Una propuesta orientada a objetivos para el análisis de requisitos en RIAs

José Alfonso Aguilar[1,3], Irene Garrigós[1], Sven Casteleyn[2], Jose-Norberto Mazón[1]

[1]Grupo de Investigación Lucentia, Departamento de Lenguajes y Sistemas Informáticos
Universidad de Alicante, España
[2]Universitat Politècnica de València, España
[3]Facultad de Informática
Universad Autónoma de Sinaloa, México
{ja.aguilar,igarrigos,jnmazon}@dlsi.ua.es
sven.casteleyn@upv.es

Resumen. Actualmente, existen diversas metodologías que brindan soporte al desarrollo de RIAs haciendo énfasis en la etapa de desarrollo e implementación. Sin embargo, no consideran una etapa de análisis de requisitos específica para RIAs que resulta necesaria para poder determinar en etapas tempranas de desarrollo qué funcionalidades deben ser especificadas en el cliente dependiendo de los requisitos no funcionales que se quiera maximizar. Con la finalidad de paliar esta limitación, en este trabajo se presenta una propuesta orientada a objetivos para el análisis y modelado de requisitos en RIAs. La propuesta incluye la definición de un algoritmo basado en el óptimo de Pareto para evaluar y seleccionar las configuraciones cliente/servidor deseables mediante la maximización de requisitos no-funcionales. Finalmente, ilustramos nuestra propuesta mediante un caso de estudio real de una empresa de bioinformática.

Keywords: Ingeniería Web, Rich Internet Applications, Ingeniería de Requisitos.

1. Introducción

La fase de especificación y análisis de requisitos es uno de los factores de éxito más importantes en el desarrollo de *software*, por tal motivo es una de las fases más complicadas de conducir en un proceso de desarrollo. En el campo de la Ingeniería Web es particularmente difícil debido a la evolución constante en las tecnologías de implementación y a que la Web es utilizada por una audiencia heterogénea. En este sentido, un factor clave en el éxito de una aplicación Web consiste en asegurar que los requisitos y los objetivos del usuario sean cumplidos en su totalidad, de esta forma es posible contribuir, entre otras cosas, a mejorar la experiencia de navegación del usuario [19].

Los requisitos funcionales describen la funcionalidad de la aplicación Web, por ejemplo "generar reporte de ventas", mientras que los requisitos no-funcionales

son restricciones en el proceso de desarrollo o en la aplicación Web, tales como el "entorno de desarrollo" y la "usabilidad', entre otros. Una definición correcta de los requisitos mejora la calidad de la aplicación a desarrollar. Desafortunadamente, los requisitos no son tomados en cuenta lo suficiente en la práctica profesional así como en las metodologías para el desarrollo de aplicaciones Web [3,22]. Incluso, en la mayoría de las metodologías solo se realiza el análisis de requisitos funcionales, ignorando por completo a los requisitos no-funcionales hasta la fase de implementación. Este problema afecta directamente al cliente y al usuario final[1]: el cliente no obtiene un producto que cumpla por completo con sus expectativas y el usuario final no puede satisfacer sus necesidades con la aplicación Web. Por lo tanto, los requisitos no-funcionales deben ser analizados desde las fases iniciales del proceso de desarrollo con el fin de mejorar la calidad de la aplicación Web.

Por otro lado, las RIAs (Rich Internet Applications) nacen como una evolución natural de la Web, este tipo de aplicaciones hace frente a la limitación impuesta por la tecnología tradicional de la Web 1.0 [8] al ofrecer una serie de características especiales, por ejemplo interfaces más atractivas (ricas) y una mejor capacidad de respuesta e interactividad con el usuario (similar a la de aplicaciones de escritorio). A pesar de las posibilidades que ofrecen, las RIAs son más difíciles de diseñar e implementar que las aplicaciones Web 1.0. Por tal motivo, las metodologías de desarrollo han sido objeto de mejoras para brindar soporte a las características particulares de las RIAs, pero se han enfocado principalmente en cuestiones de presentación [18] y en capacidades de interacción [23] descuidando la etapa de análisis y especificación de requisitos. Esto es una desventaja muy importante a considerar, debido a que el diseñador tiene que lidiar con los requisitos de la Web 1.0, como lo es la navegación más los requisitos específicos de las RIAs, tales como la capacidad de respuesta o la reducción del ancho de banda. Asimismo, el diseñador debe de considerar la distribución de los requisitos funcionales, es decir, si son implementados en el cliente o en el servidor y como es que la distribución afecta el cumplimiento de los requisitos no-funcionales. Por lo tanto, elegir dónde serán implementados los requisitos funcionales (cliente o servidor) es una decisión fundamental para el correcto desempeño de la RIA.

En nuestro trabajo previo [13,20], se presentó una propuesta para el análisis y especificación de requisitos en la Web 1.0 dónde se adaptó el *framework i** con la taxonomía propuesta en [11] para ser utilizado en Web. En este artículo se extiende la propuesta para brindar soporte para el análisis y especificación de requisitos en RIAs. Concretamente, nos centramos en la distribución de los requisitos funcionales entre el servidor y el cliente a partir de la maximización balanceada de los requisitos no-funcionales (*softgoals*). Para lograrlo, presentamos un algoritmo basado en el óptimo de Pareto, el cual es utilizado cuando existen múltiples objetivos en conflicto que necesitan ser balanceados. El algoritmo

[1] Usuario final no es necesariamente sinónimo de cliente. Una compañía puede ser un comprador de *software*, pero el usuario final puede ser solo un empleado o grupo de empleados dentro de la compañía.

ofrece un conjunto de soluciones optimizadas en base a las contribuciones de los requisitos a las *softgoals*, así, el diseñador podrá seleccionar la solución a implementar priorizando las *softgoals*. Para ejemplificar la aplicabilidad de nuestra propuesta se presenta un caso de estudio real de una empresa de bioinformática.

El trabajo esta estructurado en las siguientes secciones: la Sección 2 describe las metodologías Web que han sido adaptadas para soportar RIAs así como las que han surgido especificamente para su desarrollo. La Sección 3, presenta un resumen de nuestra propuesta para el análisis de requisitos Web 1.0. En la Sección 4 se describe nuestra propuesta para el análisis de requisitos en RIA. El caso de estudio es detallado en la Sección 5. Finalmente, las conclusiones son presentadas en la Sección 6.

2. Trabajo Relacionado

La investigación en el campo de la Ingeniería Web ha demostrado que el desarrollo de RIAs es un reto difícil que requiere ampliar las metodologías Web existentes [27] para mejorar el proceso de desarrollo. Esto permitirá adaptar los modelos conceptuales para dar soporte a las características interactivas de las interfaces de usuario de las RIAs. A continuación se presentan algunos métodos que han sido modificados para brindar soporte al desarrollo de RIAs así como los que han surgido específicamente para su desarrollo.

UML-based Web Engineering (UWE) [16] es una metodología para el desarrollo de aplicaciones Web basado en UML (*Unified Modeling Language*) y MDA (*Model-Driven Architecture*) [15]. UWE ha sido objeto de mejoras para soportar el desarrollo de RIAs, a raíz de estas mejoras han surgido extensiones tales como (i) UWE-Patterns, utilizada para la definición de patrones que especifiquen el comportamiento de las RIAs [17], (ii) UWE-R, ofrece nuevos componentes de modelado para cubrir aspectos como la navegación y presentación para el desarrollo de las RIAs [21] y (iii) UWE-RUX, combinación de UWE y RUX-Model [18] para el diseño de interfaces de usuario (UIs) en RIAs [28]. UWE modela los requisitos de la aplicación Web mediante diagramas de casos de uso con descripciones textuales y diagramas de actividades.

Web Modeling Language (WebML)[6] es un lenguaje visual para el modelado de aplicaciones Web. WebML dispone de una extensión para soportar el desarrollo de RIAs [5,4]. Para lograr este soporte, WebML introduce nuevos elementos de modelado en su modelo de hipertexto, con lo que habilita su ejecución de lado del cliente. Además, WebML ha sido combinado con RUX-Model [18] para soportar el diseño de RIAs así como generación automática de código [28]. Los requisitos son especificados utilizando casos de uso y diagramas de actividades.

Object-Oriented Approach for Web Solutions Modeling (OOWS) es un método orientado a objetos para el desarrollo de aplicaciones Web [26]. En lo que respecta al soporte para RIAs, OOWS ha definido un metamodelo genérico para el desarrollo de UIs para RIAs [33]. Este metamodelo soporta la definición de *widgets* así como los eventos de la UI accionados por el usuario.

En [34], los autores proponen un proceso de generación de código para obtener RIAs. La fase de requisitos es realizada por medio de una serie de estrategias que implementan (i) FRT (*Function Refinement Tree)*, (ii) casos de uso y (iii) una serie de diagramas de tareas, especificación de tareas y descripción de datos.

Object-Oriented Hypermedia Design Method (OOHDM) [23] es un método para el desarrollo de aplicaciones Web. Para el diseño de RIAs OOHDM dispone de una extensión que utiliza el modelo ADV(*Abstract Data View*) para especificar la estructura y comportamiento de las interfaces RIA [32]. Las técnicas que utiliza OOHDM para la especificación de requisitos son: casos de uso, diagramas de interacción y reglas en lenguaje natural para los esquemas de navegación.

Internet Application Modelling Language (IAML) [36] es un nuevo lenguaje para el modelado de RIAs. IAML utiliza conceptos propios de RIAs como elementos de primer orden basado en su trabajo previo sobre los requisitos básicos de las RIAs [35]. IAML provee una herramienta de soporte para el desarrollo de RIAs basada en EMF (*Eclipse Modeling Framework*) y GMF (*Graphical Modeling Framework*) [36].

The RUX-Model [29] es una extensión para las aplicaciones Web basadas en HTML (*HyperText Markup Language*) que permite el diseño de UIs para las RIAs (soporte multimedia). No es propiamente una metodología debido a que necesita combinarse con otras.

OOH4RIA [23] es una extensión del método OOH [14] que define un proceso dirigido por modelos para obtener aplicaciones RIA. Específicamente, OOH4RIA permite obtener aplicaciones para el *framework* GWT (*Google Web Toolkit*). OOH4RIA dispone de soporte en herramienta mediante OOH4RIA-Tool [25,24].

Abstract Design of Rich Internet Applications (ADRIA) [10] es una aproximación basada en UML y en el análisis orientado a objetos para diseñar RIAs. Principalmente, utiliza casos de uso como modelos de tareas y espacios de interacción como mecanismos para el diseño de eventos accionados por la interacción del usuario con la aplicación Web [9].

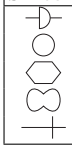
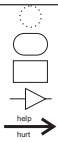
En la actualidad se han propuesto nuevos métodos para el diseño de RIAs (ADRIA, IAML), también existe una amplia gama de métodos desarrollados por la comunidad académica para modelar aplicaciones Web, de los cuales, algunos de ellos se han extendido para soportar el modelado de RIAs (UWE, OOWS, WebML, OOHDM). Lamentablemente, algunos métodos no consideran una fase de Ingeniería de Requisitos (OOH4RIA, IAML, RUX), otros utilizan técnicas clásicas que no están específicamente adaptadas para soportar el diseño de RIAs. Nuestra enfoque complementa estas técnicas y permite centrarse en la satisfacción los objetivos del usuario considerando las características específicas de las RIAs, en particular, la distribución cliente/servidor.

3. Análisis de Requisitos Orientado a Objetivos en Ingeniería Web

En este apartado se describe de forma breve nuestra propuesta para el análisis y especificación de requisitos en Ingeniería Web por medio del *framework i** [13,2]. El *framework i** [37] es una técnica de modelado orientado a objetivos (goal-oriented), su enfoque principal consiste en analizar y modelar explícitamente las relaciones entre múltiples *stakeholders*[2] (actores en la notación *i**). Además, es uno de los más utilizados para analizar los objetivos de los actores y cómo el sistema a diseñar debería satisfacerlos, ha sido probado útilmente para representar las intensiones de los *stakeholders* (motivaciones y objetivos), las dependencias entre los *stakeholders* para alcanzar sus objetivos así como los efectos positivos o negativos de los objetivos en cada *stakeholder*. De esta forma, es posible seleccionar alternativas de diseño para la aplicación a desarrollar y con ello maximizar el cumplimiento de los objetivos de los *stakeholders*.

El *framework i** consiste en dos modelos: SD (Strategic Dependency), utilizado para describir las relaciones de dependencia entre varios actores en un contexto organizacional y el modelo SR (Strategic Relational), empleado para describir los intereses del actor y como podrían abordarse. El *framework i** lo constituyen una serie de elementos entre los que se encuentran los elementos intencionales, formados por objetivos (*Goals*), tareas (*Tasks*), recursos (*Resources*) y *Softgoals*. Finalmente, las relaciones intencionales, las cuales son *links* del tipo *Means-end*, encargados de representar formas alternativas para satisfacer objetivos, los *Task decomposition links*, quienes representan los elementos necesarios para que una tarea sea realizada y los *Contribution links*, los cuales pueden ser del tipo (*Help, Some+, Hurt, Some-, Make, Break y Unknown*) y sirven para modelar cómo es que un elemento intencional contribuye a la satisfacción de una *softgoal*. La Tabla 1 muestra la notación utilizada para modelar en *i**.

Tabla 1: La notación de *i**.

Símbolo	Nombre	Símbolo	Nombre
	SD Model		SR Model
	Actor		Goal
	Task		Resource
	Softgoal		Means-end link
	Task-decomposition link	help hurt	Contribution link

Como explicamos en [13], nuestra propuesta utiliza la taxonomía de requisitos Web presentada en [11], compuesta por requisitos de contenido (*Content*), servicio (*Service*), navegación (*Navigational*), interfaz (*Layout*), personalización (*Personalization*). Para poder utilizar la taxonomía y el *framework i** para el análisis de requisitos en Ingeniería Web se ha implementado un Metamodelo ECORE en EMF (*Eclipse Modeling Framework*). Los requisitos no funcionales se modelan directamente utilizando el elemento (*softgoals*) de *i**.

[2] Las personas u organizaciones que afectan o son afectados directa o indirectamente por el proyecto de desarrollo en una forma positiva o negativa[30].

Cabe destacar que la propuesta brinda soporte para la derivación automática de modelos conceptuales a partir del modelo de requisitos por medio de una serie de reglas transformación [1]. Por último, la propuesta ha sido implementada en el contexto de la metodología A-OOH *(Adaptive Object Oriented Hypermedia method)* [12], pero puede ser aplicada a cualquier otro método de Ingeniería Web.

4. Análisis y Especificación de Requisitos RIA

A continuación se describe nuestra propuesta para el análisis de requisitos Web en RIAs. Nuestra propuesta se resume en la Figura 1, en donde se puede ver una serie de pasos para obtención de las posibles distribuciones de los requisitos entre el cliente y el servidor, los pasos se describen a continuación.

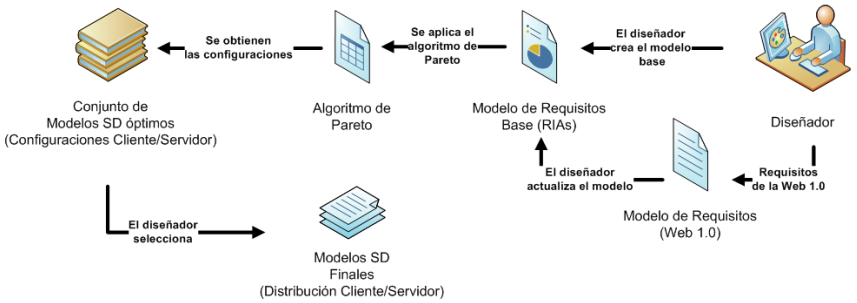


Figura 1: Visión general de la propuesta para el análisis de requisitos RIA.

Paso 1.- Crear el modelo de requisitos base para la RIA. En este paso, el diseñador creará un modelo de requisitos utilizando el *framework i**, con la diferencia de que serán incluidos requisitos específicos de las RIAs, tales como "Proporcionar interfaz gráfica" y las *softgoals*, como el caso de "Tiempo de respuesta" y "Ancho de banda limitado". Cabe destacar que, en este paso, aún no consideramos la distribución de la funcionalidad de los requisitos entre el cliente y el servidor, por consiguiente, el tipo de contribuciones que recibirá cada *softgoal* aún no puede ser determinado debido a que depende del lugar donde se implemente el requisito del que recibe la contribución. Por lo tanto, en el modelo de requisitos base, las etiquetas de las contribuciones de los requisitos funcionales hacia las *softgoals* se mantendrán en estado "*unknown*" por el momento, hasta que sea considerada la distribución de los requisitos funcionales entre el servidor y el cliente. En algunas ocasiones, la contribución de un requisito funcional a una *softgoal* puede ser expresada desde el modelo de requisitos base si resulta ser demasiado obvia, es decir, si ese requisito en específico se sabe que deberá

ser implementado en el cliente o en el servidor, así se sabrá de antemano el tipo de contribución que realizará a determinada *softgoal*.

En caso de existir un modelo de requisitos que no especifíque requisitos RIA (como el caso de un modelo de requisitos para la Web 1.0), el diseñador podrá adaptarlo añadiendo los requisitos junto con sus respectivas contribuciones hacia las *softgoals* (etiquetadas como "*unknown*"). Asimismo, si en el modelo ya existieran contribuciones de los requisitos hacia las *softgoals*, las etiquetas de las contribuciones serán removidas en caso de que se vea afectado el requisito que la origina por la distribución cliente/servidor.

Paso 2.- Obtener la distribución cliente/servidor óptima (Pareto) de los requisitos RIA. El objetivo es obtener un conjunto de posibles distribuciones (configuraciones) de los requisitos entre el cliente y el servidor. Es decir, para cada requisito, el diseñador necesita decidir en donde contribuirá mejor a la satisfacción de la *softgoal*, en el cliente o en el servidor. Esta no es una decisión fácil, por que cada requisito realiza contribuciones a diferentes *softgoals* de distinta forma dependiendo de donde sea implementado. Además, cuando se maximiza una *softgoal*, por lo regular se tiende a disminuir la contribución a otra, por lo que se puede decir que no existe una configuración cliente/servidor neutral y es necesario realizar una compensación entre las distintas *softgoals*. Debido a esto, en nuestro trabajo se desarrolló un algoritmo basado en la eficiencia de Pareto, el cual permite obtener un conjunto de configuraciones óptimas para la distribución de los requisitos entre el cliente y el servidor balanceando la maximización de las *softgoals* en conjunto y no así balanceando la maximización de una sola *softgoal*. La eficiencia de Pareto [31,7] es una temática que proviene del estudio de la economía y ha sido ampliamente aplicada en la Ingeniería de *Software*, en este caso en particular lo enunciamos como: *"Dado un conjunto de ubicaciones alternativas (cliente y servidor) y un conjunto de individuos (requisitos), la ubicación A es una optimización sobre la ubicación B solo si A puede al menos, optimizar a un individuo mejor que B (maximizar softgoals), sin deteriorar a otro individuo (afectar negativamente a las softgoals")*. Por lo tanto, llamamos configuración óptima de Pareto a aquella configuración que mejor satisfaga a una *softgoal* mientras satisface a las demás de igual forma. El conjunto de soluciones óptimas de Pareto puede ser utilizado por el diseñador para la toma de decisiones, por ejemplo, cuando necesite seleccionar la configuración que mejor balancee la compensación entre las *softgoals* y cuando necesite considerar la maximización sobre una sola *softgoal*.

Encontrar el conjunto de configuraciones óptimas de Pareto se define como: encontrar un vector de decisión integrado por variables de decisión X (distribución de los requisitos entre cliente y servidor) el cual maximice un vector formado por M funciones objetivo $f_i(X)$, por ejemplo, la satisfacción de la *softgoal* i en la distribución cliente/servidor X, donde $i = 1..M$ (sea M la cantidad de *softgoals*). Para lograrlo, el concepto de dominio entre vectores es definido como: un vector de decisión X domina a un vector de decisión Y ($X \succ Y$) si y solo si sus vectores objetivos de funciones objetivo $f_i(X)$ y $f_j(X)$ satisface: $\forall i \in \{1...M\} f_i(X) \geq f_i(Y)$ y $\exists i \in \{1...M\} f_i(X) > f_i(Y)$, se dice entonces que

todos los vectores de decisión que no son dominados por otros vectores de decisión forman el conjunto de configuraciones óptimas de Pareto, mientras que los vectores objetivo constituyen la frontera de Pareto.

La configuración óptima de Pareto permitirá obtener la distribución de los requisitos entre el cliente y el servidor, a la par, las *softgoals* serán maximizadas y balanceadas. Se ha definido el siguiente algoritmo para obtener la configuración óptima de Pareto:

2.1.- Cada posible distribución de N requisitos entre el cliente y el servidor es almacenada en un vector de decisión X_v: $\forall v0 \preceq v < 2^N$, $\forall i \in \{1...N\} X_{v_i} = T_i$, donde X_{v_i} es el $i-$ésimo elemento de X_v, $T_i = S$ si el requisito es ubicado en el servidor y $T_i = C$ si el requisito debe ser implementado en el cliente.

2.2.- Dependiendo de donde sea implementado el requisito (cliente o servidor), la contribución de cada requisito a cada *softgoal* debe de ser cuantificada. Para hacerlo es necesario crear una matriz utilizando los pesos que se muestran a continuación:

- w= 0 si el requisito no realiza ninguna contribución a las *softgoal*.
- w= +1 si existe una contribución del tipo *Help*.
- w= -1 si existe una contribución del tipo *Hurt*.
- w= +2 si existe una contribución del tipo *Some +*.
- w= -2 si existe una contribución del tipo *Some -*.
- w= +4 si existe una contribución del tipo *Make*.
- w= -4 si existe una contribución del tipo *Break*.

Cabe destacar que los pesos indicados son representados con valores del 0 al 4 para poder indicar la fuerza que ejerce la contribución, por ejemplo, una contribución del tipo *Make* recibe un peso de $w = +4$ para indicar que es la contribución positiva de mayor fuerza que se puede realizar por parte de un requisito a una o varias *softgoals*.

Ahora bien, cada entrada de la matriz $W_{i_j}^k$ corresponde a la contribución del i-ésimo requisito al elemento j en la distribución k (cliente o servidor): $\forall i \in \{1...N\}$, $\forall j \in \{1...M\}$, $\forall k \in \{C, S\}$ $W_{i_j}^k = w$, donde N es la cantidad de requisitos, M representa la cantidad de *softgoals* y w es definido como se describió previamente.

2.3.- Para cada *softgoal* j es calculada su correspondiente función objetivo F_j con respecto al vector de decisión X_v, por lo que es necesario sumar las contribuciones que todos los requisitos realizan a cada *softgoal* j considerando la distribución cliente/servidor definida en x_v: $\forall j \in \{1...M\}$, $\forall v0 \preceq v < 2^N$ $F_j(X_v) = \Sigma_{j=1}^{M} W_{i_j}^k$, donde N es la cantidad de requisitos y M de *softgoals*.

2.4.- Por último, se calcula la suma de todas las funciones objetivo con respecto al vector de decisión X_v para obtener la sumatoria general del vector de decisión X_v: $\forall j \in \{1...N\}$, $\forall v0 \preceq v < 2^N$ $\Sigma_{j=1}^{M} F_j(X_v)$, donde N es la cantidad de requisitos y M de *softgoals*.

Paso 3.- Seleccionar la distribución final cliente/servidor. Una vez calculado el conjunto de distribuciones óptimas de Pareto, le corresponde al diseñador seleccionar la distribución a implementar. Para poder hacerlo, el diseñador debe de considerar cuáles son las *softgoals* a maximizar (de acuerdo con el *stakeholder*) y seleccionar la configuración que las maximice sin que cause un impacto negativo en las otras *softgoals* (o que el impacto sea el menor posible). Finalmente, cabe destacar que nuestra aproximación ofrece una visión global al

diseñador sobre el impacto positivo o negativo que puede causar en las *softgoals* el implementar algún requisito en el servidor o en el cliente, por lo tanto, el diseñador dispondrá de varias configuraciones que le permitirán satisfacer, en lo posible, las necesidades del *stakeholder* considerando las *softgoals*.

5. Caso de Estudio

En esta sección se presenta un caso de estudio para demostrar la aplicabilidad de nuestra prupuesta. El caso de estudio es acerca de una compañía de bioinformática la cual tiene como objetivo ofrecer servicios en línea sobre el análisis del genóma humano. La aplicación Web permitirá al usuario subir la información sobre un gen, analizar la información y obtener reportes personalizados. El primer y único servicio disponible para el usuario será el reporte de enfermedades al que es vunerable, para lograrlo, se comparará la información proporcionada por el usuario con la base de datos de genes de la compañía. La compañía, desea ofrecer una Web basada en RIA con la finalidad de ofrecer una aplicación Web lo suficientemente atractiva e interactiva para los usuarios. Por tal motivo, el equipo de desarrollo de la compañía quiere estudiar y analizar los beneficios que se obtendrán de la distribución de los requisitos funcionales entre el servidor y el cliente así como el impacto que tendrán en las *softgoals*. Para efectos demostrativos, en el caso de estudio nos enfocaremos en los requisitos necesarios para el reporte de enfermedades.

De acorde con la propuesta presentada en la Sección 4, el primer paso consiste en la especificación del modelo de requisitos base para RIAs por parte del diseñador. En él, el diseñador deberá especificar los objetivos, los requisitos funcionales y las *softgoals* necesarias para modelar el reporte de enfermedades (Figura 2).

Como se explicó en el Paso 1 de la Sección 4, aún no sabemos cuales requisitos serán ubicados de lado del cliente o de lado del servidor, así que las contribuciones por parte de los requisitos a las *softgoals* son etiquetadas como "*unknown*". En el particular caso de la *softgoal* "Compatibilidad", las contribuciones recibidas por parte de los requisitos quedan establecidas en este paso debido a que no dependen del lugar donde se implementará el requisito (cliente o servidor). Las contribuciones originadas en el requisito "Proporcionar interfaz gráfica" también son etiquetadas debido a que el requisito solo puede ser implementado del lado del cliente.

El siguiente paso consiste en aplicar el algoritmo para obtener la eficiencia de Pareto. Primero es necesario identificar a los requisitos que realizan contribuciones a las *softgoals*, no se deberá considerar el requisito "Proporcionar interfaz gráfica" por que, como lo mencionamos antes, solo es posible implementarlo del lado del cliente. La Tabla 2 muestra los requisitos y *softgoals* a utilizar.

El siguiente paso consiste en calcular los vectores de decisión posibles, es decir todas las posibles configuraciones cliente/servidor, en donde C representa si el requisito se implementará en el Cliente y S si se implementará en el Servidor, como puede verse en la Tabla 3, de la columna 2 a la columna 5. Después

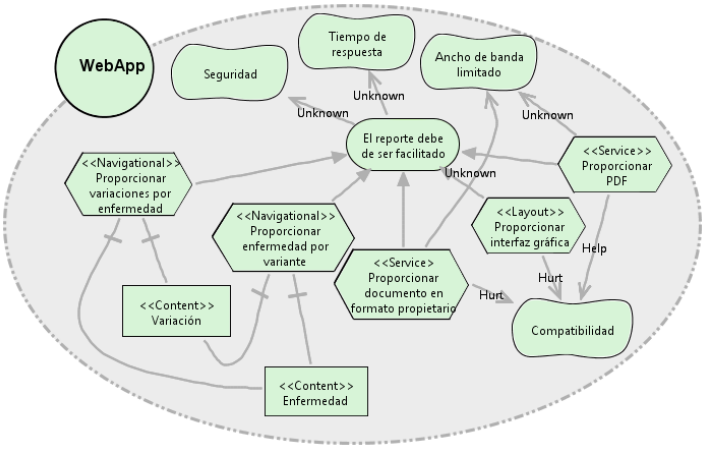


Figura 2: Modelo de requisitos RIA base.

Tabla 2: *Softgoals* y Requisitos detectados en el modelo de requisitos RIA base.

Softgoals	**Requisitos**
S1.- Seguridad	R1.- Proporcionar variaciones por enfermedad
S2.- Tiempo de respuesta	R2.- Proporcionar enfermedad por variante
S3.- Compatibilidad	R3.- Proporcionar PDF
S4.- Ancho de banda limitado	R4.- Proporcionar documento en formato propietario

se deberá calcular el resultado de las correspondientes funciones objetivo, para hacerlo, se han definido dos matrices (Matriz Cliente (1) y Matriz Servidor (2)) que representan la contribución de cada requisito a cada una de las *softgoals*, por ejemplo, la fila 3 (requisito "Proporcionar PDF"), columna 4 (*softgoal* "Tiempo de respuesta") muestra +1 en la Matriz Cliente, por lo que es una contribución del tipo "*Help*" si el requisito es implementado en el Cliente, en cambio, en la Matriz Servidor muestra -1, indicando una contribución del tipo "*Hurt*" si el requisito se implementa en el Servidor. El resultado de las funciones objetivo es mostrado de la columna 6 a la columna 9 de la Tabla 3. Por último, se indica si el vector de decisión esta en la frontera de Pareto (última columna).

$$M^k_{i_j} = \begin{pmatrix} -1 & +1 & 0 & 0 \\ -1 & +1 & 0 & 0 \\ -1 & +1 & +1 & -1 \\ -1 & +1 & -1 & -1 \end{pmatrix} \quad (1)$$

$$M^k_{i_j} = \begin{pmatrix} +1 & -1 & 0 & 0 \\ +1 & -1 & 0 & 0 \\ +1 & -1 & +1 & +1 \\ +1 & -1 & -1 & +1 \end{pmatrix} \quad (2)$$

En la Tabla 3, las filas de color gris son la frontera de Pareto. El diseñador puede seleccionar la frontera de Pareto que mejor satisfaga las necesidades del *stakeholder*, es decir, considerando las *softgoals* de mayor interés. Por mencionar

Tabla 3: La posible distribución de los requisitos entre el servidor y el cliente a través de la compensación de las *softgoals*.

Configuración	R1	R2	R3	R4	F(S1)	F(S2)	F(S3)	F(S4)	Σ	Frontera de Pareto
X1	C	C	C	C	-4	+4	0	+2	+2	Si
X2	C	C	C	S	-2	+2	0	0	0	No debido a X5
X3	C	C	S	C	-2	+2	0	0	0	No debido a X6
X4	C	C	S	S	0	0	0	-2	-2	No debido a X13
X5	C	S	C	C	-2	+2	0	+2	+2	Si
X6	C	S	C	S	0	0	0	0	0	Si
X7	C	S	S	C	0	0	0	0	0	Si (como X6)
X8	C	S	S	S	+2	-2	0	-2	-2	No debido a X14
X9	S	C	C	C	-2	+2	0	+2	+2	Si (como X5)
X10	S	C	C	S	0	0	0	0	0	Si (como X6)
X11	S	C	S	C	0	0	0	0	0	Si (como X6)
X12	S	C	S	S	+2	-2	0	-2	-2	No debido a X14
X13	S	S	C	C	0	0	0	+2	+2	Si
X14	S	S	C	S	+2	-2	0	0	0	Si
X15	S	S	S	C	+2	-2	0	0	0	Si (como X14)
X16	S	S	S	S	+4	-4	0	-2	-2	Si

un ejemplo, tenemos que la configuración X1 (Tabla 3) es la mejor opción si la *softgoal* "Tiempo de respuesta" es la prioridad, en la solución cada requisito será implementado de lado del Cliente y de acuerdo con el resultado positivo de la columna 10 (Σ) las *softgoals* restantes serán maximizadas o permanecerán igual, es decir, no serán afectadas negativamente. Por otro lado, la mejor opción respecto a la *softgoal* "Seguridad" es la configuración X16 (Figura 3), de acuerdo con Σ (-2), mejorar la seguridad afectará negativamente en algunas de las *softgoals*.

Por último, el resto de configuraciones de la frontera de Pareto (Tabla 3) son configuraciones intermedias que estarán disponibles para que el diseñador pueda seleccionarlas de acuerdo a la compensación de *softgoals* que necesite, por ejemplo, las configuraciones X6 y X14, ambas tienen $\Sigma = 0$, por lo tanto, todas las *softgoals* están balanceadas. En la configuración X14, los requisitos R1, R2 y R4 son ubicados en el servidor y R2 en el cliente, la solución quiza proporcione una buena compensación en caso de que la seguridad sea una prioridad por parte del *stakeholder*, pero no se mejorarán el resto de *softgoals*, incluso, la *softgoal* " Tiempo de respuesta" será afectada negativamente.

6. Conclusiones

En este artículo se ha presentado una extensión a nuestro método para el análisis de requisitos en Ingeniería Web. Con esta extensión se brinda soporte a la especificación y análisis de requisitos para RIAs. Debido a que nuestro método es orientado a objetivos, le permite al diseñador modelar y analizar los objetivos y expectativas del usuario así como evaluar y decidir sobre la distribución de los requisitos funcionales, es decir, si los requisitos serán implementados en el cliente o en el servidor. Para lograr esto, se ha adaptado un algoritmo basado en el óptimo de Pareto para compensar y maximizar las *softgoals*. Con el algoritmo el diseñador dispone de un conjunto de configuraciones de donde podrá eligir la que necesite de acuerdo a la prioritización de las *softgoals* por parte del *stakeholder*. Además, se ha ejemplificado nuestra propuesta utilizando un extracto de un caso de estudio real aplicado a una empresa de bioinformática. Nuestro trabajo

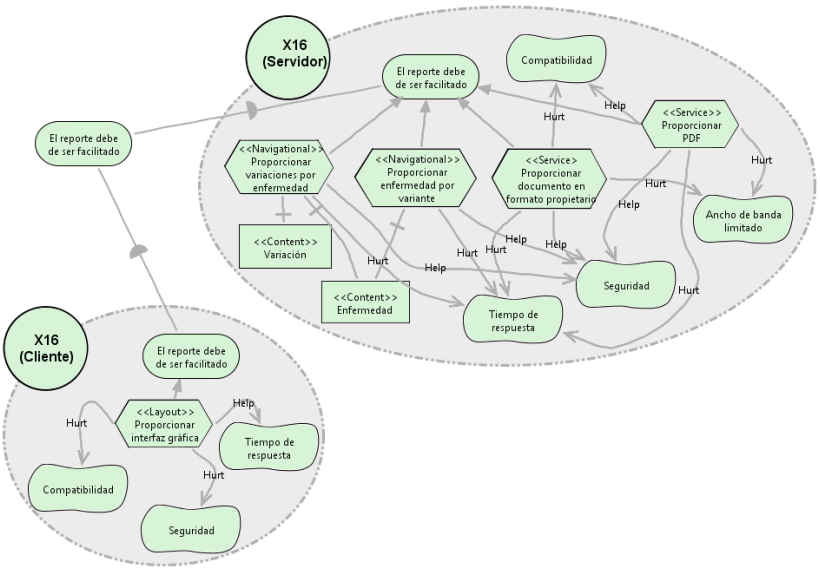


Figura 3: Configuración X16: distribución de los requisitos entre el cliente y el servidor.

futuro consistirá en la integración de la extensión en un proceso de Ingeniería Web Dirigida por Modelos (MDWE).

Finalmente, la propuesta se ha implementado utilizando el método Web A-OOH pero puede aplicarse a otros métodos de Ingeniería Web.

Agradecimientos. El trabajo presentado ha sido financiado por el proyecto MANTRA (GV/2011/035) del Ministerio Español de Eduación y Ciencia, MANTRA (GRE0917) de la Universidad de Alicante, SERENIDAD (PEII-11-0327-7035) de la Junta de Comunidades de Castilla La Mancha (España) y por el proyecto MESOLAP (TIN2010-14860) del Ministerio Español de Educación y Ciencia. José Alfonso Aguilar es subvencionado por el Consejo Nacional de Ciencia y Tecnología México (CONACYT) y por la Universidad Autónoma de Sinaloa, México. Sven Casteleyn está subvencionado por la Comisión Europea bajo el programa Marie Curie Intra-European Fellowship for Career Development (IEF), FP7-PEOPLE-2009-IEF, N° 254383 (SeMaRi).

Referencias

1. Aguilar, J.A., Garrigós, I., Mazón, J.N.: Modelos de weaving para trazabilidad de requisitos web en a-ooh. In: In DSDM: Actas del VII Taller sobre Desarrollo de Software Dirigido por Modelos, JISBD, Congreso Espanol de Informatica (CEDI). pp. 146–155. SISTEDES, Valencia, Espana (2010)

2. Aguilar, J.A., Garrigós, I., Mazón, J.N., Trujillo, J.: An mda approach for goal-oriented requirement analysis in web engineering. J. UCS 16(17), 2475–2494 (2010)
3. Aguilar, J.A., Garrigós, I., Mazón, J.N., Trujillo, J.: Web engineering approaches for requirement analysis - a systematic literature review. In: WEBIST (1). pp. 187–190 (2010)
4. Bozzon, A., Comai, S., Fraternali, P.: Current Research on the Design of Web 2.0 Applications Based on Model-Driven Approaches. ICWE 2008 Workshops pp. 25–31 (2008)
5. Bozzon, A., Comai, S., Fraternali, P., Carughi, G.T.: Conceptual modeling and code generation for rich internet applications. In: Proceedings of the 6th international conference on Web engineering - ICWE '06. p. 353. ACM Press, New York, New York, USA (2006)
6. Ceri, S., Fraternali, P., Bongio, A.: Web modeling language (webml): a modeling language for designing web sites. The International Journal of Computer and Telecommunications Networking 33(1-6), 137–157 (2000)
7. Collette, Y., Siarry, P.: Multiobjective optimization: principles and case studies. Springer Verlag (2003)
8. Cormode, G., Krishnamurthy, B.: Key differences between Web 1.0 and Web 2.0. First Monday 13(6), 2 (2008)
9. Dolog, P., Stage, J.: ADRIA: A Method for Abstract Design of Rich Internet Applications for the Web 2.0
10. Dolog, P., Stage, J.: Designing interaction spaces for rich internet applications with uml. Web Engineering pp. 358–363 (2007)
11. Escalona, M., Koch, N.: Requirements engineering for web applications-a comparative study. Journal of Web Engineering 2, 193–212 (2004)
12. Garrigós, I.: A-OOH: Extending Web Application Design with Dynamic Personalization. Ph.D. thesis, University of Alicante, Spain (2008)
13. Garrigós, I., Mazón, J.N., Trujillo, J.: A requirement analysis approach for using i* in web engineering. In: ICWE. pp. 151–165 (2009)
14. Gómez, J., Cachero, C., Pastor, O.: Extending a conceptual modelling approach to web application design. In: CAiSE '00: Proceedings of the 12th International Conference on Advanced Information Systems Engineering. pp. 79–93. Springer-Verlag, London, UK (2000)
15. Koch, N.: Transformation techniques in the model-driven development process of UWE. In: Workshop proceedings of the sixth international conference on Web engineering. p. 3. ACM (2006)
16. Koch, N., Kraus, A.: The expressive power of uml-based web engineering. In: IWWOST 02: Second Int. Worskhop on Web-oriented Software Technology (2002)
17. Koch, N., Pigerl, M., Zhang, G., Morozova, T.: Patterns for the model-based development of rias. In: Proceedings of the 9th International Conference on Web Engineering. pp. 283–291. ICWE '9, Springer-Verlag, Berlin, Heidelberg (2009)
18. Linaje, M., Preciado, J.C., Sanchez-Figueroa, F.: Engineering rich internet application user interfaces over legacy web models. IEEE Internet Computing 11, 53–59 (2007)
19. Lowe, D.: Web system requirements: an overview. Requirements Engineering 8, 102–113 (2003)
20. Luna, E.R., Garrigós, I., Mazón, J.N., Trujillo, J., Rossi, G.: An i*-based approach for modeling and testing web requirements. J. Web Eng. 9(4), 302–326 (2010)
21. Machado, L., Filho, O., Ribeiro, J.a.: Uwe-r: an extension to a web engineering methodology for rich internet applications. WSEAS Trans. Info. Sci. and App. 6, 601–610 (April 2009)

22. McDonald, A., Welland, R.: Web engineering in practice. In: Proceedings of the fourth WWW10 Workshop on Web Engineering. pp. 21–30 (2001)
23. Meliá, S., Gómez, J., Pérez, S., Díaz, O.: A model-driven development for GWT-based Rich Internet Applications with OOH4RIA. In: Web Engineering, 2008. ICWE'08. Eighth International Conference on. pp. 13–23. IEEE (2008)
24. Meliá, S., Martínez, J., Mira, S., Osuna, J., Gómez, J.: An Eclipse Plug-in for Model-Driven Development of Rich Internet Applications. Web Engineering pp. 514–517 (2010)
25. Meliá, S., Martínez, J.J., Pérez, Á., Gómez, J.: Ooh4ria tool: Una herramienta basada en el desarrollo dirigido por modelos para las rias. In: JISBD. pp. 219–222 (2009)
26. Pastor, O., Abrah ao, S.M., Fons, J.: An object-oriented approach to automate web applications development. In: EC-Web 2001: Proceedings of the Second International Conference on Electronic Commerce and Web Technologies. pp. 16–28. Springer-Verlag, London, UK (2001)
27. Preciado, J.C., Linaje, M., Sanchez, F., Comai, S.: Necessity of methodologies to model rich internet applications. In: Proceedings of the Seventh IEEE International Symposium on Web Site Evolution. pp. 7–13. IEEE Computer Society, Washington, DC, USA (2005)
28. Preciado, J., Linaje, M., Comai, S., Sanchez-Figueroa, F.: Designing rich internet applications with web engineering methodologies. In: Web Site Evolution, 2007. WSE 2007. 9th IEEE International Workshop on. pp. 23–30. IEEE (Oct 2007)
29. Preciado, J., Linaje, M., Sanchez-Figueroa, F.: Adapting Web 1.0 User Interfaces to Web 2.0 Multidevice User Interfaces using RUX-Method. Journal of Universal Computer Science 14(13), 2239–2254 (2008)
30. Sommerville, I.: Software Engineering. Addison-Wesley, 6th edn. (2001)
31. Szidarovszky, F., Gershon, M., Duckstein, L.: Techniques for multiobjective decision making in systems management. Elsevier (1986)
32. Urbieta, M., Rossi, G., Ginzburg, J., Schwabe, D.: Designing the interface of rich internet applications (2007)
33. Valverde, F.: OOWS 2.0: Un Método de Ingeniería Web Dirigido por Modelos para la Producción de Aplicaciones Web 2.0. Ph.D. thesis, Universidad Politecnica de Valencia (2010)
34. Valverde, F., Pastor, O.: Facing the Technological Challenges of Web 2.0: A RIA Model-Driven Engineering Approach. Web Information Systems Engineering-WISE 2009 pp. 131–144 (2009)
35. Wright, J.M., Dietrich, J.B.: Requirements for rich internet application design methodologies. In: Proceedings of the 9th international conference on Web Information Systems Engineering. pp. 106–119. WISE '08, Springer-Verlag, Berlin, Heidelberg (2008)
36. Wright, J.: A Modelling Language for Interactive Web Applications. In: Automated Software Engineering, 2009. ASE'09. 24th IEEE/ACM International Conference on. pp. 689–692. IEEE (2010)
37. Yu, E.: Modelling Strategic Relationships for Process Reenginering. Ph.D. thesis, University of Toronto, Canada (1995)