, "*The Practice of Management*, Reissue." Collins, 2006.

[15] B. W. Boehm, “Value-Based Software Engineering: Overview and Agenda” in *Value-Based Software Engineering*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 3–14.

[16] T. L. Saaty, "*The analytic hierarchy process: planning, priority setting, resources allocation.*", McGraw-Hill Inc, 1980.

[17] J. D. McKeen and H. Smith, “Making IT Happen: Critical Issues in IT Management”, *Wiley Chichester Hoboken NJ*, pp. 1–366, 2003.

[18] J. Karlsson and K. Ryan, “A cost-value approach for prioritizing requirements”, *IEEE Softw.*, vol. 14, no. 5, pp. 67–74, 1997.

[19] S. J. Bleistein, K. Cox, J. Verner, and K. T. Phalp, “B-SCP: A requirements analysis framework for validating strategic alignment of organizational IT based on strategy, context, and process”, *Inf. Softw. Technol.*, vol. 48, no. 9, pp. 846–868, 2006.

[20] K. Beck *et al.*, “Manifesto for Agile Software Development”, *The Agile Alliance*, 2001. [Online]. Available: http://agilemanifesto.org/.

[21] K. Beck, M. Beedle, and a Van Bennekum, “Principles behind the agile manifesto”, *Retrieved*, pp. 2–3, 2001.

[22] A. Rauf and M. AlGhafees, “Gap Analysis between State of Practice and State of Art Practices in Agile Software Development”, in *2015 Agile Conference*, 2015, pp. 102–106.

[23] Wikipedia, “Iterative and incremental development.” [Online]. Available: https://en.wikipedia.org/wiki/Iterative_and_incremental_development.

[24] Wikipedia, “Iterative design.” [Online]. Available: https://en.wikipedia.org/wiki/Iterative_design.

[25] Atlassian, “The product backlog: your ultimate to-do list.” [Online]. Available: https://www.atlassian.com/agile/backlogs.

[26] A.-W. Scheer and M. Nüttgens, “ARIS Architecture and Reference Models for Business Process Management”, *LNCS*, vol. 1806, pp. 366–379, 2000.

[27] Object Management Group, “Business Process Model And Notation (BPMN)”, 2008. [Online]. Available: http://www.omg.org/spec/BPMN/2.0/PDF/.

[28] International Telecommunication Union, “User Requirements Notation (URN) - Language requirements and framework”, 2011. [Online]. Available: http://www.itu.int/rec/T-REC-Z.150-201102-I/en.

[29] W. M. P. Van Der Aalst and A. H. M. Ter Hofstede, “YAWL: Yet another workflow language”, *Inf. Syst.*, vol. 30, no. 4, pp. 245–275, 2005.

[30] Organization for the Advancement of Structured Information Standards, “Web Services Business Process Execution Language Version 2.0”, 2007. [Online]. Available: http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html.

[31] R. Endl, G. Knolmayer, and M. Pfahrer, "*Modeling processes and workflows by business rules.*", Database Technology Research Group, 1998.

[32] G. Polančič, “The Popularity Of BPMN Just Keeps Rising” 2014. [Online]. Available: http://blog.goodelearning.com/bpmn/popularity-bpmn-rising/.

[33] Z. Racheva, M. Daneva, and K. Sikkel, “Value Creation by Agile Projects: Methodology or Mystery?”, Springer Berlin Heidelberg, 2009, pp. 141–155.

[34] International Telecommunication Union, “About ITU.” [Online]. Available: https://www.itu.int/en/about/Pages/default.aspx.

[35] E. Braun, D. Amyot, and T. C. Lethbridge, "*Generating Software Documentation in Use Case Maps from Filtered Execution Traces.*", Springer, 2015.

[36] R. B. Svensson *et al.*, “Prioritization of quality requirements: State of practice in eleven companies” in *Proceedings of the 2011 IEEE 19th International Requirements Engineering Conference, RE 2011*, 2011, pp. 69–78.

[37] P. Achimugu, A. Selamat, R. Ibrahim, and M. N. Mahrin, “A systematic literature review of software requirements prioritization research”, *Inf. Softw. Technol.*, vol. 56, no. 6, pp. 568–585, Jun. 2014.

[38] Z. Racheva, M. Daneva, K. Sikkel, A. Herrmann, and R. Wieringa, “Do We Know Enough about Requirements Prioritization in Agile Projects: Insights from a Case Study” in *2010 18th IEEE International Requirements Engineering Conference*, 2010, pp. 147–156.

[39] J. Highsmith, "*Agile Project Management: Creating Innovative Products"*, vol. 69. 2004.

[40] OBS Business School, “12 técnicas para la estimación de costes en proyectos.” [Online]. Available: http://www.obs-edu.com/blog-investigacion/project-management/12-tecnicas-para-la-estimacion-de-costes-en-proyectos/.

[41] PMI, *A Guide to the Project Management Body of Knowledge*, vol. 44, no. 3. 2013.

[42] Wolf Project, “Cómo estimar los costos de un proyecto.” [Online]. Available: http://wolfproject.es/como-estimar-los-costos-de-un-proyecto/.

[43] R. Popli and N. Chauhan, “Cost and effort estimation in agile software development” in *2014 International Conference on Reliability Optimization and Information Technology (ICROIT)*, 2014, pp. 57–61.

[44] M. Usman, E. Mendes, F. Weidt, and R. Britto, “Effort estimation in agile software development” in *Proceedings of the 10th International Conference on Predictive Models in Software Engineering - PROMISE ’14*, 2014, pp. 82–91.

[45] Wikipedia, “Planning poker.” [Online]. Available: https://en.wikipedia.org/wiki/Planning_poker.

[46] M. Brambilla, J. Cabot, and M. Wimmer, "*Model-Driven Software Engineering in Practice.*", Morgan & Claypool publishers 2012.

[47] T. Stahl and M. Voelter, "*Model-Driven Software Development: Technology, Engineering, Management*." 2006.

[48] E. Seidewitz, “What models mean”, *IEEE Softw.*, vol. 20, no. 5, pp. 26–32, Sep. 2003.

[49] J. Bezivin, O. Gerbe, I. C. Society, and S. Ieee Computer, "*Towards a precise definition of the OMG/MDA framework", 16th Annual International Conference on Automated Software Engineering*. 2001.

[50] A. Kleppe, J. Warmer, and W. Bast, "*MDA Explained: The Model Driven Architecture: Practice and Promise"*, vol. 83. 2003.

[51] Object Management Group, “MDA Guide Version 1.0.” [Online]. Available: http://www.omg.org/mda/mda_files/MDA_Guide_Version1-0.pdf.

[52] T. Kühne, “Matters of (meta-) modeling”, *Softw. Syst. Model.*, vol. 5, no. 4, pp. 369–385, 2006.

[53] Object Management Group, “Meta Object Facility (MOF) Version 2.5”, 2015. [Online]. Available: http://www.omg.org/spec/MOF/2.5/.

[54] Object Management Group, “Object Constraint Language (OCL), Version 2.4” 2014.

[55] F. Jouault, F. Allilaire, J. Bézivin, and I. Kurtev, “ATL: A model transformation tool”, *Sci. Comput. Program.*, vol. 72, no. 1–2, pp. 31–39, Jun. 2008.

[56] F. Jouault and I. Kurtev, “Transforming models with ATL”, *Lect. notes Comput. Sci.*, vol. 3844, p. 128, 2006.

[57] Object Management Group, “Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification”, 2007. [Online]. Available: http://www.lifl.fr/~dumoulin/enseign/pje/docs/QVT-07-07-07.pdf.

[58] S. Biffl, A. Aurum, B. Boehm, H. Erdogmus, and P. Grünbacher, Eds., "*Value-Based Software Engineering*.", Berlin, Heidelberg: Springer Berlin Heidelberg, 2006.

[59] J. M. Akkermans and J. Gordijn, “Value-based requirements engineering: exploring innovative e-commerce ideas”, *Requir. Eng.*, vol. 8, no. 2, pp. 114–134, 2003.

[60] M. Guzmán Carretero, “Especificación de requisitos para servicios cloud dirigido por valor”, Universitat Politècnica de València, 2015. Trabajo Final de Grado (TFG).

[61] D. Amyot, S. Ghanavati, J. Horkoff, G. Mussbacher, L. Peyton, and E. Yu, “Evaluating goal models within the goal-oriented requirement language”, *Int. J. Intell. Syst.*, vol. 25, no. 8, pp. 841–877, 2010.

[62] J. Azar, R. Smith, and D. Cordes, “Value-Oriented Requirements Prioritization in a Small Development Organization”, *IEEE Softw.*, vol. 24, no. 1, pp. 32–37, Jan. 2007.

[63] V. De Castro, E. Marcos, and J. M. Vara, “Applying CIM-to-PIM model transformations for the service-oriented development of information systems”, *Inf. Softw. Technol.*, vol. 53, no. 1, pp. 87–105, Jan. 2011.

[64] M. Valeria de Castro, “Aproximación MDA para el desarrollo orientado a servicios de sistemas de información web: del modelo de negocio al modelo de composición de servicios web”, Universidad Rey Juan Carlos, 2007.

[65] visual-paradigm, “Business Process to User Stories Mapping”, 2016. [Online]. Available: https://www.visual-paradigm.com/tutorials/business-process-to-user-stories-mapping.jsp. [Accessed: 14-May-2017].

[66] S. Glen, “Does IBM BPM = agile development?”, 2016. [Online]. Available: https://www.ibm.com/developerworks/bpm/library/techarticles/1602_glen-trs/1602_glen.html. [Accessed: 16-May-2017].

[67] SAP, "BPMN 2.0 Metamodel Implementation for Eclipse Get it and Use it." [Online]. Available: https://www.sap.com/documents/2015/09/a0eccfe3-567c-0010-82c7-eda71af511fa.html.

[68] E. S. K. Yu, "Towards modelling and reasoning support for early-phase requirements engineering", in *Proceedings of ISRE '97: 3rd IEEE International Symposium on Requirements Engineering*, 1997, pp. 226–235.

[69] W. B. Frakes and R. Baeza-Yates, "Information retrieval: data structures and algorithms", *Prentice Hall PTR*, 1992.

[70] S. Abrahão, C. Gravino, E. Insfran, G. Scanniello, and G. Tortora, "Assessing the effectiveness of sequence diagrams in the comprehension of functional requirements: Results from a family of five experiments", *IEEE Trans. Softw. Eng.*, vol. 39, no. 3, pp. 327–342, 2013.

[71] F. D. Davis, "Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology", *MIS Q.*, vol. 13, no. 3, pp. 319–340, 1989.

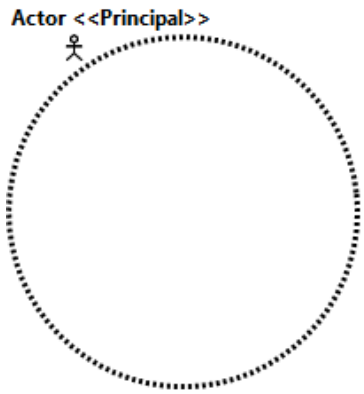
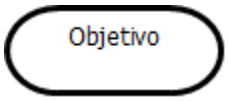
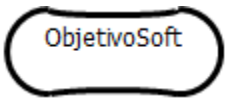
[72] D. Kundisch and T. John, "Business model representation incorporating real options: An extension of e3-value", in *Proceedings of the Annual Hawaii International Conference on System Sciences*, 2011, pp. 4456–4465.

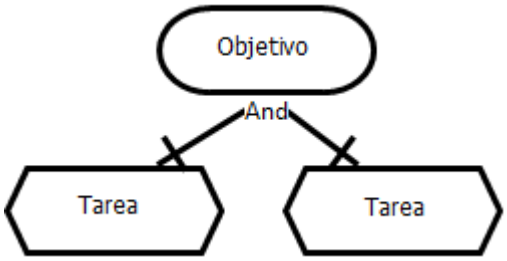
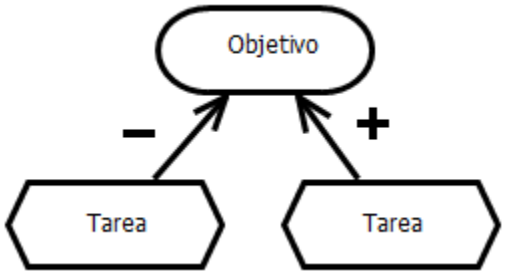
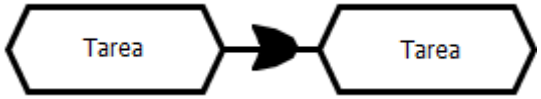
[73] H. Estrada, K. Najera, B. Vázquez, A. Martinez, J. C. Tellez, and J. J. Hierro, "Applying Tropos modeling for smart mobility applications based on the FIWARE platform", *CEUR Workshop Proc.*, vol. 1674, pp. 85–90, 2016.

[74] M. Serrano and J. C. S. D. P. Leite, "Development of agent-driven systems: From i*; architectural models to intentional agents' code", *CEUR Workshop Proc.*, vol. 766, no. iStar, pp. 55–60, 2011.

[75] B. Kitchenham, J. Fry, and S. Linkman, "The case against cross-over designs in software engineering", in *Proceedings - 11th Annual International Workshop on Software Technology and Engineering Practice, STEP 2003*, 2004, pp. 65–67.

Anexo A: Guía de Value@GRL

Elemento intencional	Descripción
	Actor: Representa una entidad activa (*stakeholder*, sistema…) que tiene intenciones y lleva a cabo acciones para conseguir sus objetivos ejerciendo su know-how. Tipos: • Principal: Representa el *stakeholder* en el que nos centramos. • Sistema: Representa al sistema que deseamos desarrollar. • Externo: Representa a un stakeholder en el cual no nos centramos. También puede ser un sistema externo.
	Objetivo: Condición o estado del mundo que un actor querría lograr.
	ObjetivoSoft: Objetivo ambiguo o cuyo criterio de logro es difícil de determinar.
	Tarea: Forma de hacer algo.

Enlaces	Descripción
	Descomposición: Permite descomponer un elemento en unidades más pequeñas y detalladas. Tipos: • And: Todos los elementos que los descomponen son necesarios. • Or: Posibles alternativas para realizar algo, pueden seleccionarse tantas como se deseen. • Xor: Solo una de todas las posibles alternativas pueden emplearse.
	Contribución: Representa un efecto de un elemento intencional sobre otro. Tipos: • Positivo: Afecta positivamente y se representa con un símbolo +. • Negativo: Afecta negativamente y se representa con un símbolo -.
	Dependencia: Representa que un elemento intencional necesita de otro para poder cumplirse. Este tipo de enlaces **sólo pueden hacerse entre actores**.

Anexo B: Guía de modelado de objetivos

- **Identificación de actores**: Se identifican los distintos participantes en la actividad de negocio, indicando además de qué tipo son (principal, sistema o externo).
- **Modelado de los elementos intencionales**: Se modelan los elementos intencionales de los actores de tipo principal y externo.
- **Modelado de los enlaces internos**: Se modelan los distintos enlaces existentes entre los elementos intencionales de un mismo actor. Puede modelarse descomposición o contribución.
- **Modelado del actor sistema**: Se modelan los elementos intencionales del actor sistema, así como los enlaces existentes entre ellos (descomposición y contribución).
- **Modelado de los enlaces externos**: Se modelan los enlaces existentes entre los distintos elementos intencionales de los actores. Puede modelarse dependencia y contribución.

Anexo C: Guía de asignación de la importancia y propagación del valor.

La definición del modelo de valor está compuesta por dos partes, por un lado, la de asignar la importancia a los elementos intencionales, por otro el de calcular en valor en base a la importancia.

Asignación de importancia

El stakeholder que representa el actor del modelo, debería ser quien asignara su importancia. Los posibles valores que pueden utilizarse son los siguientes:

- 0: Insignificante
- 25: Poco importante
- 50: Importante
- 75: Muy importante
- 100: Imprescindible

Además de asignar la importancia de los distintos elementos intencionales, también hay que asignar la importancia de las contribuciones, indicando cuánto contribuye un elemento a otro. Los posibles valores a asignar son los mismos que los mostrados anteriormente, aunque hay que tener en cuenta que, si la contribución es negativa, los valores deben ser negativos.

Cálculo del valor

$$Valor = IEimp + \sum_{i=0}^{n}\left(\frac{IEcontrib_i}{100} * Valor(IEimpDest_i)\right) + \sum_{i=0}^{n}\left(Valor(IEimpDest_i)\right) + \left(\frac{IEPadre}{\#IEHijos}\right)$$

Siendo:

- **IEimp**: La importancia o valor propio de cada elemento.
- **IEcontrib**: El valor del enlace de contribución, pudiendo ser este 0, 25, 50, 75 o 100.
- **IEimpDest**: El valor del destino, es decir, es el valor que tiene el elemento con el cual está relacionado.
- **IEPadre**: Es el valor o importancia que tiene el elemento padre en una descomposición

- **#IEHijos**: Cantidad de elementos hijos que tiene una descomposición.

La fórmula debe emplearse en el siguiente orden:

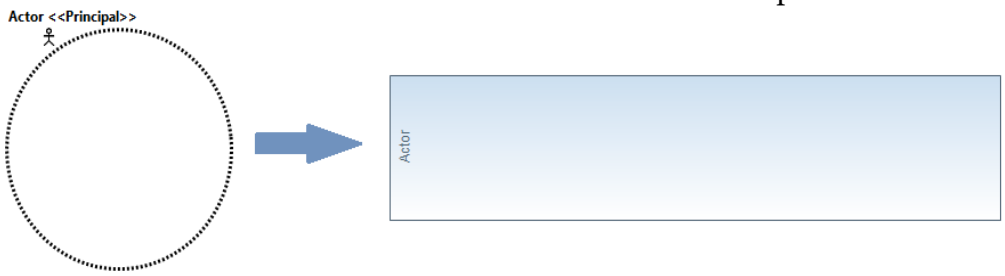
1. Calculamos el valor de los actores externos y principales
2. Calculamos el valor del actor sistema

   Nota: Hay que calcular el valor siempre de elementos padre a elementos a elementos hijo.

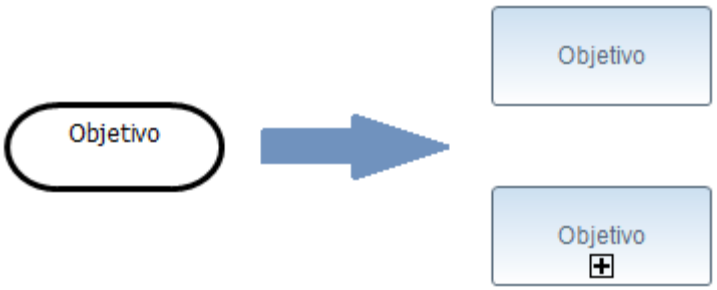
Anexo D: Guía de mapeo entre los elementos del modelo de objetivos y el modelo de procesos de negocio.

Los mapeos necesarios entre el modelo de valor y el modelo de procesos de negocio son los mostrados a continuación.

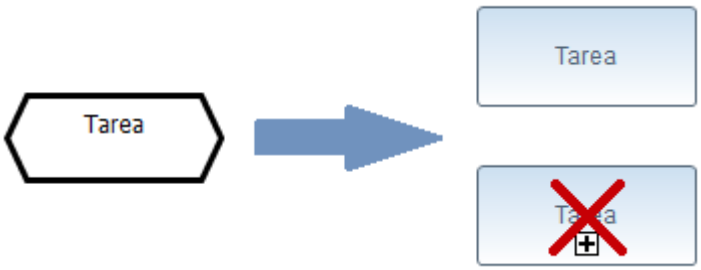
Un **actor** va a transformarse en una **calle** de un modelo de procesos.



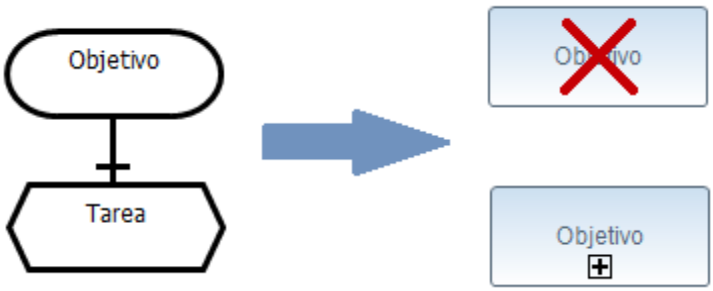
Un **objetivo** va a transformarse en un **subproceso** si está en un **actor principal o sistema** o en una **actividad** si está en **un actor externo**.



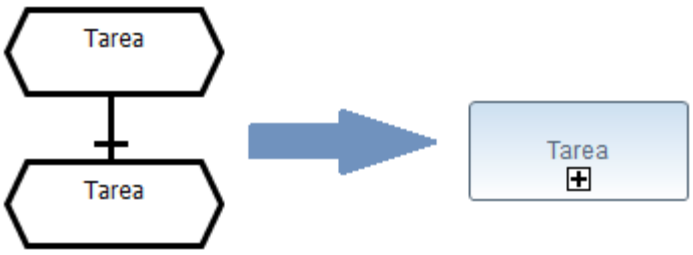
Una **tarea** va a transformarse en una **actividad**.



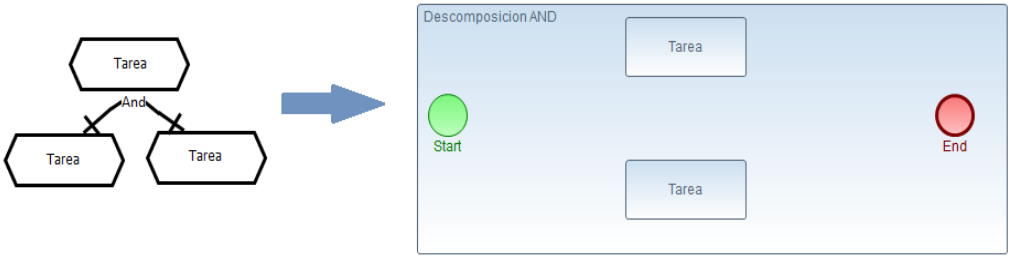
Un **objetivo con descomposición** va a transformase en un **subproceso**.



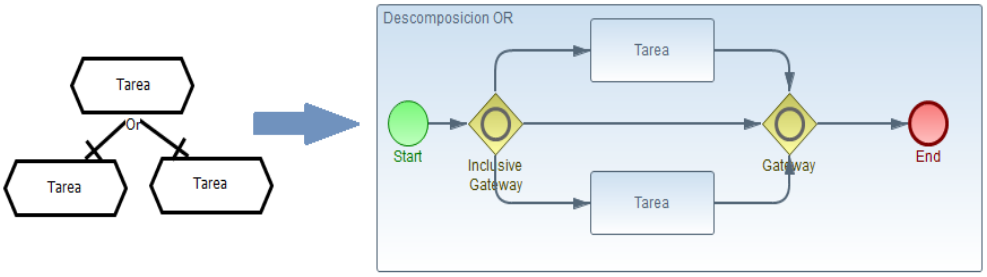
Una **tarea con descomposición** va a transformarse en un **subproceso**.



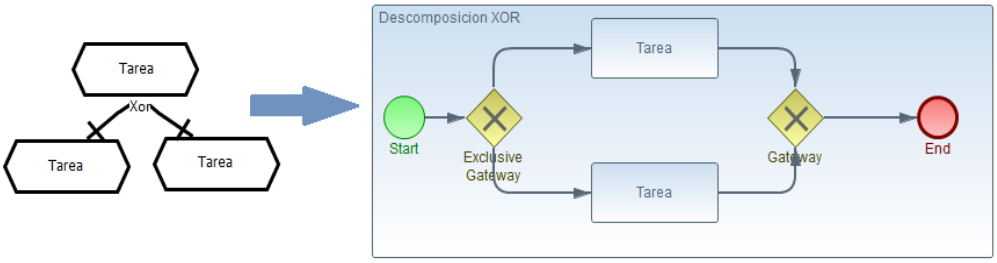
Una **descomposición de tipo AND** va a generar un **start**, y **end event**, y **las tareas** que la componen van a estar dentro del subproceso pero **sin un flujo** concreto.



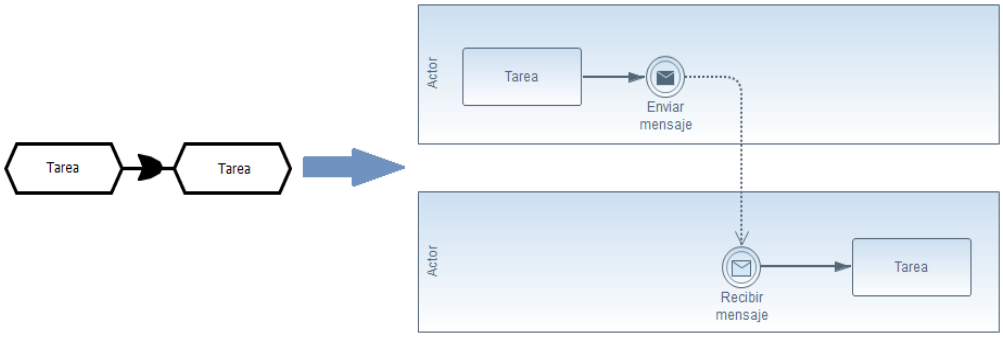
Una **descomposición de tipo OR** va a generar un **start** y **end event**, además de **dos puertas inclusivas** que deberán tener un **flujo de control con las tareas** que componen la descomposición.



Una **descomposición de tipo XOR** va a generar un **start** y **end event**, además de **dos puertas exclusivas** que deberán tener un **flujo de control con las tareas** que componen la descomposición.



Una **dependencia entre actores** generará un **intermediateThrowEvent de tipo mensaje** y un **intermediateCatchEvent de tipo mensaje**, así como los **flujos de control** entre los mismos.



Anexo E: Encuesta

Las preguntas realizadas en esta encuesta para poder medir las variables de utilidad percibida, facilidad de uso percibida e intención de uso son las mostradas a continuación.

Encuentro el procedimiento para aplicar el método de modelado de valor complejo y difícil de seguir	1 ○	2 ○	3 ○	4 ○	5 ○	Encuentro el procedimiento para aplicar el método de modelado de valor simple y fácil de seguir
Creo que este método de modelado de valor reduciría el tiempo necesario para especificar modelos de valor de negocio	1 ○	2 ○	3 ○	4 ○	5 ○	Creo que este método de modelado de valor aumentaría el tiempo necesario para especificar modelos de valor de negocio
Creo que sería fácil para mí conseguir ser hábil en el uso de este método de modelado de valor	1 ○	2 ○	3 ○	4 ○	5 ○	Creo que sería difícil para mí conseguir ser hábil en el uso de este método de modelado de valor
Encuentro el método de modelado de valor difícil de aprender	1 ○	2 ○	3 ○	4 ○	5 ○	Encuentro el método de modelado de valor fácil de aprender
Pretendería usar este método de modelado de valor en el futuro, si tengo que especificar modelos de valor de negocio	1 ○	2 ○	3 ○	4 ○	5 ○	NO pretendería usar este método de modelado de valor en el futuro, si tengo que especificar modelos de valor de negocio

Creo que las especificaciones obtenidas con este método son desordenadas, confusas, vagas y ambiguas	1 ○	2 ○	3 ○	4 ○	5 ○	Creo que las especificaciones obtenidas con este método son claras, concisas y no ambiguas.
Creo que este método de modelado de valor NO TIENE suficiente expresividad para especificar los elementos de valor y sus relaciones de impacto	1 ○	2 ○	3 ○	4 ○	5 ○	Creo que este método de modelado de valor tiene suficiente expresividad para especificar los elementos de valor y sus relaciones de impacto
Creo que este método de modelado de valor NO provee un medio efectivo para especificar modelos de valor de negocio	1 ○	2 ○	3 ○	4 ○	5 ○	Creo que este método de modelado de valor provee un medio efectivo para especificar modelos de valor de negocio
En general, encuentro el método de modelado de valor útil	1 ○	2 ○	3 ○	4 ○	5 ○	En general, NO encuentro el método de modelado de valor útil
Usando este método de modelado de valor mejoraría mi rendimiento en la especificación de requisitos debido a un mejor conocimiento de los objetivos del negocio	1 ○	2 ○	3 ○	4 ○	5 ○	Usando este método de modelado de valor NO mejoraría mi rendimiento en la especificación de requisitos debido a un mejor conocimiento de los objetivos del negocio

En general, encuentro este método de modelado de valor difícil de usar	1 ○	2 ○	3 ○	4 ○	5 ○	En general, encuentro este método de modelado de valor fácil de usar
NO recomendaría el uso de este método de modelado de valor	1 ○	2 ○	3 ○	4 ○	5 ○	Recomendaría el uso de este método de modelado de valor

Además de todas estas preguntas realizadas una escala Likert, también se añadieron tres preguntas de respuesta abierta, para saber la opinión de los participantes con respecto los lenguajes de modelado, así como también ayudar a saber por qué opinan así del lenguaje.

Las preguntas de respuesta abierta son:

1. ¿Tiene alguna sugerencia para hacer este método de modelado de valor más fácil de usar?

2. ¿Tiene alguna sugerencia para hacer que este método de modelado de valor se más útil para representar el valor y sus relaciones de impacto entre los distintos actores del negocio?

3. ¿Cuáles podrían ser las razones por qué o por qué no estaría interesado en usar este método en el futuro?