Universidad de Castilla-La Mancha

Escuela Superior de Ingeniería Informática

Departamento de Sistemas Informáticos
Programa Oficial de Postgrado en Tecnologías Informáticas Avanzadas

Trabajo Fin de Máster

CSRML: un Lenguaje para la Especificación de Requisitos en Sistemas Colaborativos

Julio de 2011

Alumno: Miguel Ángel Teruel Martínez

Tutores: Dr. D. Francisco Montero Simarro
Dr. D. Pascual González López

Índice de Contenidos

Índice de Contenidos I
Índice de Figuras V
Índice de Tablas VII

Parte I. Currículum Vitae

1. Curriculum Vitae 3
   1.1. Titulación académica 3
   1.2. Otras titulaciones académicas 3
   1.3. Participación en Proyectos de Investigación 3
   1.4. Becas disfrutadas 3
   1.5. Participación en Seminarios, Congresos y en Eventos de Difusión Científica 4
   1.6. Otros méritos 5
      1.6.1. Experiencia laboral 5
      1.6.2. Cursos y Formación Complementaria 6

Parte II. Resumen de las asignaturas de Máster realizadas

2. Resumen de las asignaturas de máster realizadas 9
   2.1. Generación de Documentos Científicos en Informática 9
   2.2. Nuevos Paradigmas en HCI 10
   2.3. Sistemas Inteligentes aplicados a Internet 12
   2.4. Calidad de Interfaces de Usuario: Desarrollo Avanzado 13
   2.5. Tecnología Software Orientada a Objetos 14
   2.6. Computación en Clusters 15

Parte III. Estado del arte, Trabajo fin de Máster y Líneas futuras de investigación

3. Estado del Arte de los conocimientos sobre el tema de investigación 19
   3.1. Requisitos, Sistemas Colaborativos y *Awareness* 19
   3.2. Técnicas de Ingeniería de Requisitos para Sistemas Colaborativos 20
      3.2.1. Casos de Uso 20
      3.2.2. *Viewpoints* 22
      3.2.3. *Goal-Oriented* 23
         NFR framework 24

Metodología KAOS 26
*i** framework 27
4. Trabajo Fin de Máster – CSRML: un Lenguaje para la Especificación de Requisitos en Sistemas Colaborativos 31
4.1. Análisis Empírico de Técnicas de Ingeniería de Requisitos para Sistemas Colaborativos 31
4.1.1. Caso de estudio 31
Cursores remotos 32
Lista de participantes y chat 32
Historial de revisiones 33
4.1.2. Evaluación empírica 33
Modelando el caso de estudio 33
Goal-Oriented 34
Casos de Uso 35
Viewpoints 35
Evaluando las técnicas de Ingeniería de Requisitos 36
4.2. Comparativa de Enfoques *Goal-Oriented* para el Modelado de Requisitos para Sistemas Colaborativos 40
4.2.1. Evaluación empírica 40
Modelando el caso de estudio 40
NFR framework 40
Metodología KAOS 41
Evaluando los enfoques Goal-Oriented 41
4.3. Especificando Requisitos de Sistemas Colaborativos con CSRML 45
4.3.1. CSRML: un lenguaje de modelado de requisitos para sistemas colaborativos 45
4.3.2. Caso de estudio: Sistema de gestión de congresos con revisiones colaborativas 47
4.3.3. Modelando el sistema con CSRML 48
4.4. Validando la comprensibilidad de CSRML mediante un experimento controlado 51
4.4.1. Diseño Experimental 51
4.4.2. Condiciendo el Experimento 53
4.4.3. Análisis de los resultados 53
4.4.4. Riesgos sobre la Validez del Experimento 56
Validez de la Conclusión 56
Validez Interna 56

Validez de Construcción 56
Validez Externa 56
4.5. Conclusiones 56
5. Líneas futuras de investigación 59
5.1. Introducción 59
5.2. Dominio del problema 59
5.3. Propuesta de tesis 59
5.3.1. Calidad de proceso 61
5.3.2. Calidad del producto 62
5.3.3. Calidad en uso 63
5.4. Planificación 64

Bibliografía 67

Índice de Figuras

Figura 1: Arquitectura del sistema Kinect Awareness Sensor 10
Figura 2: Aplicación detectora de usuarios para Kinect 11
Figura 3: Componentes de un diagrama de casos de uso 21
Figura 4: Ejemplo de *viewpoint* 22
Figura 5: Ejemplo de Modelo de Objetivos 23
Figura 6: Modelo de un sistema con *NFR framework* 25
Figura 7: Elementos del *NFR framework* 26
Figura 8: Modelo KAOS 26
Figura 9: Construcciones básicas del *framework* KAOS 27
Figura 10: Modelo *i** de lógica estratégica 28
Figura 11: Objetos en *i** 29
Figura 12: Relaciones en *i** 29
Figura 13: Interfaz de Google Docs 32
Figura 14: Cursor remoto y texto seleccionado remotamente 32
Figura 15: Dos usuarios chateando a través de la lista de participantes 32
Figura 16: Historial de revisiones mostrando una eliminación de texto 33
Figura 17: Modelo *Goal-Oriented* 34
Figura 18: Diagrama de casos de uso con extensión para NFRs 35
Figura 19: *Viewpoint* para los requisitos de *awareness* y los factores de calidad (adaptado de [14]) 36
Figura 20: Resultados del análisis empírico 39
Figura 21: Resultados relativos a las distintas características 40
Figura 22: Modelo *Goal-Oriented* NFR 41
Figura 23: Modelos KAOS de Objetivos y Responsabilidades 42
Figura 24: Resultado del análisis para los enfoques GO 44
Figura 25: Resultados relativos a las distintas características para los enfoques GO 45
Figura 26: Elementos de CSRML 47
Figura 27: Diagrama de objetivos del sistema 48
Figura 28: Diagrama de responsabilidades 48
Figura 29: Diagrama de refinamiento de la tarea *Crear congreso* 49
Figura 30: Diagrama de refinamiento de la tarea *Envío de artículos* 49
Figura 31: Diagrama de refinamiento de la tarea *Asignación de artículos* 50
Figura 32: Diagrama de refinamiento de la tarea *Revisión de artículos* 50
Figura 33: Diagrama de factores de calidad 50
Figura 34: Gráfico de cajas de los resultados obtenidos 54
Figura 35: Resultados de las preguntas del Puzzle 55
Figura 36: Resultados de las preguntas del Congreso 55
Figura 37: Sistemas Colaborativos desde una perspectiva de calidad 60
Figura 38: Modelos en MDA 62
Figura 39: Actores en el proyecto de tesis 64
Figura 40: Distribución temporal de las actividades 66

Índice de Tablas

Tabla 1: Elementos del *Workspace Awareness* relacionados con el presente..... 20
Tabla 2: Elementos del *Workspace Awareness* relacionados con el pasado..... 20
Tabla 3: Relación entre elementos de *awareness* y requisitos funcionales 34
Tabla 4: Relación entre factores de calidad y funcionalidades de *awareness*..... 34
Tabla 5: Lista de características para la evaluación de las técnicas 37
Tabla 6: Importancia de las características 37
Tabla 7: Escala para evaluar el soporte de una característica (de [31])..... 37
Tabla 8: Evaluación para *Goal-Oriented*..... 38
Tabla 9: Evaluación para Casos de Uso..... 38
Tabla 10: Evaluación para Casos de Uso con extensión NFR..... 38
Tabla 11: Evaluación para *Viewpoints*..... 39
Tabla 12: Lista de características para la evaluación de los enfoques GO 42
Tabla 13: Importancia de las características de los enfoques GO 43
Tabla 14: Evaluación para *NFR framework* 43
Tabla 15: Evaluación para *i* framework* 43
Tabla 16: Evaluación para la metodología KAOS 43
Tabla 17: Características principales del experimento..... 52
Tabla 18: Diseño del experimento 53
Tabla 19: Media y desviación estándar (entre paréntesis) para la comprensibilidad en el experimento..... 54
Tabla 20: Resultados del test ANOVA del experimento..... 54

Parte I. Currículum Vitae

1. Currículum Vitae

1.1. Titulación académica

INGENIERÍA INFORMÁTICA
Universidad de Castilla – La Mancha
Julio 2010

1.2. Otras titulaciones académicas

INGENIERÍA TÉCNICA EN INFORMÁTICA DE SISTEMAS
Universidad de Castilla – La Mancha
Junio 2005

1.3. Participación en Proyectos de Investigación

NUEVOS ENTORNOS Y PARADIGMAS DE INTERACCION – Contratado a cargo al proyecto
Universidad de Castilla – La Mancha
01/04/2009 - 31/03/2012
Investigador: González López, Pascual J.

WIDGET CATALOG DEVELOPMENT (OBSV) – Becario de investigación
Universidad de Castilla – La Mancha
22/07/2009 - 31/12/2010
Investigador: Molina Masso, José Pascual

1.4. Becas disfrutadas[1]

Universidad de Castilla – La Mancha
Beca de Colaboración - Desarrollo de aplicaciones Web con PHP y MySQL. Diseño gráfico
De Marzo de 2010 a Noviembre de 2010
Escuela Superior de Ingeniería Informática de Albacete

Universidad de Castilla – La Mancha
Beca de Colaboración - Administración y mantenimiento de equipos informáticos
De Enero de 2010 a Marzo de 2010
Facultad de Medicina de Albacete

Universidad de Castilla – La Mancha
Beca de Colaboración - Apoyo a los sistemas informáticos del Servicio de Ayuda al Estudiante Discapacitado
De Enero de 2006 a Octubre de 2006
Servicio de Ayuda al Estudiante Discapacitado

[1] Actualmente está solicitada una beca FPU al Ministerio de Educación / Junta de Comunidades de Castilla – La Mancha

Universidad de Castilla – La Mancha
Beca de Colaboración - Encargado de laboratorio multimedia. Administrador de servidor Windows 2000 Advanced Server
De enero de 2005 a Diciembre de 2005
Facultad de Humanidades de Albacete

1.5. Participación en Seminarios, Congresos, Cursos y en Eventos de Difusión Científica

An Empirical Evaluation of Requirement Engineering Techniques for Collaborative Systems
Miguel A. Teruel, Elena Navarro, Víctor López-Jaquero, Francisco Montero and Pascual González
15th International Conference on Evaluation and Assessment in Software Engineering (EASE 2011) – April 2011 – Durham (United Kingdom) – Core A (40% acceptance rate)

A Comparative of Goal-Oriented Approaches to Modelling Requirements for Collaborative Systems
Miguel A. Teruel, Elena Navarro, Víctor López-Jaquero, Francisco Montero and Pascual González
6th International Conference on Evaluation of Novel Software Approaches to Software Engineering (ENASE 2011) – June 2011 – Beijing (China) – Core B (30% acceptance rate)

Modelado de Requisitos de Sistemas Colaborativos con CSRML
Miguel A. Teruel, Elena Navarro, Víctor López-Jaquero, Francisco Montero and Pascual González
XVI Jornadas de Ingeniería del Software y Bases de Datos (JISBD 2011) – September 2011 – A Coruña (Spain)

CSRML: A Goal-Oriented Approach to Model Requirements for Collaborative Systems
Miguel A. Teruel, Elena Navarro, Víctor López-Jaquero, Francisco Montero and Pascual González
30th International Conference on Conceptual Modeling (ER 2011) – October 2011 – Brussels (Belgium) – Core A (20% acceptance rate)

An extension of i* to Model Requirements for CSCW Systems Applied to Conference Preparation System with Collaborative Reviews (Enviado)
Miguel A. Teruel, Elena Navarro, Víctor López-Jaquero, Francisco Montero and Pascual González
5th International i Workshop (iStar 2011) – August 2011 – Trento (Italy)*

Assesing the Understandability of Collaborative Systems Requirements Notations: an Empirical Study (Enviado)
Miguel A. Teruel, Elena Navarro, Víctor López-Jaquero, Francisco Montero and Pascual González
1st International Workshop on Empirical Requirements Engineering (EmpiRE 2011) – August 2011 – Trento (Italy)

A Family of Experiments for the Evaluation of Understandability of CRSML (En preparación)
Miguel A. Teruel, Elena Navarro, Víctor López-Jaquero, Francisco Montero and Pascual González
Information and Software Technology – Elsevier - ISSN 0950-5849

1.6. Otros méritos

1.6.1. Experiencia laboral

Formación en informática a nivel usuario (Windows, Internet, Office)
Ayuntamiento de Bonete
De Noviembre de 2008 a Marzo de 2009

Formación en informática a nivel usuario (Windows, Internet, Office)
Ayuntamiento de Alpera
De Noviembre de 2007 a Mayo de 2008

Formación a usuarios de la tercera edad en nuevas tecnologías (Informática básica, Windows, Word, Internet, Fotografía Digital)
Fundación Ínsula Barataria
Julio de 2007

Formación en tecnologías .net, C# y SQL Server 2005 a programadores de sistemas AS/400 de la empresa
Procesos de Gestión S.L.
De Julio de 2007 a Diciembre de 2008

Encargado del Centro de Internet de Alpera. Administración de equipos, formación a usuarios
Ayuntamiento de Alpera
De Mayo de 2007 a Junio de 2008

Formación en informática a nivel usuario (Windows, Internet, Office) en el Centro de Internet de Higueruela
Ayuntamiento de Higueruela
De Noviembre de 2006 a Junio de 2008

Creación y administración de la empresa. Venta y reparación de equipos informáticos. Montaje y mantenimiento de redes. Diseño Web. Desarrollos a medida. Outsourcing. Formación
MTMinfo
De Julio de 2006 a Marzo de 2009

Diseño, implementación y mantenimiento de aplicaciones distribuidas y multiusuario en plataformas .Net con bases de datos DB2 sobre AS/400. Comercial para la provincia de Albacete. Servicio técnico. Montaje de redes. Diseño Web. Aplicaciones para PDA.
Procesos de Gestión S.L.
De Noviembre de 2005 a Junio de 2006

Diseñador Gráfico. Diseño y maquetación de diversas publicaciones
Imagrafic Artes Gráficas
De Junio de 2005 a Agosto de 2005

Montaje y mantenimiento de sala de libre acceso a Internet mediante un sistema distribuido Linux basado en Linux Terminal Server Project
Ayuntamiento de Alpera
Junio de 2004 a Septiembre de 2004

1.6.2. Cursos y Formación Complementaria

Soluciones de Movilidad y Seguridad Informática en las Pymes (60 horas)
FEDETICAM – Federación de Empresas de Tecnologías de la Información de Castilla – La Mancha
Desde Diciembre de 2007 a Febrero de 2008

Aplicación de las Nuevas Tecnologías En la gestión y dirección de empresas (140 horas)
FEDETICAM – Federación de Empresas de Tecnologías de la Información de Castilla – La Mancha
Desde Noviembre de 2007 a Febrero de 2008

Curso de Firma y Facturación Electrónica (30 horas)
FEDETICAM – Federación de Empresas de Tecnologías de la Información de Castilla – La Mancha
Desde Noviembre de 2007 a Diciembre de 2007

Parte II. Resumen de las asignaturas de Máster realizadas

2. Resumen de las asignaturas de máster realizadas

De las asignaturas ofertadas por este máster, he seleccionado aquellas que, o bien encuadraban mejor con mi línea de investigación o, en su defecto, me resultaban de interés personal. Las asignaturas son las siguientes:

2.1. Generación de Documentos Científicos en Informática

Esta asignatura tiene como finalidad formar al alumno en la redacción y presentación de documentos de índole científica con el fin de publicar el fruto de sus investigaciones.

La asignatura se divide en tres bloques. El primero de ellos, impartido por el Dr. D. José A. Gámez nos muestra en líneas generales el mundo de la investigación. En él se nos muestra la metodología asociada al campo de la investigación y se nos muestra como presentar públicamente nuestros resultados. Además se nos comenta temas relacionados con la evaluación de la investigación, cienciometría, índices de impacto, etc.

El segundo de los bloques, impartido por el Dr. D. Francisco Parreño está relacionado con los contrastes estadísticos. Aquí se nos enseña a contrastar nuestros métodos e hipótesis con el objetivo de realizar publicaciones asociadas a los mismos. En este bloque se nos presentan diferentes herramientas estadística y, en especial, el lenguaje y entorno estadístico R.

En el último bloque, el Dr. D. Luis de la Ossa nos adiestra en el uso del lenguaje LaTeX, ampliamente usado en la generación de documentos científicos de todo tipo. Además, se presenta el entorno Beamer para la realización de presentaciones.

Para la evaluación de esta asignatura se realizarán dos trabajos. El primero, de carácter opcional, consistirá en la realización de un estudio sobre un grupo de investigación con el objetivo de obtener su índice h. En mi caso, lo realicé sobre el grupo LoUISE, al cual pertenezco.

En segundo lugar, se nos pedía la confección y presentación de un documento científico en LaTeX en el cual se realizara un contraste estadístico a partir de una serie de datos obtenidos en un experimento previamente realizado. Dado que carecía de datos que tuviesen relación con mi actual línea de investigación, me decanté por la utilización de los resultados obtenidos en mi proyecto final de carrera [1].

2.2. Nuevos Paradigmas en HCI

Esta asignatura impartida por los doctores D. Antonio Fernández Caballero, Dña. Mª Teresa López Bonal y D. José Pascual Molina Massó nos muestra aspectos relacionados con las últimas tendencias en interacción persona-ordenador.

En esta asignatura vemos desde los fundamentos, aspectos de diseño y programación de las interfaces de usuario 3D hasta aspectos de visión artificial, pasando por la detección de movimiento y el reconocimiento de patrones en escenas tridimensionales. En el marco de la asignatura, también se nos dio una conferencia sobre agentes software en HCI a cargo del Dr. D. Francisco J. Garijo.

Para esta asignatura, intentando ligar los contenidos presentados y mi línea de investigación, elaboré un trabajo denominado "Videojuegos y HCI". En este trabajo se estudió la evolución de los dispositivos más avanzados (para la época en que se crearon) de HCI aplicados a los videojuegos. Tras realizar un recorrido histórico desde 1980 viendo estos dispositivos, realicé un estudio sobre el último y, a día de hoy, más popular de ellos: Microsoft Kinect, el cual se aplicará a un área muy dispar de la que es el ocio digital: los sistemas colaborativos.

En este caso de estudio se implementó un elemento de awareness para saber que usuarios hay conectados en el sistema, correspondiente a la técnica *Participant List* de la tesis doctoral de Carl Gutwin [2]

Para ello se creó una arquitectura distribuida con los siguientes elementos (Figura 1):

- Kinect: Un equipo con un Kinect conectado que detectará los usuarios que están trabajando en él.
- Servidor Web: Servidor en el que residirá la arquitectura de servicios web necesaria para la comunicación.
- Cliente: Usuario que percibirá la presencia remota de otros usuarios.

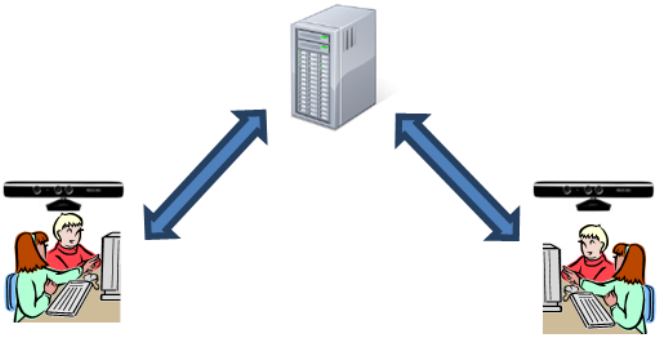

Figura 1: Arquitectura del sistema Kinect Awareness Sensor

Para que Kinect, un dispositivo creado para Xbox 360 funcione con un PC se usó un driver "no oficial" implementado sobre la plataforma OpenNI y para facilitar la implementación se utilizó el Middleware NITE de PrimeSense, el cual aporta una serie de librerías predefinidas de captura de movimiento.

Usando OpenNI, OpenGL, Glut y NITE, y partiendo de ejemplos desarrollados por la comunidad de programadores surgida a raíz del lanzamiento del driver no-oficial

de Kinect se creó la aplicación que se ejecutarán en los equipos que tengan instalado Kinect. Esta aplicación detectará a los usuarios que estén delante de la cámara infrarroja y lo comunicará vía servicios web.

En la Figura 2 podemos ver la aplicación ejecutándose. En ella se aprecia la detección de dos usuarios (azul y verde) y su distinción del fondo (escala de grises).

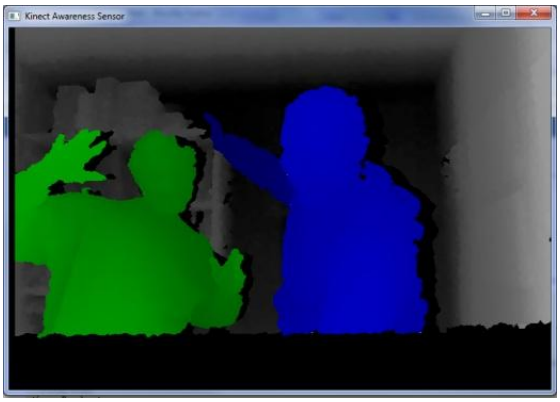

Figura 2: Aplicación detectora de usuarios para Kinect

Por otro lado, se crearon los servicios web que gestionarán la comunicación entre los distintos puestos de trabajo. Estos servicios web se implementarán usando .Net Framework 3.5 y serán ejecutados en un servidor web IIS.

Por último se desarrolló una aplicación de prueba que hace uso de los servicios web para comprobar que usuarios hay conectados al sistema. Esta aplicación puede ser ejecutada en uno o varios equipos, tanto locales a los usuarios de Kinect como remotos.

2.3. Sistemas Inteligentes aplicados a Internet

La temática de esta asignatura está relacionada con los campos de la inteligencia artificial y la minería de datos. Impartida por los doctores D. Ismael García Varea, D. José Miguel Puerta Callejón y Dña. M. Julia Flores Gallego, esta asignatura está dividida en tres bloques temáticos.

En el primero de ellos se nos presentan las redes bayesianas, su funcionamiento y sus consideraciones de diseño para en el segundo de ellos tratar del aprendizaje automático de las mismas. También en estos bloques se nos presenta la herramienta Elvira.

Ya en el tercero de los bloques se tratan las técnicas Metaheurísticas y la minería de datos. En esta parte se estudian entre otras propuestas los algoritmos genéticos y los algoritmos de estimación de distribuciones. En este bloque se realizan demostraciones prácticas con la herramienta Weka.

Para la superación de la asignaturas se realizó un estudio sobre aplicaciones de redes bayesianas a un caso real, así como una serie de ejercicios tanto de redes bayesianas como sobre minería de datos.

Es importante resaltar, ligando los contenidos de esta asignatura con mi línea de investigación, que en ciertos estudios de ingeniería del software se han aplicado redes bayesianas. Por ejemplo, en [3] se usan redes bayesianas para predecir defectos en el software, o en [4], para predecir la mantenibilidad de una aplicación.

2.4. Calidad de Interfaces de Usuario: Desarrollo Avanzado

Esta asignatura versa sobre el desarrollo de interfaces de usuario desde un punto de vista de la calidad, centrado en el usuario y guiado por procesos ingenieriles guiados por modelos.

Esta asignatura está dividida en tres bloques. En el primero de ellos, el Dr. D. Pascual González López nos presenta aspectos de calidad dentro de las interfaces de usuario, haciendo hincapié en la componente humana de la interacción, los modelos de usuario y el concepto de *awareness* y sus implicaciones a la hora de desarrollar interfaces de usuario.

El segundo bloque, impartido por el Dr. D. Víctor López Jaquero trata del Diseño de Interfaces de usuario basado en modelos y adaptación. En este bloque se estudian aspectos relacionados con MDA, modelos en MB-UID, el lenguaje de descripción de IU UsiXML, transformaciones de modelos en MB-UID y adaptación de interfaces de usuario.

Para concluir, el bloque impartido por el Dr. D. Francisco Montero Simarro trata sobre la calidad y desarrollo centrado en el usuario y sobre la integración de experiencia y evaluación de las interfaces.

Para superar la asignatura se presenta una comparativa de distintas técnicas de especificación de requisitos para sistemas colaborativos en la que se primará la representación del awareness y la calidad (véanse apartados 3 y 4).

2.5. Tecnología Software Orientada a Objetos

La asignatura Tecnología Software Orientada a Objetos, la cual está dividua en tres módulos, es impartida por los doctores Dña. Elena Navarro Martínez, Dña. María Dolores Lozano Pérez y D. Víctor Ruiz Penichet.

En el primero de los tres módulos, impartido por la Dra. Dña. Elena Navarro Martínez, se define formalmente el concepto de modelo (en particular desde la perspectiva orientada a objetos y de la arquitectura sotware), soportarlo sobre diversos formalismos que permitan su compilación. También se estudia su aplicación a diferentes campos (tecnologías): MDA, MOF, compilación de modelos, interoperabilidad semántica, plataformas de gestión de modelos, persistencia, evolución del SW, migración automática, metaprogramación.

El segundo módulo, el cual trata sobre Tendencias Actuales en el Desarrollo de Interfaces de Usuario es impartido por la Dra. María Dolores Lozano Pérez. En él, tras una introducción al desarrollo de interfaces de usuario se nos presentan los *Model-Based User Interface Development Environments* (MB-UIDEs) así como un entorno metodológico basado en modelos para el desarrollo de GUIs. Además, se realiza un ejercicio práctico sobre el desarrollo de un prototipo de interfaz basado en modelos. Finalmente, el módulo concluye con lala aplicación de MDA al desarrollo de interfaces para entornos ubícuos sensibles al contexto.

En el último de los módulos, el Dr. Víctor Ruiz Penichet nos presenta TOUCHE, una modelo de proceso para el desarrollo de interfaces en sistemas colaborativos, centrado en los usuarios y dirigido por tareas.

La asignatura concluye con una charla sobre experimentación en Ingeniería del Software y revisiones sistemáticas, impartida por la Dra. Dña. Marcela Genero Bocco, Profesora Titular de Universidad en el Departamento de Tecnologías y Sistemas de Información de la Universidad de Castilla-La Mancha, en Ciudad Real.

Como trabajo para la superación de la asignatura se presentará CSRML, un lenguaje para la especificación de requisitos de sistemas colaborativos (véase apartado 4.3).

2.6. Computación en Clusters

Esta asignatura es impartida por los doctores D. Francisco José Quiles Flor, D. José Luis Sánchez García y D. Francisco José Alfaro Cortés.

Un cluster es una tipo de arquitectura paralela distribuida que consiste de un conjunto de computadores independientes interconectados operando de forma conjunta como un único recurso computacional.

En esta asignatura se estudian y analizan los diferentes aspectos de un cluster de computadores, describiéndose las tendencias en cuanto a los nuevos sistemas de interconexión y E/S como por ejemplo Infiniband, PCI-X, RapidIO, etc. y las posibilidades y problemas a resolver en cuanto a la configuración de plataformas de tipo cluster, ilustrando ejemplos de aplicaciones que, por sus requisitos permiten aprovechar las prestaciones que proporcionan estas arquitecturas.

Así mismo, se analizan los entornos software necesarios para que una arquitectura de este tipo de la sensación al usuario de estar trabajando con un único recurso computacional. Adicionalmente se estudiarán los principales cluster existentes en el mercado.

Esta asignatura queda complementada por dos charlas. La primera de ellas, impartida por el Dr. D. Pedro Javier García García trata del control de congestión en redes de interconexión. En la segunda, D. Sergi Girona nos habla sobre el BSC (*Barcelona Supercomputing Center*).

Para la superación de ésta asignatura se procedió a paralelizar el código originalmente secuencial de mi Proyecto Final de Carrera [1], consiguiendo un *SpeedUp* (aceleración) de 6,64, sobre el código original, siendo el *SpeedUp* ideal de 8.

Parte III. Estado del arte, Trabajo fin de Máster y Líneas futuras de investigación

3. Estado del Arte de los conocimientos sobre el tema de investigación

Un sistema colaborativo es un software distribuido que permite a varios usuarios trabajar juntos y llevar a cavo tareas de colaboración, comunicación y coordinación. Para realizar estas tareas, los usuarios deben ser conscientes de las acciones de los otros usuarios, usualmente mediante un conjunto de técnicas de *awareness*. Cuando definimos un sistema colaborativo, las técnicas de *awareness* pueden ser consideradas requisitos no funcionales ligados a algunos factores de calidad, como la usabilidad. Así, pueden encontrarse serios defectos durante la especificación de estos sistemas si se usan las técnicas de Ingeniería de Requisitos actualmente disponibles debido a sus limitaciones expresivas cuando se trata con requisitos no funcionales.

3.1. Requisitos, Sistemas Colaborativos y *Awareness*

Al final de los 80, el Trabajo Cooperativo Asistido por Computador (*Computer Supported Cooperative Work*, CSCW) emergió como una nueva área de investigación centrada en el estudio del comportamiento humano en el contexto del trabajo y también, en el diseño de herramientas (*groupware*) que soportaran grupos de trabajo [6]. El problema de este tipo de sistemas es mantener el llamado *workspace awareness.*

Podemos definir el concepto de *Workspace Awareness* (WA) como el conocimiento instantáneo de la interacción de otra persona dentro de un espacio de trabajo compartido. WA incluye conocimiento acerca de *donde* están trabajando los demás usuarios, *que* están haciendo *ahora*, y *que* va a hacer *después* [7].

En un espacio de trabajo cara a cara, el *awareness* sobre alguien es relativamente fácil de mantener y los mecanismos de colaboración son naturales, espontáneos y no forzados. Desafortunadamente, el WA es mucho más difícil de mantener en espacios de trabajo *groupware* que en los entornos cara a cara y no es una tarea trivial determinar quien está presente en el espacio de trabajo, donde está trabajando ni que está haciendo.

Gutwin [7] presenta un marco conceptual para establecer qué tipo de información constituye el WA. Los elementos básicos son el conjunto de preguntas “quien, que, donde, cuando y como”. Así, cuando trabajamos con otras personas en un espacio físico compartido, sabemos con quien estamos trabajando, lo que hacen los demás y cuando y como ocurren determinados eventos.

Las Tablas 1 y 2 muestran esos elementos y enumera las preguntas que cada elemento puede responder. La Tabla 1 contiene aquellos elementos relacionados con el presente mientras la Tabla 2 contiene los relacionados con el pasado. Todos los elementos son cosas de sentido común relacionados con las interacciones entre una persona y su entorno.

Tabla 1: Elementos del *Workspace Awareness* relacionados con el presente

Categoría	Elemento	Preguntas específicas
Quién	Presencia	¿Hay alguien en el espacio de trabajo?
	Identidad	¿Quién está participando?
	Autoría	¿Quién está haciendo eso?
Qué	Acción	¿Qué están haciendo?
	Intención	¿Con que objetivo realiza esa acción?
	Artefacto	¿Con que objeto están trabando?
Dónde	Localización	¿Dónde están trabajando?
	Mirada	¿Dónde están mirando?
	Vista	¿Qué pueden ver?
	Alcance	¿Qué pueden alcanzar?

Tabla 2: Elementos del *Workspace Awareness* relacionados con el pasado

Categoría	Elemento	Preguntas específicas
Cómo	Histórico de acciones	¿Cómo ocurrió eso?
	Histórico de artefactos	¿Cómo llegó el artefacto a ese estado?
Cuándo	Histórico de eventos	¿Cuándo ocurrió ese evento?
Quién	Histórico de presencia	¿Quién estuvo aquí y cuándo?
Dónde	Histórico de localización	¿Dónde ha estado esa persona?
Qué	Histórico de acción	¿Qué ha estado haciendo esa persona?

Los requisitos de *awareness* pueden considerarse requisitos no funcionales (*Non-Functional Requirements*, NFR) o requisitos extrafuncionales (*Extra-Functional Requirements*, EFR) ya que suelen ser requisitos de calidad (por ejemplo, la fiabilidad o la usabilidad) [8]. Así, la especificación de este tipo de requisitos no es una cuestión trivial debido al alto número y a la diversidad de requisitos con los que están relacionados y por su alto impacto en términos de la arquitectura final de sistema. Por lo tanto, la selección adecuada de una técnica de especificación de requisitos se convierte en una decisión difícil e importante. Desafortunadamente, hasta donde nosotros sabemos, no hay trabajos destinados a determinar que enfoque de Ingeniería de Requisitos (*Requirements Engineering*, RE) se adapta mejor a la especificación de los sistemas colaborativos.

En [5], estudiamos la aplicabilidad de tres técnicas de RE (Casos de Uso [9], *Viewpoints* [10] y *Goal-Oriented* [11]) para la especificación de sistemas colaborativos, poniendo especial atención en los requisitos de *awareness*. Para llevar a cabo ese estudio, se especificaron algunos requisitos de awareness de un sistema real (Google Docs [12]). Una vez que el sistema es modelado, se realizó un análisis empírico para comparar esas tres técnicas.

3.2. Técnicas de Ingeniería de Requisitos para Sistemas Colaborativos

Las tres técnicas que se analizaron para realizar nuestros estudios fueron las siguientes:

3.2.1. Casos de Uso

Casos de Uso (*Use Cases*, UC) es quizá una de las aproximaciones más populares para la especificación de requisitos ya que ha sido ampliamente aceptada por la comunidad industrial debido a su notación y aplicación sencilla. Esas propiedades

permiten a los *stakeholders*[2] entender los requisitos con facilidad, y contribuye a la elicitación y validación de los mismos. Otro factor que denota su popularidad es que UC es la única notación incluida en UML [13] para el modelado de requisitos.

De acuerdo con Cockburn [9], un caso de uso captura un contrato entre los *stakeholders* de un sistema sobre su comportamiento. El caso de uso describe el comportamiento del sistema bajo varias condiciones y responde a una petición de uno de los *stakeholders* llamado actor principal. El actor principal inicia una interacción con el sistema para cumplir un objetivo. El sistema responde, protegiendo os intereses de todos los *stakeholders*. Diferentes secuencias de comportamiento (o escenarios) pueden desarrollarse en función de las solicitudes particulares realizadas y de las condiciones que rodean a esas solicitudes. El caso de uso agrupa los diferentes escenarios.

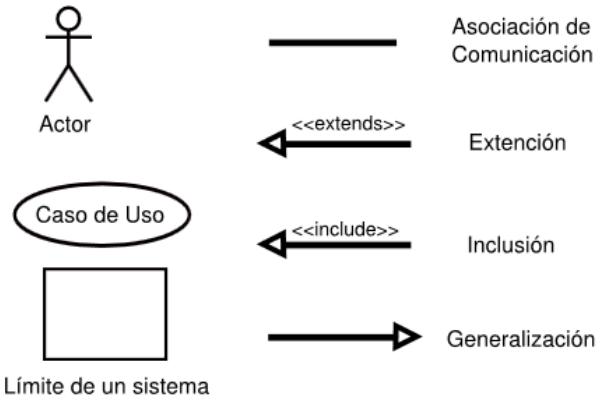

Figura 3: Componentes de un diagrama de casos de uso

Las relaciones entre los diferentes casos de usos y los actores se muestran en los diagramas de casos de uso de UML [13] (Figura 3). En este tipo de diagramas encontramos cuatro tipos de relaciones:

- *Inclusión*: una relación dirigida entre dos casos de uso, implicando que el comportamiento del primer caso de uso es insertado en el comportamiento del segundo.
- *Extensión*: el comportamiento del primer caso de uso puede ser insertado en el segundo bajo ciertas condiciones.
- *Generalización*: un caso de uso dado puede tener comportamientos, requisitos, restricciones y asunciones comunes con un caso de uso más general.
- *Asociación*: existe una asociación cuando un actor es involucrado en una interacción descrita por un caso de uso.

Usando estos diagramas podemos representar requisitos funcionales. No obstante, el diagrama de casos de uso propuesto por Booch [13] no nos proporciona la suficiente expresividad para representar requisitos no funcionales, existiendo una

[2] También llamados interesados o involucrados en un problema determinado que necesitan una solución óptima. Desde el punto de vista del desarrollo de sistemas, un *stakeholder* es aquella persona o entidad que está interesada en la realización de un proyecto o tarea, auspiciando el mismo ya sea mediante su poder de decisión o de financiamiento.

plantilla textual llamada *Especificación Suplementaria* como única alternativa para su representación. Esta falta de expresividad tiene asociados ciertos problemas como la pérdida de la trazabilidad durando el proceso de desarrollo software ya que no se proporciona un soporte automático para establecer la relación entre casos de uso y NFRs.

3.2.2. *Viewpoints*

En esta aproximación, el futuro sistema es definido de acuerdo al contexto donde realizará su principal computación. De esta forma, es definido considerando a todos los *stakeholders* involucrados y asignando un *viewpoint* (punto de vista) a cada uno. Cada *viewpoint* es un modelo que encapsula conocimiento parcial acerca del futuro sistema y del dominio, especificado en un esquema de representación particular y adecuado [10]. Por lo tanto, los requisitos del sistema son descritos usando una combinación de *viewpoints*, cada uno creado por una persona diferente implicada en el proceso de diseño (Figura 4).

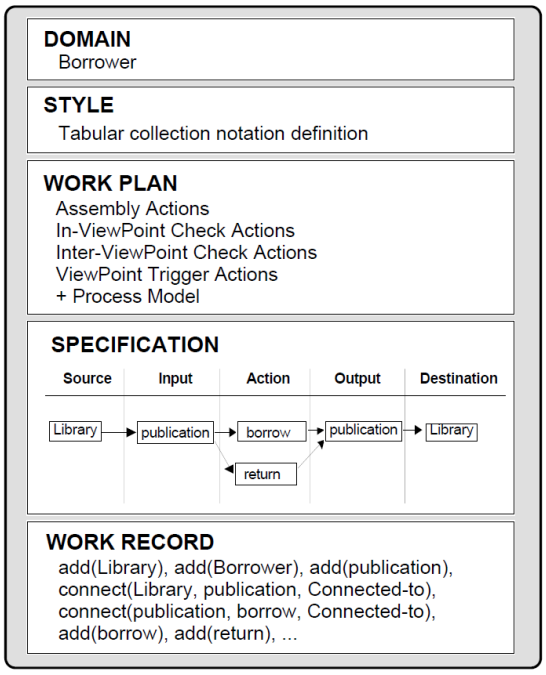

Figura 4: Ejemplo de *viewpoint*

No existe una notación estándar que pueda usarse para describir *viewpoints* pero cada propuesta identifica diferentes conceptos como relevantes para la descripción de un *viewpoint*. Una de las propuestas más aceptadas es la presentada por Nuseibeh [14], la cual usa cinco *slots*:

- Estilo: descripción del esquema de representación usado por el *viewpoint.*
- Plan de trabajo: descripción de las acciones de desarrollo, procesos y estrategia del *viewpoint.*
- Dominio: identifica el área de conocimiento del *viewpoint* con respecto al sistema en desarrollo.
- Especificación: describe el dominio del *viewpoint* con la notación descrita en el *slot* de estilo.

- Registro de trabajo: mantiene el estado del desarrollo y el historial de la especificación del viewpoint (en término de los planes de trabajo realizados).

Uno de los principales problemas de este enfoque es que permite a cada *stakeholder* usar una notación diferente, más apropiada para el dominio al que pertenece el *viewpoint*. Quizá, para evitar esta gran ambigüedad inherente a esta representación tan abierta, Finkelsetin [15] propone el uso de plantillas basada en *viewpoints* vacío con ciertos *slots* parcialmente rellenados. En concreto, define las siguientes plantillas: (i) Jerarquía funcional, (ii) Diagrama de bloques del sistema, (iii) Tablas de acción, (iv) Estructura de objetos y (v) Redes de Petri.

La principal ventaja que este enfoque posee es la facilidad para encontrar conflictos entre requisitos especificados por distintos *stakeholders*. Además, algunas notaciones proporcionan soporte a la trazabilidad de abajo a arriba y de arriba a abajo. De hecho, Nuseibeh [16] usan el Registro de trabajo para documentar cada acción o proceso sufridos por el viewpoint a través de su historia. Además, proporciona una notación específica para tratar con NFRs.

3.2.3. *Goal-Oriented*

En el contexto de la Ingeniería de Requisitos, el enfoque *Goal-Oriented* (GO) [17] ha demostrado ser útil en la elicitación y definición de requisitos. Las técnicas de análisis más tradicionales, como los Casos de Uso, solo se centran en establecer las características (por ejemplo, actividades y entidades) que un sistema debería soportar. No obstante, las propuestas GO se centran en el porqué de la construcción de un sistema, proporcionando la motivación y la lógica para justificar los requisitos del software. No solo son útiles para el análisis de objetivos (*goals*), sino también para elaborarlas y refinarlas.

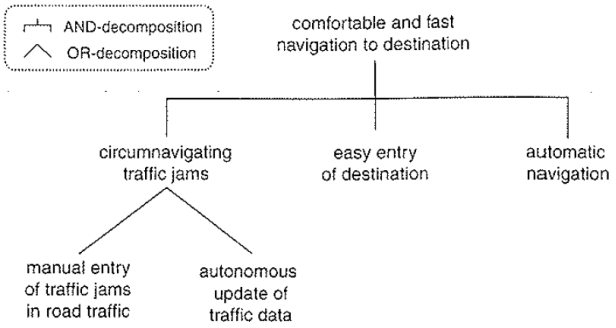

Figura 5: Ejemplo de Modelo de Objetivos

Un Modelo de Objetivos (*Goal Model*) se construye como un grafo dirigido por medio del un refinamiento de los objetivos del sistema (Figura 5). Este refinamiento acaba cuando los objetivos tienen la suficiente granularidad y detalle como para ser asignados a un agente (software o entorno) para que sean verificables en el futuro sistema. Este proceso de refinamiento se realiza usando relaciones de refinamiento AND/OR/XOR.

Hay un amplio número de propuestas que cubren desde la etapa de elicitación hasta la de validación en el proceso de la Ingeniería de Requisitos (véase [18] para una comparativa exhaustiva). Así, algunos conceptos son comunes a todas ellas:

- Un objetivo (*goal*) describe el porqué del desarrollo de un sistema desde el punto de vista del negocio, la organización o el sistema propiamente dicho. Para especificarlo hay que determinar tanto los objetivos funcionales (*functional goals*) (por ejemplo, los servicios esperados del sistema) así como los *softgoals* relacionados con la calidad de servicio, restricciones de diseño, etc.
- Un agente (agent) es cualquier componente activo, ya sea del sistema propiamente dicho o del entorno, cuya cooperación es necesaria para definir la operacionalización de un objetivo, o sea, la forma en la que el objetivo va a ser proporcionado por el futuro sistema. Esta operacionalización de los objetivos es aprovechada para mantener la trazabilidad a través del proceso de desarrollo software.
- Las relaciones de refinamiento (refinement relationships) son relaciones AND/OR/XOR que permiten la construcción del modelo de objetivos como un grafo dirigido. Esas relaciones se aplican por medio de un proceso de refinamiento (desde objetivos genéricos hasta sub-objetivos) hasta que se obtiene la suficiente granularidad para poder asignar operacionalizaciones específicas.

Hay que señalar que una de las principales ventajas de este enfoque es que introduce mecanismos para razonar sobre la especificación. Esto facilita el proceso de evaluar diseños o especificaciones alternativas del futuro sistema. Uno de ellos es el *framework* propuesto por Giorgini [19] que incluye relaciones AND/OR entre objetivos, pero también permite definir relaciones cualitativas entre objetivos, llamadas *contribuciones*, que describen en qué cantidad una operacionalización contribuye a alcanzar un objetivo. Además de esta formalización cualitativa, este *framework* cuenta con un algoritmo de propagación de etiquetas y semánticas cuantitativas para las nuevas relaciones.

Cysneiros y Yu también proponen un *framework* [20] con un gran poder expresivo para tratar con NFRs, trabajando con ellos desde las primeras etapas del desarrollo software. Este framework considera los NFRs como objetivos que pueden entrar en conflicto mutuamente y deben ser representados como *softgoals* a ser satisfechos. El concepto de *softgoal* se introdujo para hacer frente a la naturaleza abstracta e informal de los NFRs. La descomposición de los softgoals, así como su tratamiento es similar a los objetivos anteriormente mencionados.

A continuación se describirán tres enfoques GO: *NFR framework, KAOS* e *i* framework*.

NFR framework

Esta aproximación GO fue propuesta por Cysneiros y Yu [20] y tiene como objeto hacer frente a los NFRs, también conocidos como Requisitos de Calidad. Al contrario que los requisitos funcionales, los NFRs especifican restricciones sobre el sistema, así como consideraciones particulares de los factores de calidad que un sistema debería cumplir, como son la precisión, usabilidad, seguridad, rendimiento, fiabilidad o seguridad. Por lo tanto, se puede afirmar que mientras que los requisitos funcionales describen “qué” debería hacer el sistema, los NFRs establecen

restricciones sobre "cómo" realizar esas tareas. Como consecuencia, los NFRs están ligados a requisitos funcionales.

Para elicitar NFRs, los autores proponen el uso de una estrategia basada en *Language Extended Lexicon* (LEL) [21]. LEL está basado en un sistema de vocabulario controlado formado por *símbolos*, siendo cada uno de ellos una entrada expresada en términos de *nociones* y respuestas de *comportamiento*. Una noción registra el significado de un símbolo y sus relaciones fundamentales con otras entradas. Una respuesta de comportamiento especifica la connotación de un símbolo en el universo de discurso. Cada símbolo puede ser representado también por uno o más alias y será clasificado como un *sujeto*, *verbo* u *objeto*. Una vez que el Léxico es completado, se enriquece con los NFRs usando una base de conocimiento, presentada como catálogos, para guiar al analista a seleccionar los NFRs necesarios y sus operacionalizaciones relacionadas.

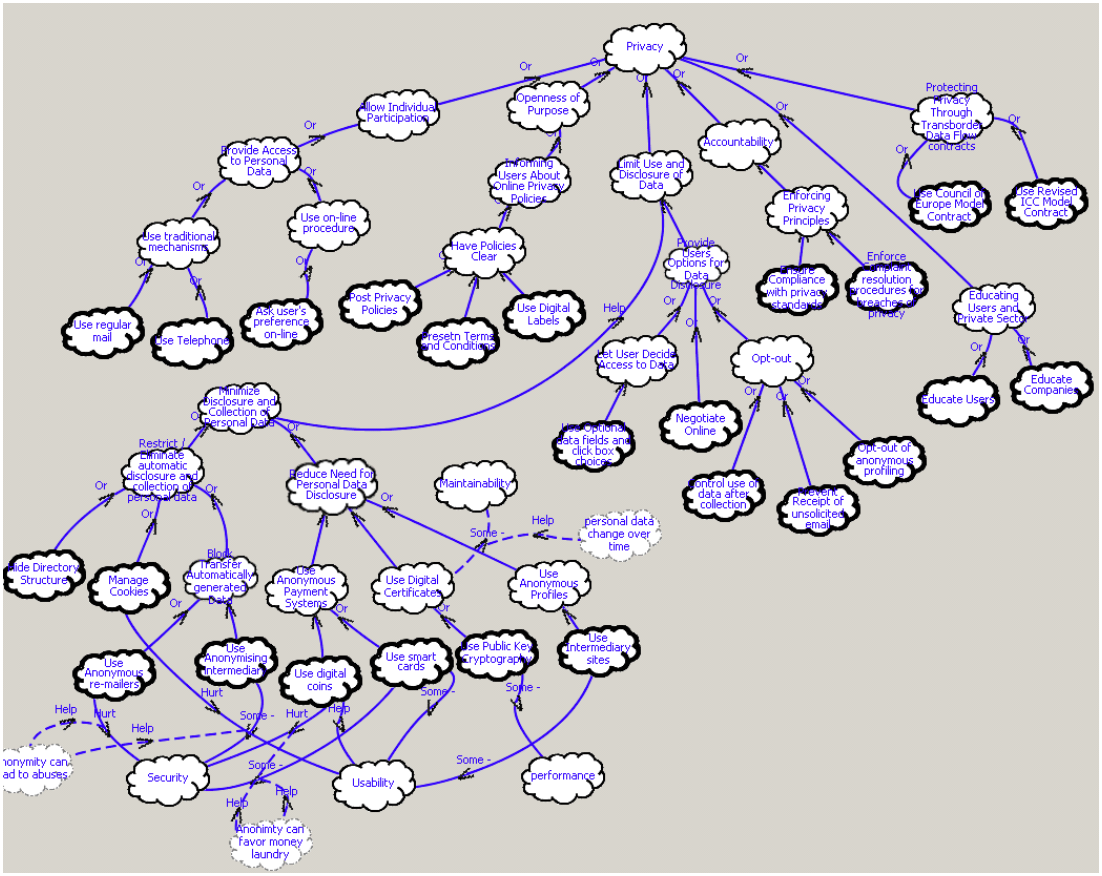

Figura 6: Modelo de un sistema con *NFR framework*

De acuerdo con este *framework*, los objetivos relacionados con los NFRs pueden entrar en conflicto entre ellos y deben ser representados como *softgoals* a satisfacer. Cada softgoal se descompone en sub-objetivos, representados por una estructura de grafo inspirada por los árboles AND/OR usados en la resolución de problemas (Figura 6). Esta descomposición se realiza por medio de enlaces de contribución, los cuales pueden ser categorizados como contribuciones *and* u *or*. Los enlaces de contribución permiten los NFRs hasta el punto en el que se puede afirmar que las operacionalizaciones de los NFRs relacionados se han cumplido. Los elementos del NFR framework pueden verse en la Figura 7.

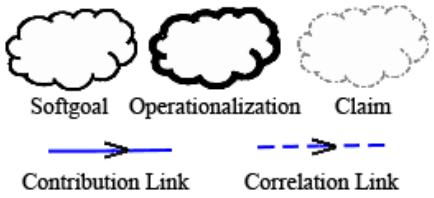

Figura 7: Elementos del *NFR framework*

Metodología KAOS

El lenguaje de modelado KAOS es la parte del *framework* KAOS [17] para elicitar, especificar y analizar objetivos, requisitos, escenarios y asignaciones de responsabilidad. Un modelo KAOS consta de seis vistas complementarias o sub-modelos (modelos de objetivos, obstáculos, objetos, agentes, operaciones y comportamiento) todos ellos relacionados por medio de enlaces de trazabilidad [22].

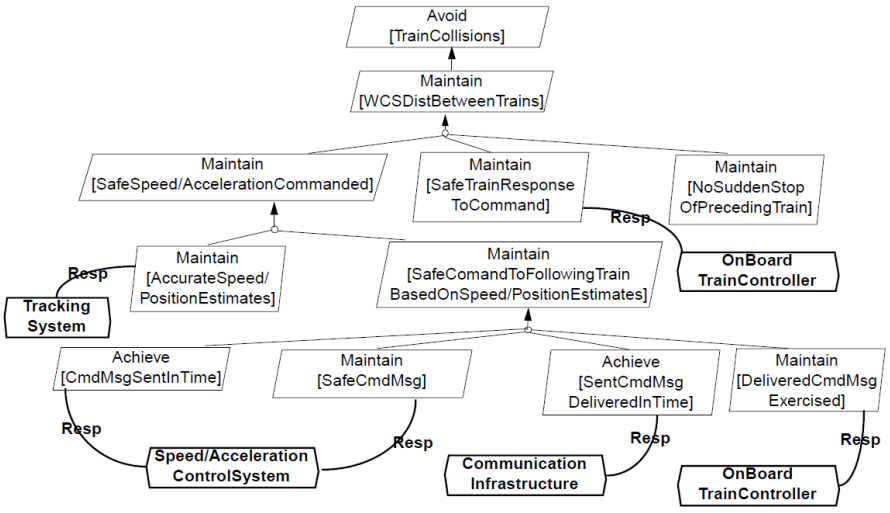

Figura 8: Modelo KAOS

La Figura 9 muestra las construcciones básicas para documentar las responsabilidades de los agentes sobre objetivos proporcionada por el *framework* KAOS. KAOS posee los siguientes elementos:

- Objetivo (*Goal*): Un objetivo describe un conjunto de comportamientos del sistema admisibles. Los objetivos han de ser definidos de una manera precisa para que pueda verificar se si el sistema los satisface o no.
- *Softgoal*: En KAOS, los *softgoals* se usan para documentar preferencias entre comportamientos del sistema alternativos. No hay criterio exacto de verificar la satisfacción de un *softgoal*, los cuales son satisfechos dentro de unos límites de aceptabilidad.
- Agente *(Agent)*: Los agentes definidos en KAOS están relacionados principalmente a usuarios y componentes de sistemas intensivos software. Por ello, un agente se define como un componente activo del sistema que tiene un rol específico para satisfacer un objetivo. Un agente puede ser agente humano, un dispositivo o un componente software.

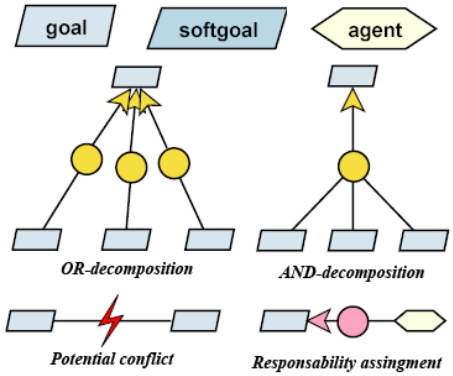

Figura 9: Construcciones básicas del *framework* KAOS

Las dependencias entre objetivos se representan en el modelo de objetivos de KAOS por medio de descomposiciones AND/OR y enlaces de conflicto. En KAOS, los objetivos pueden ser asignados a agentes por medio de enlaces de asignación de responsabilidad. Las dependencias entre objetivos son las siguientes:

- Descomposición AND/OR (*AND/OR descomposition*): Las descomposiciones AND/OR relacionan un objetivo con un conjunto de sub-objetivos, documentando que el objetivo es satisfecho si todos (AND) / alguno (OR) de los sub-objetivos es satisfecho.
- Conflicto potencial (*Potential conflict*): Este enlace documenta que satisfacer un objetivo puede evitar la satisfacción de otro objetivo, bajo determinadas condiciones.
- Asignación de responsabilidad (*Responsibility assignment*): Este enlace entre un objetivo y un agente significa que este agente es responsable de la satisfacción del objetivo.

i framework*

El *framework i** [23] consiste en un enfoque GO para tratar con los requisitos en varias fases del proceso de desarrollo software (fases de análisis de requisitos temprana y tardía, fases de diseño arquitectónico y estructural).

Durante el análisis temprano de requisitos, el ingeniero de requisitos obtiene y analiza las intenciones de los *stakeholders*. Esos requisitos son modelados como objetivos, los cuales, a través de algún análisis GO, derivarán en requisitos funcionales y no funcionales del futuro sistema. En *i**, se asume que los requisitos tempranos involucran a actores sociales que dependen los unos de los otros para satisfacer objetivos, realizar tareas, y utilizar recursos. El *framework i** incluye el *modelo de dependencias estratégicas* para describir la red de relaciones entre actores, así como el *modelo de lógica estratégica* (Figura 10) para describir y razonar sobre las anteriores relaciones.

El análisis tardío de requisitos deriva en una especificación de requisitos que describe los requisitos funcionales y no funcionales del futuro sistema. Durante la etapa de diseño arquitectónico se elegirá entre estilos arquitectónicos distintos usando como criterio las cualidades deseadas identificadas previamente en el proceso. En fase de diseño detallado se introducen detalles adicionales sobre los componentes arquitectónicos del sistema.

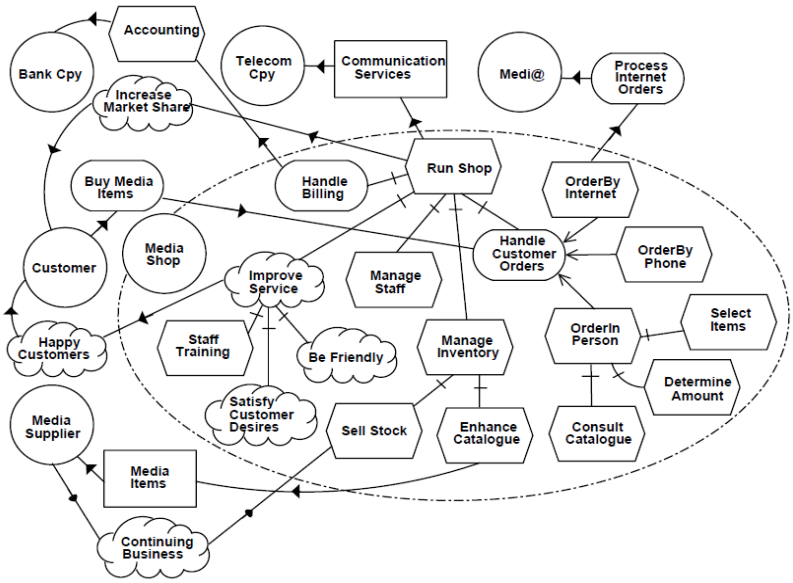

Figura 10: Modelo *i de lógica estratégica**

El *framework i** hace distinción entre dos tipos de elementos: objetos y relaciones (Figura 11). Los objetos considerados en i* son:

- Un actor es una persona o sistema que tiene una relación con el sistema a desarrollar. *i** refina este actor en tres tipos:
  - Un agente (*agent*) es un actor que tiene una representación física concreta, por ejemplo una persona o un sistema.
  - Un rol (*role*) define el comportamiento de un actor dentro de un contexto específico. Un actor puede tener varios roles, y un rol puede ser asignado a múltiples actores.
  - Una posición (*position*) es un conjunto de roles que pueden ser representados típicamente por un agente. Un agente puede representar varias posiciones.
- Un objetivo (*goal*) responde a "¿porqué?". Describe un cierto estado que un actor desearía alcanzar. No obstante, un objetivo no describe como debería ser alcanzado.
- Una tarea (*task*) especifica una forma particular de hacer algo. Normalmente, una tarea consiste en un número de pasos (o sub-tareas) que un actor debería realizar para ejecutar dicha tarea.
- Un recurso (*resource*) es una entidad (física o de información) que un actor necesita para conseguir un objetivo o realizar una tarea. El mayor interés sobre un recurso es si está disponible y para quién lo está.
- Un *softgoal* es una condición que a un actor le gustaría conseguir, aunque al contrario que para el concepto de objetivo (*hardgoal*), la condición para conseguirlo no está claramente definida.

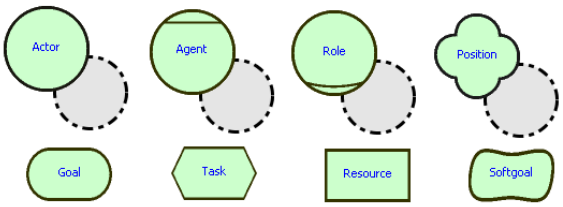

Figura 11: Objetos en *i

Los objetos anteriores están relacionados entre sí mediante el siguiente conjunto de relaciones (Figura 12):

- Una dependencia (*dependency*) en *i** documenta una relación entre dos actores. Un actor depende de otro para conseguir un objetivo, realizar una tarea o usar un recurso. *i** distingue cuatro tipos de dependencias: de objetivo, tarea, recurso o softgoal.
- Un *means-end link* documenta que softgoals, tareas y/o recursos contribuyen a conseguir un objetivo. Estos enlaces facilitan la documentación y la evaluación de las distintas formas de satisfacer un objetivo, por ejemplo, realizando varias descomposiciones de un objetivo en distintos softgoals, tareas y recursos.
- Un enlace de descomposición de tareas (*task decomposition link*) documenta los elementos esenciales de una tarea. Este enlace relaciona una tarea con sus componentes, los cuales pueden ser cualquier combinación de subobjetivos, sub-tareas, recursos o softgoals. La descomposición de una tarea abarca las sub-tareas que pueden realizarse, los sub-objetivos que deberían cumplirse, los recursos que se necesitan y los softgoals que típicamente definen objetivos de calidad para la tarea.
- Una contribución (*contribution*) documenta una influencia positiva o negativa de ciertos softgoals sobre otros softgoals o tareas.

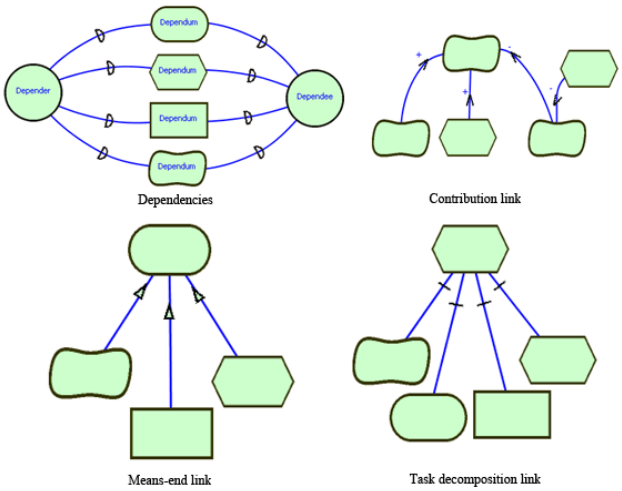

Figura 12: Relaciones en *i

4. Trabajo Fin de Máster – CSRML: un Lenguaje para la Especificación de Requisitos en Sistemas Colaborativos

Como trabajo final, se realizará un análisis de las distintas técnicas de especificación de requisitos descritas en el apartado anterior aplicadas a sistemas colaborativos. Así identificaremos cual es la técnica más adecuada para la representación de los requisitos de este tipo de sistema, de naturaleza no funcional y ligados con la colaboración, el *awareness* y la calidad.

Tras determinar cuál es la técnica más adecuada a trajes del modelado de un caso de estudio, la expenderemos para adaptarla a las peculiaridades de los sistemas colaborativos y realizaremos un experimento con usuarios para determinar su validez.

4.1. Análisis Empírico de Técnicas de Ingeniería de Requisitos para Sistemas Colaborativos

En este análisis se estudiará la aplicabilidad de tres técnicas de ingeniería de Requisitos (Casos de Uso [24], *Viewpoints* [15] y *Goal-Oriented* [20]) a la especificación de sistemas colaborativos, prestando especial atención a los requisitos de *awareness*. Para realizar este estudio, se especificarán los requisitos de awareness de un sistema real (Google Docs [12]). Una vez que el sistema sea modelado, re realizará un análisis empírico para comparar la tres técnicas anteriormente mencionadas.

4.1.1. Caso de estudio

Como caso de estudio para comprobar cómo de adecuadas son estas técnicas de Ingeniería de Requisitos se usará Google Docs [12] (Figura 13). Google Docs está formado por un procesador de texto, una hoja de cálculo y un editor de presentaciones y formularios gratuitos y basados en una aplicación web proporcionados por Google. Google Docs sirve como herramienta colaborativa para editar documentos que podrán ser compartidos, abiertos y editados por múltiples usuarios a la vez. Se ha seleccionado este sistema para el análisis porque es ampliamente conocido y posee de un claro enfoque colaborativo.

Como punto de partida para esta evaluación de las técnicas de requisitos, se identificarán aquellas soluciones de diseño para los requisitos de awareness de Google Docs a partir del conjunto de técnicas propuestas por Gutwin [25]. Estas técnicas, que serán comentadas en las siguientes sub-secciones, pueden ser encontradas como patrones para la colaboración entre usuarios en [26].

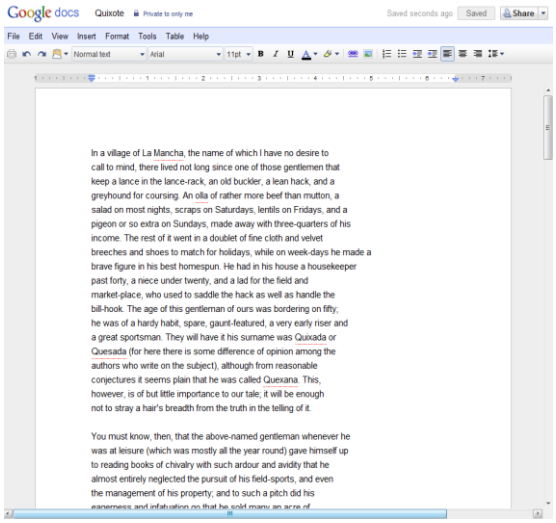

Figura 13: Interfaz de Google Docs

Cursores remotos

Esta técnica, basada en los telepunteros de Gutwin [25], nos permite ser conscientes de la posición del cursor del resto de usuarios, así como de si han seleccionado o no algún fragmento de texto (Figura 14). Así, cuando un usuario remoto está escribiendo, el resto de usuarios pueden comprobarlo en tiempo real. Cerca del cursor, el aparece el *nickname* del usuario. Además, si el usuario selecciona un texto, éste será remarcado cono el color del usuario.

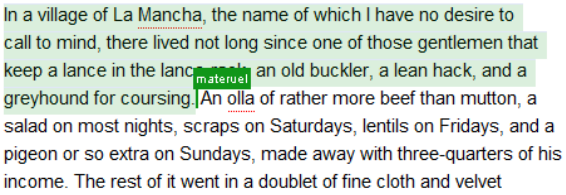

Figura 14: Cursor remoto y texto seleccionado remotamente

Lista de participantes y chat

Google Docs no implementa la técnica de avatares de Gutwin [25] propiamente dicha. En lugar de eso, muestra una lista de los participantes que están editando simultáneamente el mismo documento (Figura 15). Por medio de esta lista, los usuarios pueden comunicarse entre sí a través de un chat, que puede mostrar u ocultarse en cualquier momento. Además, usando la vista del chat, los usuarios conocen el color asignado a cada uno de sus colaboradores.

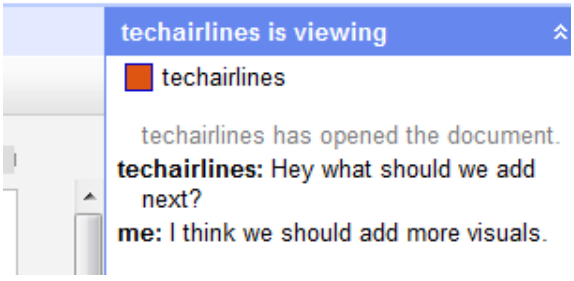

Figura 15: Dos usuarios chateando a través de la lista de participantes

Historial de revisiones

Las técnicas identificadas por Gutwin como *Expresando información sobre la autoría / sobre el pasado* [25] se usan para hacer disponible a los usuarios el historial de los cambios realizados. Estas técnicas han sido implementadas por Google Docs usando un historial de revisiones que permite al sistema mantener el seguimiento de todos los cambios realizados a los documentos. En este historial de revisiones, los cambios realizados por cada usuarios se denotan usando distintos colores. Además, si el cambio consiste en un borrado, el texto además será tachado. Esta funcionalidad puede ser activada o desactivada en cualquier momento. El historial de revisiones tiene dos niveles de detalle, dependiendo de la cantidad de información mostrada, que los que el usuario puede alternar cuando lo desee.

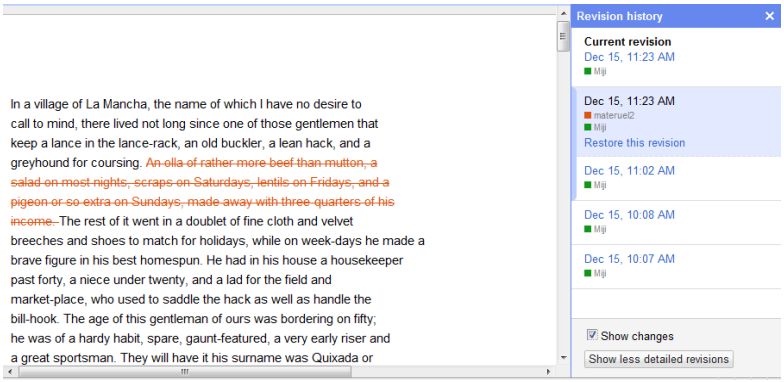

Figura 16: Historial de revisiones mostrando una eliminación de texto

4.1.2. Evaluación empírica

Para evaluar las distintas técnicas de Ingeniería de Requisitos, de modelarán los requisitos de *awareness* del caso de estudio usando cada una de las técnicas. Primero, hay que distinguir que las características de Google Docs pueden ser modeladas usando requisitos funcionales y no funcionales. Las técnicas de los *telepunteros* y los *avatares* resultan en NFRs ya que contribuyen a incrementar factores de calidad en uso, como la usabilidad. No obstante, la tercera característica (*Expresando información sobre la autoría / sobre el pasado*), a pesar de contribuir positivamente a los anteriormente mencionados factores de calidad, debe considerarse funcional, debido al almacenamiento de información histórica y a la función de reversión. Además, se asociarán se asociaran las funcionalidades de awareness tanto con las tres características de los sistemas colaborativos (colaboración, comunicación y coordinación) como con las características del modelo ISO/IEC 25010 (*Software engineering-Software product Quality Requirements and Evaluation,* SQuaRE) [27]. Este estándar ayudará a organizar correctamente la especificación del sistema siguiendo las recomendaciones de Moreira [28].

Modelando el caso de estudio

Tras analizar las características de Google Docs descritas anteriormente, y de acuerdo con el *framework* de Gutwin para sistemas colaborativos, se especificarán los requisitos funcionales del sistema (Tabla 3). A continuación, como puede observarse en la Tabla 4, cada funcionalidad de *awareness* detectada en el sistema

se relaciona con algunos factores de calidad del estándar SQuaRE para identificar los NFRs de Google Docs. Hay que señalar que se están describiendo parcialmente los requisitos de Google Docs para facilitar la comprensibilidad de la evaluación.

Tabla 3: Relación entre elementos de *awareness* y requisitos funcionales

Categoría	Elemento	Requisito funcional
Quién	Presencia	Saber quién está participando
Qué Dónde	Acción Localización	Ver las acciones de los otros usuarios
Quién Cuándo	Autoría Histórico de eventos	Mantener la autoría de los cambios

Tabla 4: Relación entre factores de calidad y funcionalidades de *awareness*

Factor de calidad	Funcionalidad de *awareness*
Adecuación funcional	Histórico de revisiones Telepunteros Lista de participantes
Fiabilidad	Histórico de revisiones
Eficiencia	Telepunteros
Operatividad	Telepunteros Lista de participantes
Seguridad	Histórico de revisiones

Goal-Oriented

Para llevar a cabo la especificación de Google Docs, se usará la notación i* [29], dado que es uno de los enfoques GO más aceptados. Usando esta notación, se especificará cada uno de los factores de calidad de SQuaRE identificados en la Tabla 4 como los *softgoals* raíz del sistema (Figura 17).

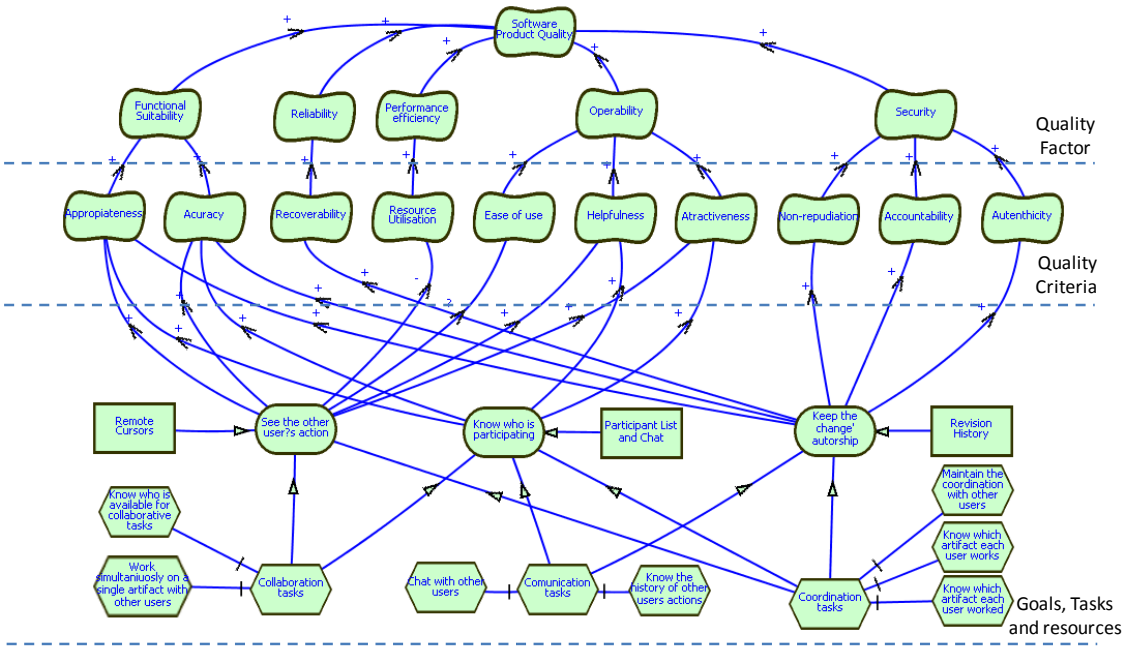

Figura 17: Modelo *Goal-Oriented*

Estos *softgoals* se refinarán seleccionando aquellos factores de calidad de SQuaRE más apropiados para el sistema. Cada una de las funcionalidades de *awareness* se especificará como un recurso proporcionado por el sistema que contribuye positivamente a satisfacer alguno de los *softgoals*, o sea, alguno de los factores de calidad. Debe notarse que alguno de ellos contribuye negativamente debido a las restricciones que impone Este es el caso de los cursores remotos, los cuales incrementan la utilización de recursos. Además, la facilidad de uso depende, entre otros factores, de la experiencia del usuario en este tipo de sistemas.

Los tres requisitos funcionales identificados en la Tabla 3 se especificarán como objetivos del sistema que tendrán dependencias con los recursos. También se especificará cómo las técnicas de *awareness* contribuyen positivamente a los aspectos funcionales de los sistemas colaborativos especificadas como tareas en el modelo.

Casos de Uso

Para representar el sistema usando esta técnica se tendrá que hacer frente a un problema: la falta de expresividad de los diagramas de casos de uso (tal y como están definidos en UML [13]) para describir NFRs. Así, si se usa esta notación, la única alternativa es describir los requisitos funcionales como se muestra en la Figura 18a y utilizar un documento diferente, como la especificación suplementaria, pala los NFRs. Esta alternativa presenta varia limitaciones, como la escasa trazabilidad entre requisitos funcionales y no funcionales.

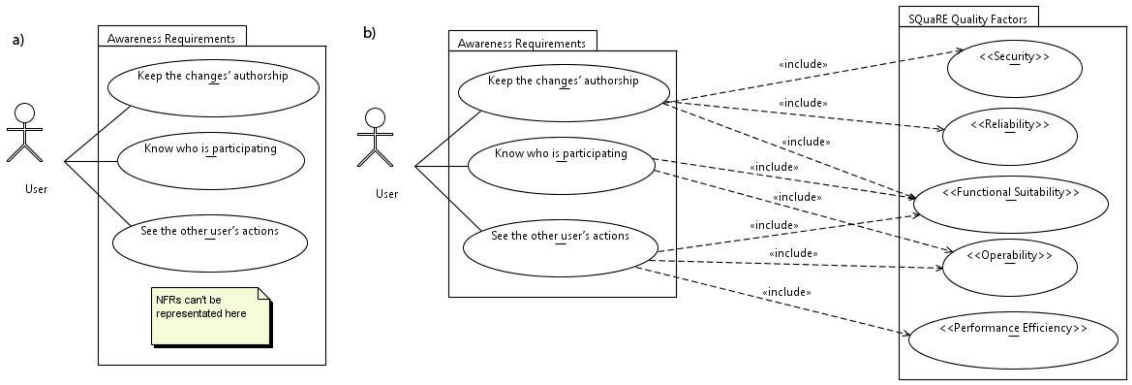

Figura 18: Diagrama de casos de uso con extensión para NFRs

Para solventar esta falta de expresividad de los casos de uso, se usará la extensión propuesta por Moreira [30] que permite describir algunos estereotipos para describir factores de calidad como la seguridad o la fiabilidad que pueden ser aplicados a los casos de uso para especificar NFRs. Usando esta extensión, los requisitos del sistema son especificados como se muestra en la Figura 18b. Como puede observarse, esta alternativa nos permite describir NFRs y trazarlos adecuadamente.

Viewpoints

Para aplica esta técnica, es necesario definir en primer lugar el estilo de representación para especificar los *viewpoints* del sistema. Para ello, se utilizará un estilo que permita relacionar las funcionalidades de *awareness* con los factores de calidad. Por ello, se definirá un estilo de representación para el *viewpoint* formado por dos objetos (factores de calidad y requisitos de *awareness*) y dos relaciones

entre esos objetos (composiciones y contribuciones). Además, se debe establecer el dominio de aplicación para aplicar la técnica.

Teniendo en cuenta el este estilo de representación, el *viewpoint* del ingeniero de calidad se define como muestra la Figura 19. Como puede observarse, los factores de calidad se definen por medio de un árbol cuyas hojas son las técnicas de awareness de Google Docs. De esta manera pueden establecerse relaciones directas entre los factores de calidad y la funcionalidad del sistema.

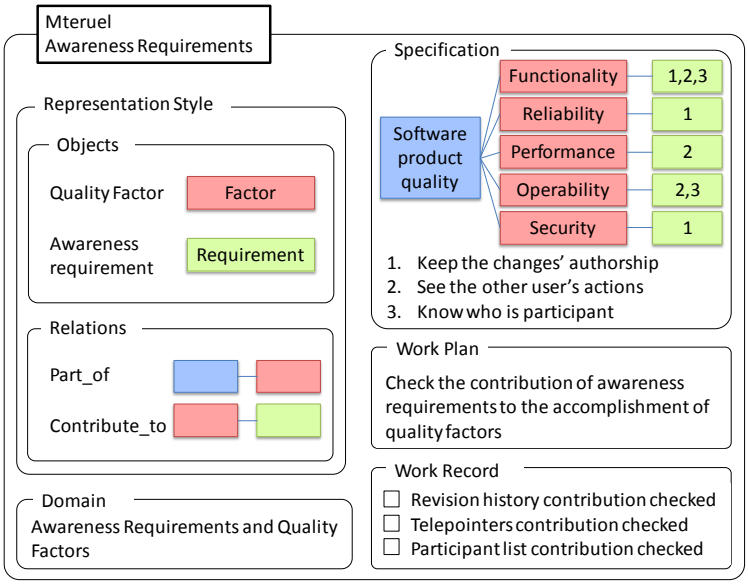

Figura 19: *Viewpoint* para los requisitos de *awareness* y los factores de calidad (adaptado de [14])

Evaluando las técnicas de Ingeniería de Requisitos

Usando como entrada las diferentes especificaciones del sistema, la evaluación de las diferentes técnicas de Ingeniería de Requisitos se llevará a cabo usando *DESMET* [31], un conjunto de técnicas aplicable a la evaluación de métodos y herramientas de Ingeniería del Software. Se usará el método basado en un caso de estudio cualitativo que describirá una evaluación basada en características. Siguiendo las líneas guía de esta técnica, se preparará una lista de características inicial que una técnica de Ingeniería de Requisitos para sistemas colaborativos debería proporcionar (Tabla 5).

Una vez que se rellena la Tabla 5, *DESMET* establece que ha de asignarse un grado de importancia a cada característica identificada. Específicamente, los grado a aplicar son:

- O: Obligatorio
- A: Altamente deseable
- D: Deseable
- R: Recomendable

Usando estos grados, se rellenará la Tabla 6. Como puede observarse, las características más importantes son las relacionadas con la representación del *awareness* y los factores de calidad.

Tabla 5: Lista de características para la evaluación de las técnicas

Característica	Descripción
Representación del *awareness*	El modelo debe permitir representar las características de awareness del sistema
Representación de los factores de calidad	El modelo debe representar las (sub)características de SQuaRE
Representación de los NFRs	El modelo debe ser capaz de representar NFRs gráficamente
Representación jerárquica	La relación entre los elementos del modelo debe ser jerárquica
Representación estándar	El modelo debe estar basado en una representación estándar ampliamente extendida
Complejidad del modelo	La complejidad del modelo no debería ser demasiado alta
Modelo cuantitativo	El modelo debe permitir cuantificar las relaciones entre los elementos representados
Trazabilidad	Los requisitos representados deben ser trazables a lo largo del proceso de desarrollo software

Tabla 6: Importancia de las características

Característica	Importancia
Representación del *awareness*	O
Representación de los factores de calidad	O
Representación de los NFRs	A
Representación jerárquica	A
Representación estándar	D
Complejidad del modelo	D
Modelo cuantitativo	D
Trazabilidad	R

A continuación, de acuerdo con *DESMET*, ha de proporcionarse una escala para evaluar cada una de las características descritas. Usaremos la escala propuesta por *DESMET* para evaluar cada característica según los siguientes factores:

- UA: Umbral de aceptabilidad
- PO: Puntuación obtenida

Tabla 7: Escala para evaluar el soporte de una característica (de [31]**)**

Generic scale point	Definition of Scale point	Scale Point Mapping
Makes things worse	Cause Confusion. The way the feature is represented makes difficult its modelling and/or encourage its incorrect use	-1
No support	Fails to recognise it. The approach are not able to model a certain feature	0
Little support	The feature is supported indirectly, for example by the use of other model/approach in a non-standard combination	1
Some support	The feature is explicitly in the feature list of the model. However, some aspects of feature use are not catered for.	2
Strong support	The feature is explicitly in the feature list of the model. All aspects of the feature are covered but its use depends on the expertise of the user	3
Very strong support	The feature is explicitly in the feature list of the model. All aspects of the feature are covered and the approach provides a guide to assist the user	4
Full support	The feature appears explicitly in the feature list of the model. All its aspects are covered and the approach provides a methodology to assist the user	5

Una vez que cada característica sea evaluada, la diferencia entre los factores UA y PO será computada, como se muestra en la columna *Diferencia* (Dif) en las Tablas 8, 9, 10 y 11.

A continuación, hay que resaltar que se ha usado una variación del método *DESMET*. La *Importancia* (Imp) de cada característica ha sido establecida sobre una escala desde uno a cuatro (Recomendable – 1, Deseable – 2, Altamente deseable – 3 Obligatorio – 4). La importancia se usará para computar el resultado final de cada característica multiplicando la importancia por la diferencia. Esta operación se muestra en la columna *Resultado* (Res). Finalmente, el resultado final de cada técnica se obtendrá sumando los resultados de todas las características. Este *framework* será utilizado para evaluar las diferentes técnicas de Ingeniería de Requisitos estudiadas.

Tabla 8: Evaluación para *Goal-Oriented*

Característica	Imp	UA	PO	Dif	RES
Representación del *awareness*	4	5	5	0	0
Representación de los factores de calidad	4	4	5	1	4
Representación de los NFRs	3	3	5	2	6
Representación jerárquica	3	3	3	0	0
Representación estándar	2	2	1	-1	-2
Complejidad del modelo	2	2	3	1	2
Modelo cuantitativo	2	2	2	0	0
Trazabilidad	1	1	1	0	0
Total					**10**

Tabla 9: Evaluación para Casos de Uso

Característica	Imp	UA	PO	Dif	RES
Representación del *awareness*	4	5	2	-3	-12
Representación de los factores de calidad	4	4	0	-4	-16
Representación de los NFRs	3	3	1	-2	-6
Representación jerárquica	3	3	1	-2	-6
Representación estándar	2	2	0	-2	-4
Complejidad del modelo	2	2	0	-2	-4
Modelo cuantitativo	2	2	5	3	6
Trazabilidad	1	1	3	3	3
Total					**-39**

Tabla 10: Evaluación para Casos de Uso con extensión NFR

Característica	Imp	UA	PO	Dif	RES
Representación del *awareness*	4	5	2	-3	-12
Representación de los factores de calidad	4	4	0	-4	-16
Representación de los NFRs	3	3	1	-2	-6
Representación jerárquica	3	3	1	-2	-6
Representación estándar	2	2	0	-2	-4
Complejidad del modelo	2	2	0	-2	-4
Modelo cuantitativo	2	2	5	3	6
Trazabilidad	1	1	3	3	3
Total					**-39**

Tabla 11: Evaluación para *Viewpoints*

Característica	Imp	UA	PO	Dif	RES
Representación del *awareness*	4	5	1	-4	-16
Representación de los factores de calidad	4	4	1	-3	-12
Representación de los NFRs	3	3	0	-3	-9
Representación jerárquica	3	3	1	-2	-6
Representación estándar	2	2	0	-2	-4
Complejidad del modelo	2	2	2	0	0
Modelo cuantitativo	2	2	0	-2	-4
Trazabilidad	1	1	1	0	0
Total					**-51**

La Figura 20 muestra gráficamente los resultados contenidos por cada una de las técnicas. Como puede observarse, la aproximación *Goal-Oriented* es la única que obtiene un resultado positivo. A pesar de este resultado, fue negativamente evaluada respecto al Modelo cuantitativo, ya que *i** solo proporciona un soporte parcial para cuantificar las relaciones entre requisitos cuando se usan enlaces de contribución. La técnica de Casos de Uso falla básicamente en la descripción del modelo de *awareness*, ya que no soporta NFRs. También falla en el Modelo cuantitativo, ya que no proporciona ninguna asistencia en ese sentido. Además, ninguna proporciona soporte a la Representación jerárquica. La limitación sobre los NFRs es solventada cuando se usa la extensión NFR, aunque no proporciona ninguna mejora para los otros dos resultados negativos. Finamente, la técnica menos adecuada para este problema son los Viewpoints, obteniendo un mal resultado para la mayoría de las características analizadas.

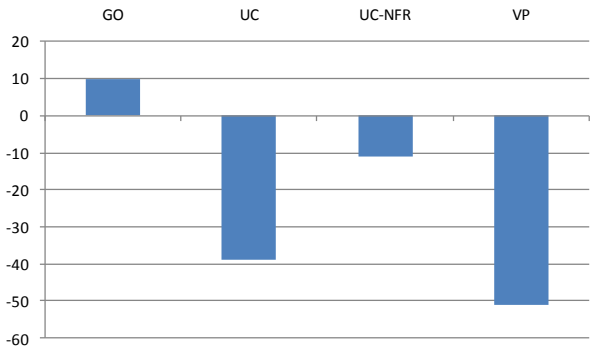

Figura 20: Resultados del análisis empírico

Adicionalmente, tal y como sugiere *DESMET*, se realizó una comparativa del porcentaje de cada característica satisfecho por cada técnica (Figura 21). Hay que señalar que ninguna de las técnicas analizadas es demasiado compleja, permitiendo a los *stakeholders* transferir información acerca del sistema de una forma fácil e intuitiva. La técnica de Casos de Uso destaca sobre las demás debido a la estandarización de su notación. En relación a la representación jerárquica, todas las técnicas analizadas excepto Casos de Uso, proporcionan algún tipo de soporte que ayuda a organizar las especificaciones de los sistemas. Cabe destacar que la técnica *Goal-Oriented* es la única que da cierto soporte al modelo cuantitativo, a pesar de la relevancia de esta característica, la cual ayuda a analizar

la especificación de los requisitos. En cuanto a la trazabilidad, Casos de Uso es la técnica más destacada, debido a su integración en RUP [32]. Tanto Goal-Oriented como Casos de Uso con la extensión NFR destacan en la representación de los NFRs y de los factores de calidad, siendo esto una necesidad de la especificación de los sistemas colaborativos. Desafortunadamente, la representación del *awareness* no está bien soportada por las técnicas analizadas, a pesar de ser una de las características más importante de éste tipo de sistemas. A la luz del análisis de los resultados, Goal-Oriented parece ser el enfoque más prometedor para la especificación de los sistemas colaborativos.

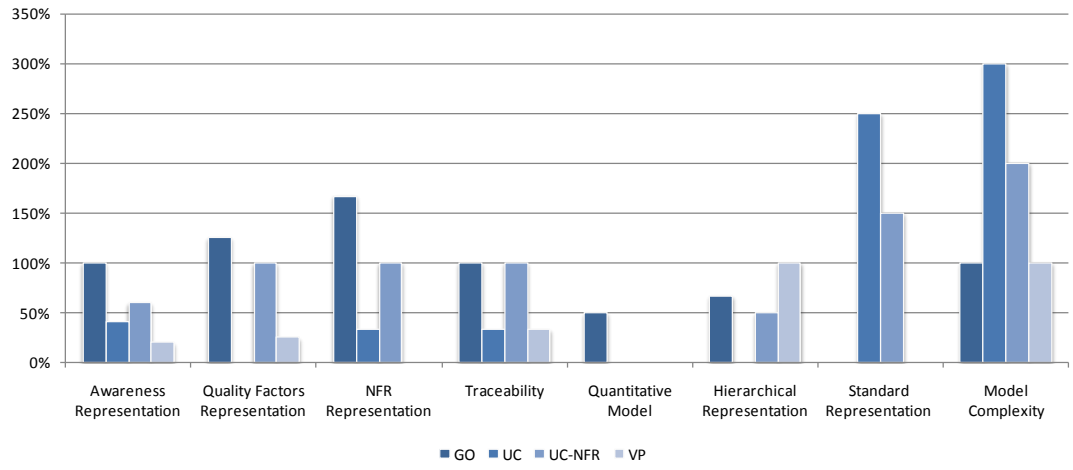

Figura 21: Resultados relativos a las distintas características

4.2. Comparativa de Enfoques *Goal-Oriented* para el Modelado de Requisitos para Sistemas Colaborativos

Una vez que se determina que *Goal-Oriented* es la técnica más adecuada para modelar requisitos de sistemas colaborativos, se estudiará la aplicabilidad de tres enfoques GO (*NFR framework* [20], *i* framework* [33] y la metodología KAOS [17]) a este tipo de sistemas, prestando especial atención a los requisitos de *awareness*.

4.2.1. Evaluación empírica

Al igual que en sub-apartado anterior, se usará Google Docs [12] como caso de estudio y se realizará un estudio empírico similar.

Modelando el caso de estudio

Al igual que en caso anterior, se modelará el caso de estudio con los tres enfoques GO a estudiar. Nótese que se prescindirá del modelado con *i** debido a que dicho modelo es similar al mostrado en el apartado 4.1.2 (técnica *Goal-Oriented*).

NFR framework

En esta aproximación, los factores de calidad de SQuaRE [27] han sido modelados usando *softgoals*. No obstante, se ha usado el estándar SQuaRE en lugar de las colecciones de NFRs propuestas en la definición [11] para crear jerarquías de NFRs. Así, puede observarse el impacto que las sub-características de calidad tienen en las características principales por medio de enlaces de contribución. De la misma manera, cada característica contribuye a conseguir la calidad del producto software (Figura 22).

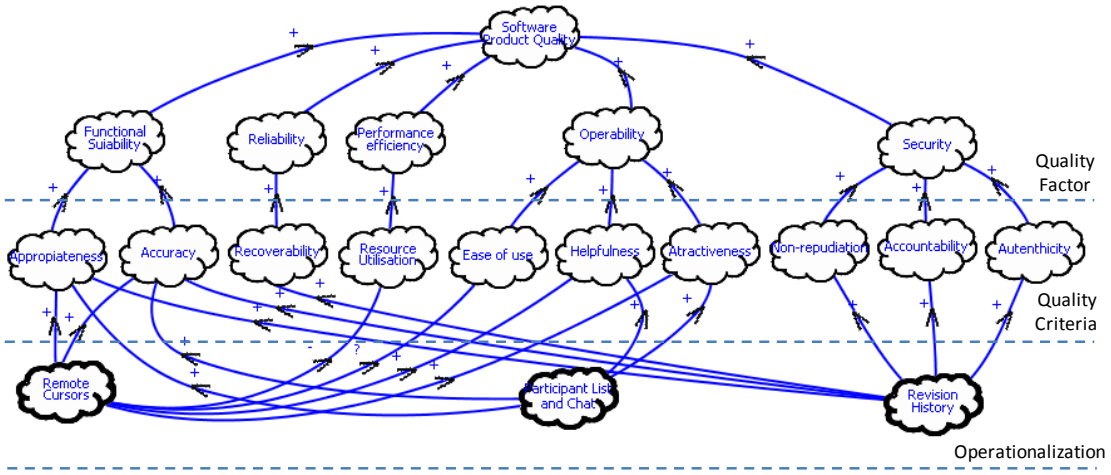

Figura 22: Modelo *Goal-Oriented* NFR

El problema aquí es que no somos capaces de representar requisitos funcionales (ya que el objetivo de este modelo son los NFRs), por lo que las tres tareas generales de los sistemas colaborativos (colaboración, comunicación y coordinación) no pueden ser definidas. Esta falta de expresividad conlleva a la obtención de una representación incompleta de los requisitos del sistema, por lo que deben usarse modelos adicionales, o extender este *framework*.

Metodología KAOS

Para modelar el sistema usando esta metodología, y a diferencia del caso de *i**, se descompondrá el modelo en tres sub-modelos, como puede verse en la Figura 23. Así, los modelos individuales representan (a) objetivos de *awareness*, (b) objetivos de sistemas colaborativos y (c) objetivos de calidad software.

Estos diagramas (Figura 23) muestran los tres principales objetivos y su descomposición en sub-objetivos. La técnicas de *awareness* implementadas han sido representadas usando agentes, ya que este elemento es usado para representar asignaciones de responsabilidad en KAOS.

Además, la Figura 23c ilustra un conflicto potencial entre dos *softgoals* relacionados con dos sub-factores de calidad. Esto es debido a que una interfaz de usuario muy atractiva causará una mayor utilización de recursos. Este conflicto queda denotado en el gráfico mediante un rayo de color rojo.

Evaluando los enfoques Goal-Oriented

Al igual que en el caso anterior nos basaremos en *DESMET* [31] para evaluar los tres enfoques. Para ello, estableceremos la lista de características a evaluar y la importancia de las mismas (Tabla 12 y Tabla 13). Los resultados del análisis se muestran en las Tablas 14, 15 y 16.

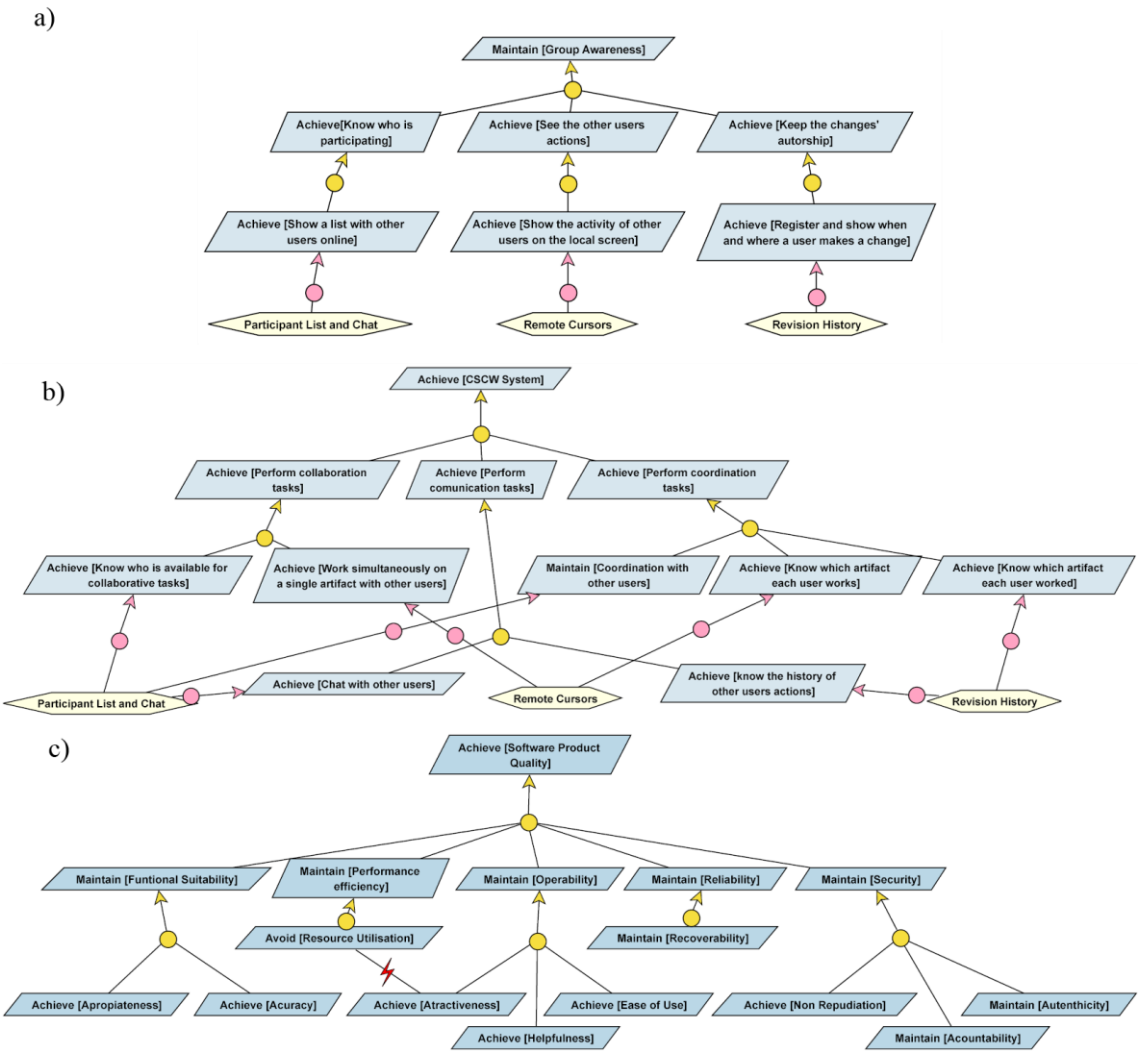

Figura 23: Modelos KAOS de Objetivos y Responsabilidades

Tabla 12: Lista de características para la evaluación de los enfoques GO

Característica	Descripción
Representación de los FRs y NFRs	El modelo debe ser capaz de representar FRs y NFRs gráficamente y diferenciar entre ambos
Características de los sistemas colaborativos	El modelo debe representar características de colaboración, comunicación y coordinación
Representación del *awareness*	El modelo debe permitir representar las características de awareness del sistema
Representación de los factores de calidad	El modelo debe representar las (sub)características de SQuaRE
Importancia de los requisitos	El modelo debe representar la importancia y preferencia entre requisitos
Representación jerárquica	La relación entre los elementos del modelo debe ser jerárquica
Complejidad del modelo	La complejidad del modelo no debería ser demasiado alta
Modelo cuantitativo	El modelo debe permitir cuantificar las relaciones entre los elementos representados
Trazabilidad	Los requisitos representados deben ser trazables a lo largo del proceso de desarrollo software

Tabla 13: Importancia de las características de los enfoques GO

Característica	Importancia
Representación de los FRs y NFRs	O
Características de los sistemas colaborativos	O
Representación del *awareness*	O
Representación de los factores de calidad	A
Importancia de los requisitos	A
Representación jerárquica	A
Complejidad del modelo	D
Modelo cuantitativo	D
Trazabilidad	R

Tabla 14: Evaluación para *NFR framework*

Característica	Imp	UA	PO	Dif	RES
Representación de los FRs y NFRs	4	5	3	-2	-8
Características de los sistemas colaborativos	4	4	1	-3	-12
Representación del *awareness*	4	4	4	0	0
Representación de los factores de calidad	3	3	5	2	6
Importancia de los requisitos	3	3	0	-3	-9
Representación jerárquica	3	3	3	0	0
Complejidad del modelo	2	2	1	-1	-2
Modelo cuantitativo	2	2	3	1	2
Trazabilidad	1	1	3	2	2
Total					**-21**

Tabla 15: Evaluación para *i* framework*

Característica	Imp	UA	PO	Dif	RES
Representación de los FRs y NFRs	4	5	5	0	0
Características de los sistemas colaborativos	4	4	5	1	4
Representación del *awareness*	4	4	5	1	4
Representación de los factores de calidad	3	3	5	2	6
Importancia de los requisitos	3	3	0	-3	-9
Representación jerárquica	3	3	3	0	0
Complejidad del modelo	2	2	1	-1	-2
Modelo cuantitativo	2	2	3	1	2
Trazabilidad	1	1	1	0	0
Total					**5**

Tabla 16: Evaluación para la metodología KAOS

Característica	Imp	UA	PO	Dif	RES
Representación de los FRs y NFRs	4	5	5	0	0
Características de los sistemas colaborativos	4	4	4	0	0
Representación del *awareness*	4	4	4	0	0
Representación de los factores de calidad	3	3	4	1	3
Importancia de los requisitos	3	3	0	-3	-9
Representación jerárquica	3	3	4	1	3
Complejidad del modelo	2	2	0	-2	-4
Modelo cuantitativo	2	2	4	2	4
Trazabilidad	1	1	2	1	1
Total					**-2**

La Figura 24 muestra gráficamente los resultados obtenidos por cada uno de los enfoques GO. Como puede observarse, el enfoque *i** es el único que obtiene un resultado positivo. A pesar de este resultado, fue negativamente evaluada respecto al modelo cuantitativo, ya que *i** solo proporciona un soporte parcial para cuantificar las relaciones entre requisitos cuando se usan enlaces de contribución. Este enfoque también falla en la representación de la importancia de los requisitos, no permitiendo representar si un requisito es más importante que otros. No obstante, esta falta de expresividad es compartida por los otros dos enfoques analizados. KAOS también falla en las mismas características que *i** pero, a diferencia de éste, KAOS obtiene un resultado inferior (o igual) en casi todas las características excepto en la representación jerárquica, gracias a su representación basada en árboles. Finalmente, *NFR framework* es el enfoque menos adecuado, obteniendo una puntuación muy baja debido a su falta de expresividad para especificar requisitos funcionales y a su falta de adaptabilidad para representar las características de los sistemas colaborativos.

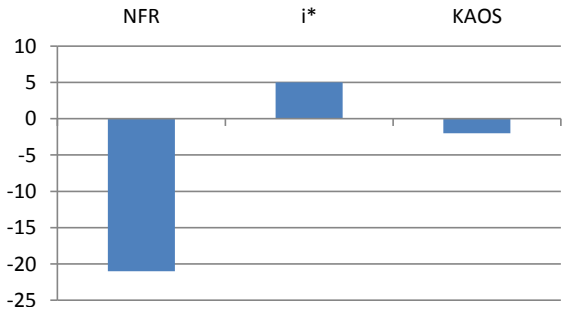

Figura 24: Resultado del análisis para los enfoques GO

Como en el caso anterior, se realizó una comparativa del porcentaje de cada característica satisfecho por cada enfoque analizado. La Figura 25 ilustra que *NFR framework* solo supera a sus competidores en la complejidad del modelo, debido a las simplicidad de sus modelos. De forma similar, KAOS supera a *i** en esta característica debido a que *i** posee más elementos de modelado en pro de su expresividad. Además, la evaluación de las características relacionadas con la representación jerárquica y trazabilidad es mejor en KAOS que en *i** ya que los modelos del último suelen definirse siguiendo una estructura de red, no proporcionando un soporte suficientemente sofisticado para la trazabilidad. Otro resultado importante es que ninguno de los enfoques es capaz de representar la importancia de los requisitos, algo a considerar en futuros trabajos. También resalta que, a pesar de que *i** y KAOS obtienen la misma puntuación para la característica de representación de FRs y NFRs, *i** supera a KAOS en la mayoría de las características importantes (las obligatorias y las altamente deseables), a excepción de la característica de trazabilidad. No obstante, KAOS obtiene un mejor resultado para las características menos valoradas, como la representación jerárquica y la complejidad de los modelos.

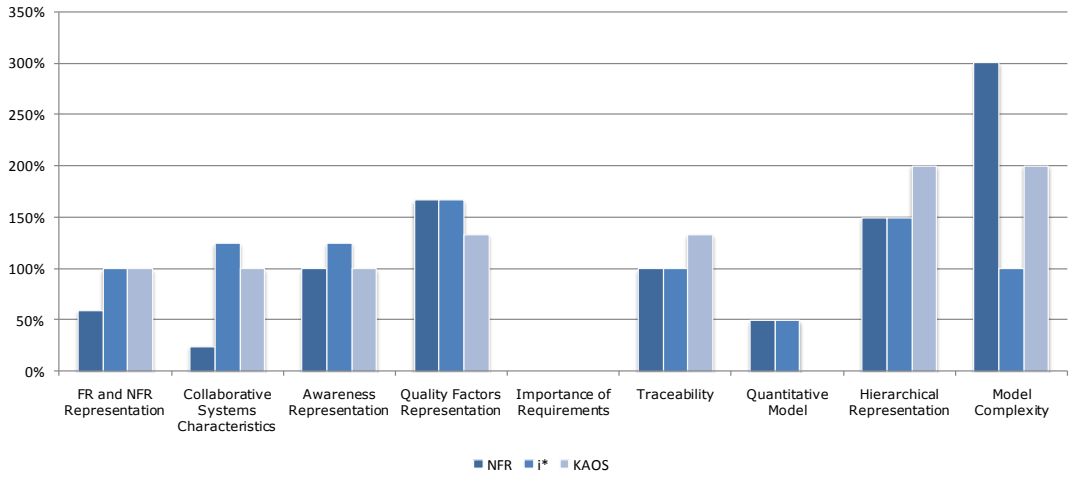

Figura 25: Resultados relativos a las distintas características para los enfoques GO

4.3. Especificando Requisitos de Sistemas Colaborativos con CSRML

Una vez que se determina por medio de estudios empíricos que las técnicas de Ingeniería de Requisitos más adecuadas para especificar los requisitos de este tipo de sistemas son las aproximaciones *Goal-Oriented*, y más concretamente, la aproximación *i**, se desarrollará CSRML, una extensión de *i** para abordar la especificación de requisitos de sistemas en los que la colaboración y la conciencia de la presencia y/o las acciones de los otros usuarios es crucial. Para validar esta propuesta, se ha llevará a cabo un caso de estudio en el que se modela un sistema de gestión de congresos en el que la revisión se realiza colaborativamente entre los distintos revisores.

4.3.1. CSRML: un lenguaje de modelado de requisitos para sistemas colaborativos

Para tratar con el tipo especial de requisitos de los sistemas colaborativos, y basándonos en los dos estudios anteriores (Apartados 4.1 y 4.2), se desarrolla CSRML (*Collaborative Systems Requirements Modelling Language*). Este lenguaje es una extensión de *i** que incluye algunos elementos para modelar las características especiales de los sistemas colaborativos. Los elementos de CSRML (Figura 26) excluyendo aquellos cuyo significado es el mismo que en *i**, son:

- Un *rol* (*role*) representa el conjunto de tareas que pueden ser realizadas por el actor que asuma dicho rol. La diferencia entre *i** y CSRML es que un actor que representa un rol puede participar en tareas individuales y colaborativas (mediante enlaces de participación) y puede ser responsable de la consecución de un objetivo (a través de enlaces de responsabilidad). Además, la notación gráfica también varía respecto al rol de *i**. Además, CSRML no comparte el concepto de límite (*boundary*) de actor/rol de *i**, a fin de facilitar la descripción de tareas colaborativas.
- Un *actor* es un usuario, programa o entidad con ciertas capacidades adquiridas que puede interpretar un rol responsable de ciertas acciones [34]. Un actor tiene que representar un rol (lo cual se especifica mediante un enlace de representación, ver Figura 26) para participar en el sistema.
- Una *tarea* (*task*) en CSRML tiene el mismo significado que en *i**. No obstante, se diferencia en la notación introducida para definir la importancia

de una tarea: uno, dos o tres signos de exclamación, dependiendo de la importancia de la tarea. Además, CSRML identifica dos tipos especiales de tareas:

- Tarea abstracta (*abstract task*): Este tipo de tarea es una abstracción de un conjunto de tareas concretas y, posiblemente, de otros elementos, como objetivos, recursos o *softgoals*. No pueden asignarse enlaces de participación directamente a este tipo de tareas.
- Tarea concreta: Estas son las tareas en las que están involucrados los participantes, los cuales serán asignados a estas tareas mediante los enlaces de participación. Las tareas abstractas se refinan en estas tareas. A su vez, se subdividen en cuatro tipos: (i) *tareas individuales*, que los actores pueden realizar sin ningún tipo de interacción con otros actores; (ii) *tareas de colaboración*; (iii) *tareas de comunicación*; y (iv) *tareas de* coordinación. Los últimos tres tipos responden a tareas en las que dos o más actores participan.
- Un *softgoal de awareness* (*awareness softgoal*) es una especialización del concepto de *softgoal* en *i**, la cual representa una necesidad especial de percepción de la presencia y/o acciones de otro usuario, sin la cual, la tarea que el usuario desea realizar se vería negativamente afectada, o incluso no podría realizarse.
- Un *recurso de awareness* (*awareness resource*) es un tipo especial de recurso correspondiente a una implementación o solución de diseño para lograr en *softgoal* de *awareness*.
- Un *enlace de representación* (*playing link*) sirve para representar cuándo un actor asume un rol. Este enlace tiene una condición de guarda con la condición que debe cumplirse para que el actor interprete el rol.
- Un *enlace de participación* (*participation link*) indica quién está involucrado en una tarea. Este enlace tiene un atributo para especificar su cardinalidad, o sea, el número de usuarios que pueden estar involucrados en una tarea.
- Un *enlace de responsabilidad* (*responsability link*) asigna un rol (interpretado por un actor) a un objetivo, *softgoal* o tarea. Este enlace representa quién es el *stakeholder* responsable del cumplimiento de una tarea u objetivo. No es necesario que en *stakeholder* esté involucrado en las sub-tareas del objetivo. No obstante, si el rol es responsable de una tarea u objetivo, este rol es también responsable de los elementos en los que éstos se dividen, a menos que un nuevo enlace de responsabilidad alcance a uno de estos elementos.

Otra diferencia entre CSRML e *i** es que el primero es prácticamente jerárquico (ver de la Figura 27 a la Figura 33), favoreciendo de esta manera la escalabilidad del modelo creado mediante el uso de esta notación. En un primer nivel, tenemos el diagrama de objetivos del sistema que permite definir los objetivos principales que el sistema debería alcanzar. A continuación, aparece el diagrama de responsabilidades, en el cual, el diagrama anterior se descompone en las tareas principales. Además, en este diagrama se definen las responsabilidades de los objetivos y las tareas.

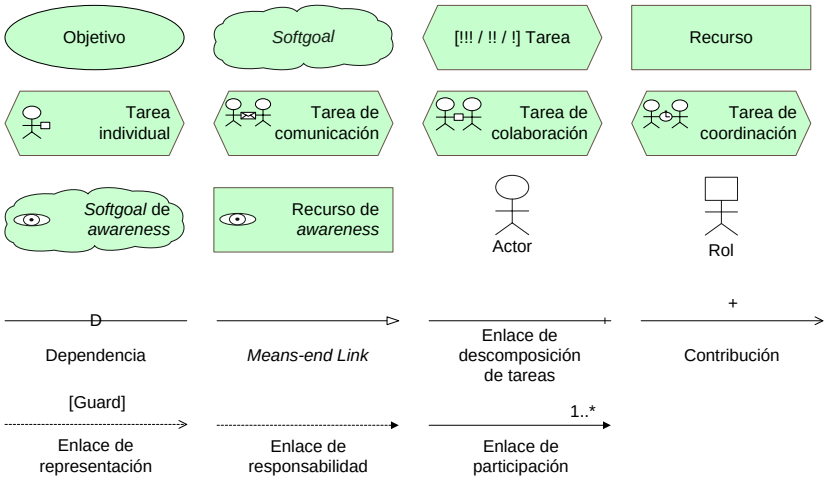

Figura 26: Elementos de CSRML

En un tercer nivel aparecen los diagramas de refinamiento de tareas, en los que las tareas principales del sistema se descomponen en nuevos objetivos, *softgoals*, tareas y recursos. Debido a que CSRML ha sido pensado para su uso en sistemas colaborativos, los límites de los actores/roles de *i** han sido descartados, usándose en su defecto enlaces de participación en estos diagramas. Además, el diagrama de factores de calidad completa la especificación del sistema mostrando los *softgoals* de calidad y los elementos que contribuyen a su cumplimiento.

4.3.2. Caso de estudio: Sistema de gestión de congresos con revisiones colaborativas

Para comprobar cuan adecuado es CSRML para el modelado de requisitos de sistemas colaborativos, en lo siguiente se presenta un caso de estudio clásico: un sistema de gestión de congresos. En este caso, a diferencia de la gestión de congresos original, se dotará al sistema de aspectos de colaboración entre usuarios. Este sistema constará de cuatro objetivos principales:

1. Creación del congreso: El *chair* del congreso dará de alta el congreso en el sistema y pre-registrará a los miembros del comité de programa (CP), los cuales habrán de confirmar ese pre-registro.
2. Envío de artículos: El autor principal de cada artículo se deberá registrar en el congreso y a su vez, pre-registrará a los co-autores que, como en el caso de los miembros de CP, deberán confirmar el pre-registro. Tras ello se podrá enviar (o re-enviar) los artículos.
3. Asignación de artículos a revisores: El *chair* del CP realizará una reunión virtual en la que se irán proponiendo los artículos a los miembros del CP, los cuales elegirán que artículo desean revisar. Esta tarea concluirá cuando cada artículo sea asignado a tres revisores.
4. Revisión de artículos: Cada artículo será revisado colaborativamente por tres revisores mediante una aplicación web. Deberá saberse en cada momento que párrafo está siendo revisado, y por quién, así como los comentarios efectuados sobre cada uno de ellos. Tras la revisión se comunicará el resultado a los autores.

Además de estas tareas, el sistema deberá adecuarse a ciertos factores de calidad del estándar SQuaRE [35], como la precisión o la facilidad de uso.

4.3.3. Modelando el sistema con CSRML

En esta sección se modela el caso de estudio anterior haciendo uso de la notación CSRML con el fin de ilustrar su capacidad expresiva en el modelado de los requisitos de los sistemas colaborativos. Para empezar, en la Figura 27 puede verse el *diagrama de objetivos del sistema*, en el que se definen el objetivo global que el sistema debería alcanzar. Como puede observarse, este objetivo se logrará por medio de la tarea principal: preparación de un congreso usando técnicas de colaboración entre usuarios.

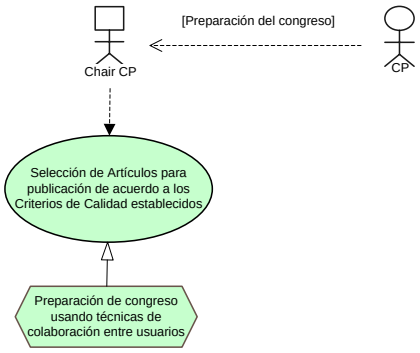

Figura 27: Diagrama de objetivos del sistema

La Figura 28 muestra el *diagrama de responsabilidades* con la tarea principal del sistema y su descomposición en *softgoals* de calidad y sub-tareas. En este diagrama puede observarse quién es el responsable de cada tarea u objetivo mediante el uso de *enlaces de responsabilidad*. Nótese que si un rol es responsable de un objetivo o tarea, también es responsable de los elementos en que se divide, a menos que un nuevo enlace de responsabilidad alcance uno de estos elementos en los que se divide. Por ejemplo, el *chair* es responsable de la preparación del congreso (tarea principal) y de la asignación de artículos (sub-tarea), pero no del envío de los artículos, tarea de la que es responsable el autor principal. Además, puede verse el uso de los *enlaces de representación* usados para indicar la condición que debe cumplirse para que un actor interprete un rol. Para aumentar la legibilidad del modelo, la descomposición de las sub-tareas se realizará en las siguientes figuras.

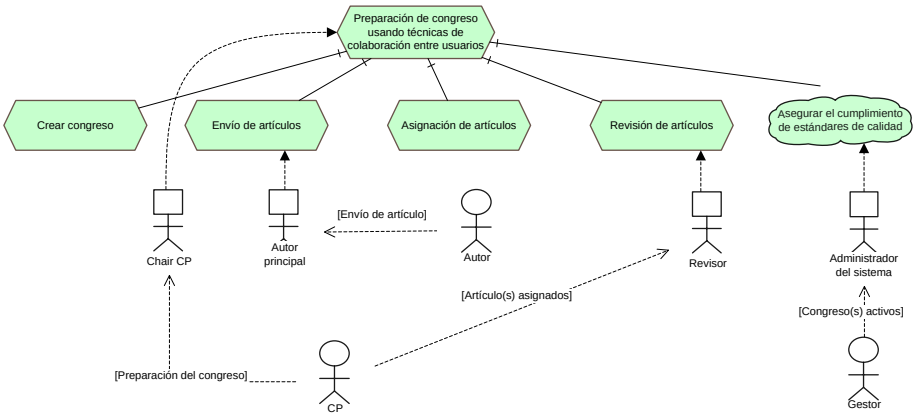

Figura 28: Diagrama de responsabilidades

La Figura 29 representa el *diagrama de refinamiento* de la tarea de creación del congreso. En esta figura, las tareas se refinan en otras más específicas y en nuevos objetivos hasta que se especifiquen tareas individuales o colaborativas (colaboración, coordinación y comunicación).

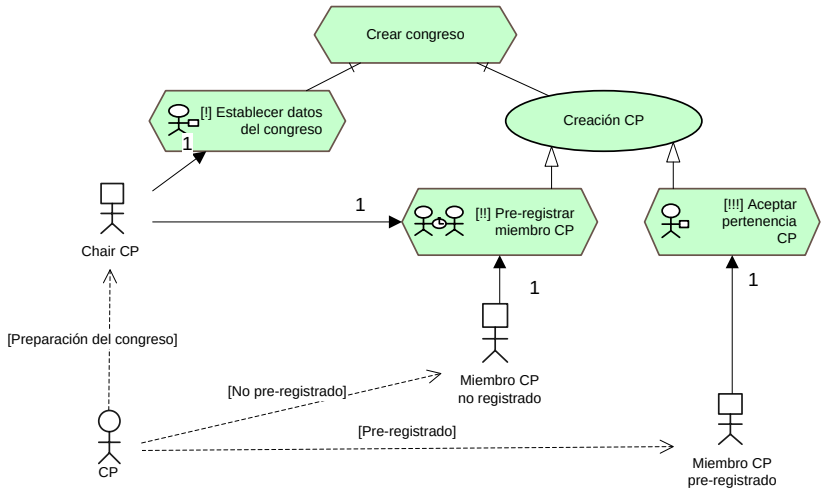

Figura 29: Diagrama de refinamiento de la tarea *Crear congreso*

A continuación, la Figura 30 presenta la descomposición de la tarea de envío de artículos en la que se ilustran los cuatro grados de prioridad que pueden asignarse a las tareas: normal, alta ([!]) , muy alta ([!!]) y la más alta ([!!!]).

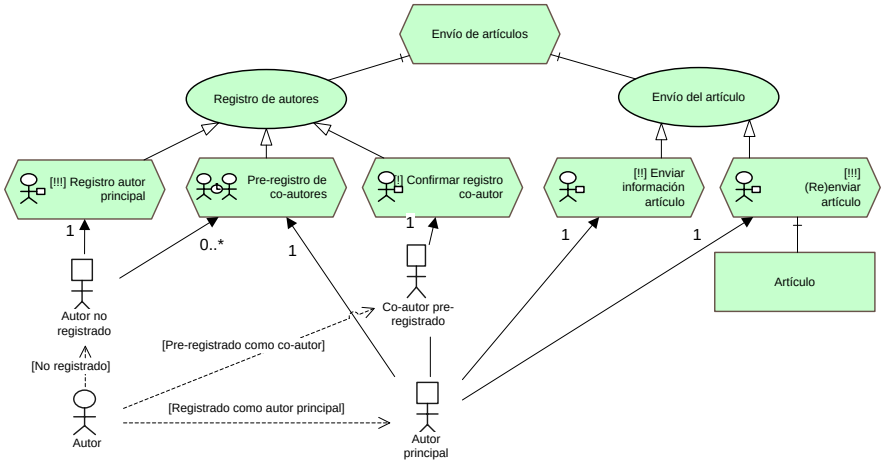

Figura 30: Diagrama de refinamiento de la tarea *Envío de artículos*

La Figura 31 ilustra el refinamiento de la tarea de asignación de artículos. En esta figura se usan diferentes cardinalidades para los *enlaces de participación*. Por ejemplo, para la tarea de proposición de artículos al CP en la que participa el *chair* y tres o más miembros del CP. Esta última cardinalidad se especifica como 3..*. Además, en este diagrama se especifica un *softgoal* de *awareness* llamado *Ser consciente del resto de miembros y de sus artículos asignados*, relacionado con la tarea principal del diagrama. Este softgoal se incluye porque para poder asignar artículos, se debe ser consciente de quién está participando en la reunión, así como de los artículos asignados a cada miembro del CP. Para lograrlo, se introduce el

recurso de awareness llamado *Lista de participantes con estado de las asignaciones*.

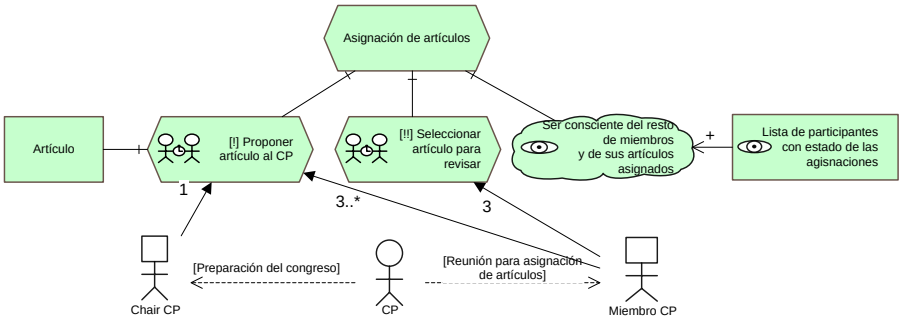

Figura 31: Diagrama de refinamiento de la tarea *Asignación de artículos*

La especificación de las tareas principales del sistema concluye con el refinamiento de la tarea de revisión de artículos (Figura 32). En esta figura se incluyen dos nuevos *softgoal* de *awareness* que harán posible el seguimiento de las revisiones de los demás revisores y el trabajo colaborativo gracias al uso de tele-punteros [25].

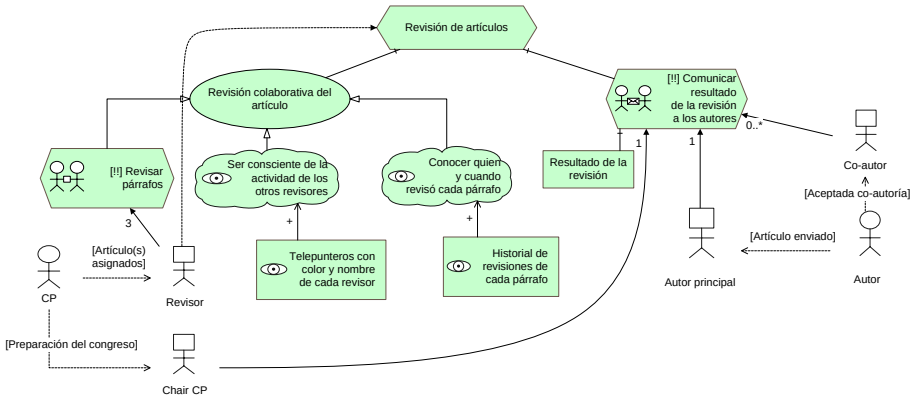

Figura 32: Diagrama de refinamiento de la tarea *Revisión de artículos*

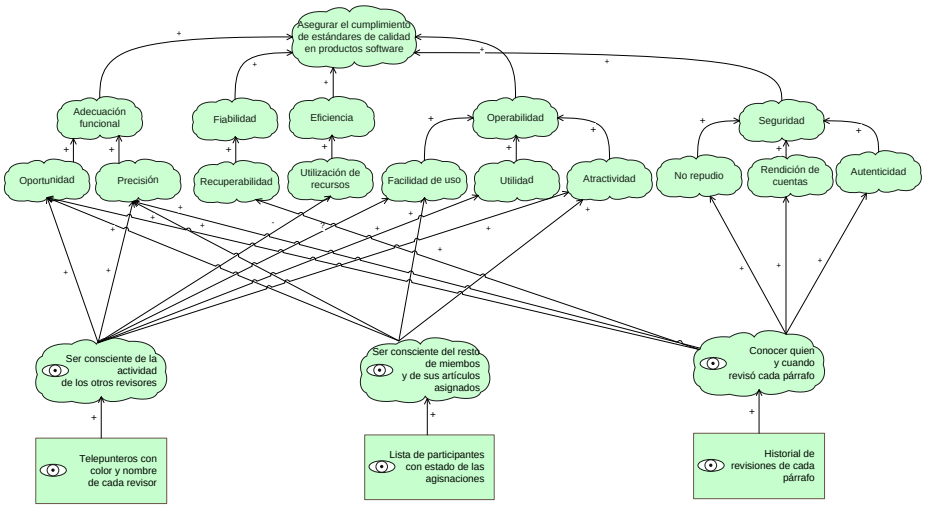

Figura 33: Diagrama de factores de calidad

Finalmente, se describe el *diagrama de factores de calidad* (Figura 33) que permite presentar los factores de calidad del estándar SQuaRE [35] que contribuyen a

lograr el objetivo principal, la Selección de Artículos para publicación de acuerdo a los Criterios de Calidad establecidos. Estos factores son representados como *softgoals* y se relacionan con el principal *softgoal* de calidad mediante *enlaces de contribución* con contribuciones positivas, aunque en algún caso son negativas o indeterminadas. Por ejemplo, el softgoal relacionado con los telepunteros, *Ser consciente de la actividad de los otros revisores*, contribuye positivamente a la utilidad del sistema, pero negativamente a la utilización de recursos, ya que hará posible el conocimiento en tiempo real del trabajo colaborativo del resto de revisores, pero aumentará el uso de la infraestructura de red debido al constante trasiego de información sobre el posicionamiento del cursor de los colaboradores.

4.4. Validando la comprensibilidad de CSRML mediante un experimento controlado

Con el fin de comprobar la comprensibilidad de CSRML para modelar requisitos de sistemas colaborativos se realizó un experimento controlado para comprobar CSRML con *i**, el lenguaje de requisitos en el que se basa, siguiendo las líneas guía definidas en [36]. Es importante resaltar que comprender, de acuerdo con [37], "es un proceso psicológico relacionado con un objeto físico o abstracto, como una persona, situación o mensaje sobre el que una persona puede pensar y usar conceptos para tratar adecuadamente con dicho objeto". Por ello, este experimento se orienta a evaluar la habilidad de los sujetos para pensar y usar conceptos de CSRML / *i** para tratar adecuadamente con los requisitos de un sistema colaborativo.

Podemos definir el objetivo del experimento mediante el uso de la plantilla *Goal Question Metric* [38] de la siguiente manera: **analizar** los lenguajes de especificación de requisitos *i** y CSRML **con el propósito de** evaluar su comprensibilidad **desde el punto de** vista de los investigadores en Ingeniería de Requisitos **en el contexto de** estudiantes de grado. Para conseguir este objetivo definiremos la siguiente hipótesis nula:

- H_0: CSRML posee la misma comprensibilidad que i* en el modelado de requisitos de sistemas colaborativos
- H_1: $\neg H_0$

4.4.1. Diseño Experimental

En este estudio se usó la variable *comprensibilidad (Und)* de los modelos de requisitos CSRML en relación con los de modelos *i*.* De este modo, se evaluó la capacidad para comprender el material experimental correctamente usando el número total de respuestas correctas dividido por el número total de preguntas sobre los modelos presentados, siendo el estudiante la unidad experimental, y, más específicamente, el resultado de su test.

Los sujetos del experimento eran estudiantes de tercero de Informática, los cuales ya habían recibido varias asignaturas relacionadas con la Ingeniería del Software. En concreto, eran alumnos de la asignatura de Ingeniería de Requisitos, por lo que estaban familiarizados con el modelado de requisitos. No obstante, no habían estudiado ninguna aproximación *Goal-Oriented*. El resto de las características del experimento pueden observarse en la Tabla 17.

Tabla 17: Características principales del experimento

Hipótesis nula	H_0: CSRML posee la misma comprensibilidad que i* en el modelado de requisitos de sistemas colaborativos. H_1: $\neg H_0$
Localización	Universidad de Castilla – La Mancha (Albacete)
Fecha	Mayo 2011
Sujetos	30 estudiantes de Informática (15 Grupo 1; 15 Grupo 2)
Variable dependiente	Comprensibilidad de notaciones de modelado de requisitos, medida mediante *Und*
Variable independiente	El dominio al que están relacionados los diagramas y el uso de *i** o CSRML

Este experimento consistió en la comprensión de (parte de) el modelo de requisitos de dos sistemas colaborativos, especificado mediante *i** y CSRML. El primer dominio usado fue un sistema de *e-learning*. En concreto, se usó una parte de una actividad de puzzle, conocida como *Reunión de expertos*, una técnica de aprendizaje cooperativo en la cual los estudiantes investigan individualmente sobre un problema propuesto y tras ellop se explican mutualmente lo que han aprendido compartiendo su punto de vista individual acerca del problema. El segundo dominio se basaba en un sistema de revisiones para congresos con aspectos colaborativos. En particular, se seleccionó la parte del sistema relacionada con el proceso de revisión, realizado de forma colaborativa por tres revisores. Junto a estos modelos se proporcionó un conjunto de preguntas sobre cada dominio que debían ser respondidas por los estudiantes, independientemente del sistema usado para el modelado de los casos de estudio. Esas preguntas trataban sobre la colaboración entre usuarios, enfatizando en las técnicas de *awareness*[3]. Además, se solicitó a los alumnos que indicaran el tiempo invertido en la realización de cada ejercicio, solo para propósitos estadísticos.

Para asignar las especificaciones de los sistemas a los estudiantes, se distribuyó a los sujetos en dos grupos diferentes, G1 y G2, evitando que dos alumnos pertenecientes al mismo grupo estuviesen cerca el uno del otro. Para cada uno de estos sistemas, se usaron dos diagramas de especificación, creados con *i** y CSRML. Se decidió que el grupo G1 realizara el experimento en primer lugar intentando entender el modelo *i** de la actividad de puzzle y después el modelo CSRML del congreso. El grupo G2 hizo lo opuesto con el modelo del puzzle especificado mediante CSRML y después el proceso de revisión con la notación *i**. Además, se elaboró material experimental adicional para proporcionar a los alumnos una descripción más detallada sobre *i** y CSRML, así como una breve descripción sobre *workspace awareness*. El diseño del experimento se resume en la Tabla 18. Este proceso de asignar a los sujetos a 4 tratamientos deferentes, obtenidos mediante la combinación de variables independientes (sistema y lenguaje) corresponde a un diseño factorial 2x2 con interacción confundida [39]. De hecho, dentro de un sistema, el lenguaje cambia junto al grupo de sujetos, por lo que el *efecto aprendizaje* [40] se cancela.

[3] El material usado para la realización de éste experimento puede ser descargado desde: http://www.dsi.uclm.es/trep.php?codtrep=DIAB-11-06-1

Tabla 18: Diseño del experimento

		Sistema	
		Puzzle	Congreso
Lenguaje	*i**	Grupo 1	Grupo 2
	CSRML	Grupo 2	Grupo 1

Además, se decidió que los estudiantes que cumpliesen al menos uno de los siguientes criterios serían excluidos del análisis de resultados:

- La edad del estudiante excede en tres años la edad media de la clase
- El estudiante no está familiarizado con la Ingeniería de Requisitos
- El estudiante posee alguna experiencia previa con *i** u otro enfoque *Goal-Oriented*.

Finalmente, se decidió que si un estudiante abandonaba el experimento, sería entrevistado acerca de la causa de su abandono, y ese hecho sería anotado en los resultados del experimento. Además, cualquier posible interrupción del experimento sería registrada.

4.4.2. Condiciendo el Experimento

El experimento tuvo lugar en el aula de clase de la asignara de Ingeniería de Requisitos. Para evitar la interacción entre los sujetos, éstos fueron debidamente distribuidos en el aula. Además, el instructor era un profesor no relacionado con el desarrollo de CSRML con el fin de evitar el sesgo potencial de su explicación. Así, los sujetos recibieron una sesión introductoria (sobre 60 minutos), en la que el instructor presentó ambas notaciones y sistemas. Estas explicaciones fueron neutrales en relación a la variable independiente (el uso de *i** o CSRML). Antes de dar los modelos a los estudiantes, se obtuvo la siguiente información sobre ellos:

- Conocimientos previos sobre Ingeniería de Requisitos (en años)
- Experiencia previa con aproximaciones *Goal-Oriented*
- Nota media
- Edad
- Sexo

Para incrementar la motivación e interés de los sujetos, el instructor comunicó que su nota final en la asignatura sería incrementada en medio punto por la participación en el experimento. Además, se informó que no se estaba evaluando a los alumnos, sino a las notaciones, y que además, sus resultados serían anónimos y de que no importaba el tiempo usado en el ejercicio, para evitar un posible *efecto techo* [40].

Durante el experimento no abandonó ningún sujeto. Solamente dos estudiantes preguntaron algunas cuestiones relacionadas con la especificación del puzzle con *i**.

4.4.3. Análisis de los resultados

Tras el experimento, los datos fueron recopilados en cuatro tablas de Excel 2010 (una por cada especificación) y fueron analizados para excluir cualquier dato

incompleto o para eliminar a los sujetos que cumpliesen alguna de las características de eliminación del experimento (Sección 4.4.1). No obstante, todos los datos estaban completos y ningún estudiante tuvo que ser descartado.

En la Tabla 19 puede verse que, aparentemente, los grupos de sujetos que recibieron los modelos CSRML obtuvieron un mejor resultado que aquellos que intentaron entender los modelos *i** (valores medios de 0.7933 y 0.8200 contra 0.5733 y 0.5933). A pesar de la clara diferencia entre ambos sistemas, los resultados para la comprensibilidad entre ellos fueron muy similares, lo que significa que el sistema no influyó en los resultados.

Tabla 19: Media y desviación estándar (entre paréntesis) para la comprensibilidad en el experimento

Dominio	**Puzzle**		**Congreso**	
Lenguaje	*i** (*n*=15)	CSRML (*n*=15)	*i** (*n*=15)	CSRML (*n*=15)
Und	0.5733 (0.1163)	0.7933 (0.1335)	0.5933 (0.2154)	0.8200 (0.1512)

Además, se realizó un test de ANOVA (Tabla 20) para rechazar la hipótesis nula H_0, el cual es el test más apropiado para explorar los resultados de un diseño factorial 2x2 con interacción confundida [39,41]. Para un α de 0.05, con df=(1,58), F debe ser al menos 4,0069 para alcanzar un P-valor<0.05, por lo que el resultado para F es estadísticamente significativo. Así, con un P-valor < 0.05, se puede concluir que hay diferencia estadísticamente significativa entre los resultados obtenidos con CSRML e *i**. La diferencia entre ambos resultados puede observarse gráficamente en la Figura 34.

Tabla 20: Resultados del test ANOVA del experimento

Origen de las variaciones	Suma de cuadrados	Grados de libertad	Promedio de los cuadrados	F	Probabilidad	Valor crítico para F
Entre grupos	0,7482	1,0000	0,7482	30,5517	0,0000	4,0069
Dentro de los grupos	1,4203	58,0000	0,0245			
Total	**2,1685**	**59**				

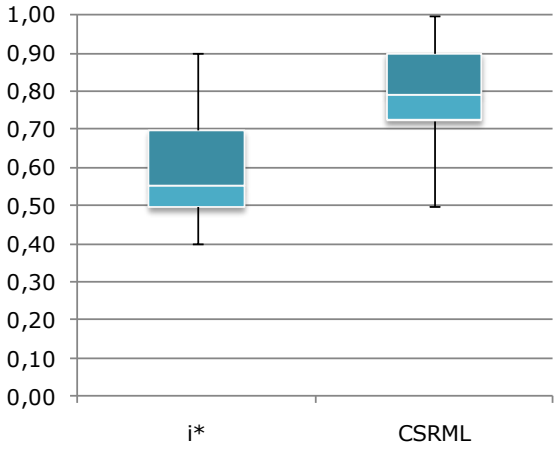

Figura 34: Gráfico de cajas de los resultados obtenidos

Una vez comprobado que CSRML obtiene un mejor resultado, se analizarán los factores que contribuyen a ello analizando los resultados agrupados por preguntas. Para ello, los resultados globales de las preguntas son mostrados en dos gráficos (Figura 35 y Figura 36) que describen los resultados relacionados con cada sistema.

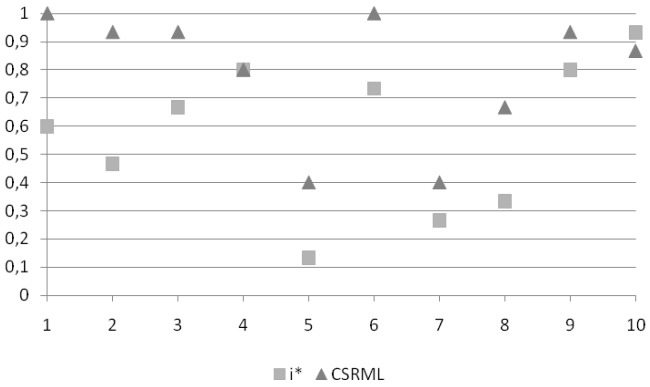

Figura 35: Resultados de las preguntas del Puzzle

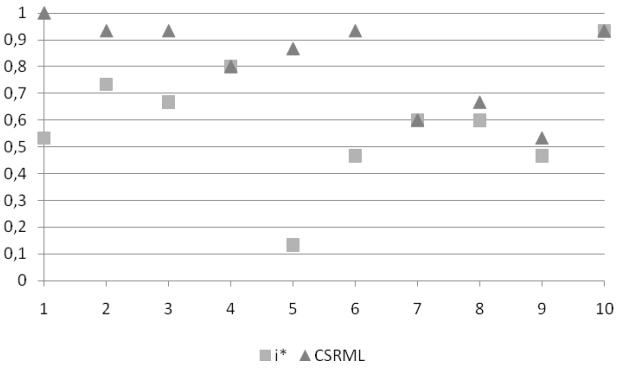

Figura 36: Resultados de las preguntas del Congreso

El primer resultado importante que puede observarse es que CSRML obtiene resultados más altos para las tres primeras preguntas que *i**, especialmente en la primera pregunta, donde CSRML obtiene una relación de acierto del 100%. Esas preguntas estaban relacionadas con tareas de colaboración, por lo que puede decirse que la representación de la colaboración es uno de los puntos fuertes de CSRML, siendo este una piedra angular para la especificación de los sistemas colaborativos.

La pregunta número cuatro obtiene el mismo resultado en ambas notaciones y dominios. Esta pregunta estaba relacionada con una tarea que solo requería de un usuario para su cumplimiento, algo fácil de representar con ambas notaciones.

En las preguntas relacionadas con los roles (5 y 6), CSRML supera a *i** una vez más, debido a sus mecanismos para representar cuando un actor puede asumir un rol, así como a la representación de las cardinalidades de las tareas. Lo mismo ocurre con las preguntas sobre *awareness* (7, 8 y 9) debido a los nuevos elementos de *awareness* introducidos.

No obstante, la última pregunta obtiene un mejor resultado (o el mismo en el caso del congreso) cuando se usa *i**. Debido a ello, en futuros experimentos se podría intentar analizar el impacto de la determinación de la importancia de una tarea usando representaciones alternativas.

4.4.4. Riesgos sobre la Validez del Experimento

En este apartado se explicarán los problemas que pueden poner en riesgo la validez de este experimento, considerando los cuatro tipos de riesgos propuestos en [40].

Validez de la Conclusión

En este experimento, como puede observarse en la sección 4.4.3, el resultado obtenido por el ANOVA es estadísticamente significativo, lo que nos permite rechazar la hipótesis nula inicial con un grado de certeza amplio.

Validez Interna

El número de sujetos que participaron en el experimento fue lo suficientemente grande (=30), según el teorema central del límite [42], para que las función de distribución de los datos se aproxime a una distribución normal, algo necesario apra obtener un conjunto de datos estadísticamente significativo.

Validez de Construcción

En el análisis se usó una medida basada en las respuestas correctas divididas entre el número total de respuestas para cuantificar la comprensibilidad. No obstante, en futuros trabajos, podría considerarse el uso de líneas guías formales como las presentadas en CTML [43].

Validez Externa

Los modelos presentados en este experimento fueron relativamente fáciles de comprender por los estudiantes. Se considerará incrementar la complejidad de los modelos en trabajos futuros en los que usemos como sujetos experimentales a profesionales de la Ingeniería del Software. En estos experimentos podrían usarse dominios más enfocados hacia la industria.

4.5. Conclusiones

Los sistemas colaborativos tienen una gran demanda en término de NFRs relacionados con el *awareness* y la calidad. Por lo cual, la selección de una técnica de Ingeniería de Requisitos que proporcione un soporte adecuado para la especificación de estas características es de gran importancia. Así, se ha comprobado que para esta especificación, las técnicas más adecuadas son los enfoques *Goal-Oriented*. No obstante, se puede concluir tras el análisis realizado que los enfoques GO no son totalmente apropiados para modelar las características de los sistemas colaborativos y sus relaciones con los requisitos de *awareness* y calidad. De las técnicas analizadas, el enfoque *i** es el único que obtiene un resultado positivo para las características analizadas, relacionadas con los sistemas colaborativos. Además, *i** es el único enfoque que proporciona un soporte (parcial) para cuantificar relaciones entre requisitos cuando se usan enlaces de contribución. No obstante, esta técnica tiene algunos problemas, como la falta de representación jerárquica o el soporte para especificar la importancia de los requisitos. Pero quizá, el problema más importante es la comprensibilidad de los requisitos de *awareness*

no es apropiada. Por ejemplo, *i** no proporciona soporte para especificar cuando una tarea es llevada a cabo por varios roles, algo muy común en los sistemas colaborativos.

Estas conclusiones apoyan la hipótesis inicial de que las técnicas actuales de Ingeniería de Requisitos deben ser enriquecidas para solventar los problemas identificados durante el presente estudio.

Para solucionar estos problemas, se propone CSRML como una extensión de *i** para modelar los requisitos de los sistemas colaborativos. De este modo, CSRML solventa la representación de la colaboración entre usuario con el uso de los enlaces de participación y la distinción de varios tipos de tareas colaborativas. A su vez, La representación del *awareness* es posible con esta notación gracias a los nuevos elementos de *awareness* incorporados. Además, la complejidad y falta de legibilidad de los diagramas *i** queda reducida mediante la división jerárquica del modelo en varios diagramas en los que los modelos tienden a descomponerse en forma da árbol, permitiendo una mayor legibilidad y extensibilidad.

Para validar cuan adecuado es este lenguaje se ha modelado un sistema colaborativo basado en un sistema de gestión de congresos con revisiones colaborativas. Este caso de estudio fue modelado debido a que poseía un conjunto de características que serían difíciles o incluso imposibles de representar usando la notación *i** original, como la asignación y la revisión colaborativa de artículos. Estas características fueron adecuadamente descritas en los modelos añadiendo un conjunto de nuevos elementos y relaciones a la notación *i**. La representación de la calidad y el *awareness* fue posible por medio de nuevos elementos de awareness y por la inclusión de un conjunto de nuevos diagramas que dotan de cierta estructura a la especificación.

En resumen, CSRML ayuda a mejorar la comprensibilidad y mantenibilidad de los modelos de requisitos de sistemas colaborativos añadiendo nuevos elementos y relaciones a *i**. Estos nuevos elementos facilitan la especificación de los requisitos de *awareness*, los cuales son un factor clave en el desarrollo de los sistemas colaborativos.

Adicionalmente, se llevó a cabo un experimento para comprobar la comprensibilidad de la notación CSRML en relación con la notación *i** original en la que éste se basa. Los participantes en este experimento fueron treinta estudiantes de Informática que intentaron dos modelos pertenecientes a dos sistemas colaborativos distintos (un sistema de *e-learning* y un sistema de revisión para congresos con soporte a la colaboración).

Como resultado de ese estudio se observó que la notación CSRML mejora la comprensibilidad de los modelos de requisitos respecto a *i**. También se encontraron algunos hechos importantes estudiando las respuestas de los alumnos. Por ejemplo, se comprobó que un modelo CSRML es más comprensible debido a sus características de representación de la colaboración, a los nuevos elementos de *awareness* introducidos y a las técnicas de majeo de roles mejoradas sobre el *i** original, hechos que pueden observarse viendo las respuestas de todas las preguntas (a excepción de la décima). No obstante, también se encontró un

problema importante cuando se establece la importancia de una tarea que deberá ser solventado en futuras versiones del lenguaje.

Para complementar estos resultados, se realizaron dos réplicas de este experimento, una en la Universidad Politécnica de Valencia y otra en la Universidad de La Plata (Argentina). Los resultados de ambos experimentos están siendo analizados actualmente.

5. Líneas futuras de investigación

5.1. Introducción

Un sistema colaborativo es aquel que, basado en computador, soporta a grupos de personas involucradas en una tarea común (u objetivo) y que provee un producto software de calidad en un ambiente compartido. En ese mismo grupo de sistemas, las aplicaciones groupware son sistemas que permiten a un grupo de usuarios comunicarse y trabajar de forma colaborativa. Esto implica que el grupo de usuarios puede coordinar actividades, solucionar problemas, editar documentos o diagramas y negociar, usando tecnologías específicas.

Uno de los principales retos que presenta el desarrollo de dichos sistemas colaborativos y aplicaciones groupware, especialmente cuando se considera a sus usuarios finales, es la parte destinada a la interfaz de usuario. El grupo LoUISE (Laboratory of User Interfaces & Software Engineering) viene desarrollando actividades de investigación relacionadas con el desarrollo de interfaces de usuario, tanto tradicionales como groupware durante los últimos 12 años. En este sentido, la presente propuesta de tesis doctoral pretende seguir haciendo contribuciones en esa misma línea de investigación y para ello se marca el objetivo de mejorar la interacción colaborativa desde una perspectiva de calidad.

5.2. Dominio del problema

Con la popularidad ganada por Internet en los últimos años, muchos escenarios de trabajo han cambiado y muchos de esos mismos escenarios se basan en la colaboración a través de la Web. En la actualidad, millones de usuarios comparten información y esfuerzos, colaborando en múltiples dominios de aplicación y haciéndolo a distancia tanto de manera síncrona como asíncrona. En este sentido, los resultados esperados de esta propuesta de tesis doctoral tienen múltiples dominios de aplicación: administrativa, comercial, educativa, médica, etc.

En definitiva, la nueva realidad que ofrece en el ámbito social las nuevas tecnologías sugiere la necesidad de herramientas, ambientes y metodologías que permitan a los usuarios colaborar para cubrir sus necesidades y objetivos, es decir, supone superar las limitaciones que presenten las actuales herramientas groupware y el desarrollo de estas.

5.3. Propuesta de tesis

La mejora de la interacción colaborativa desde una perspectiva de calidad y de los usuarios pasa por sentar sus bases en mejorar la calidad en uso de un producto software [27], es decir, mejorar la calidad percibida por dichos usuarios en tanto en cuanto éstos utilizan productos software en sus actividades cotidianas [44,45]. Dicha calidad, la calidad en uso, depende de la calidad del producto y ésta, a su vez, de la calidad de proceso (véase Figura 37) [27].

Figura 37: Sistemas Colaborativos desde una perspectiva de calidad

Fruto de la revisión de la literatura realizada hasta el momento y de mucha de la actividad investigadora llevada a cabo en el grupo LoUISE, identificamos la necesidad de tratar conjuntamente desarrollo y calidad de herramientas groupware y, por ello, partimos de las siguientes hipótesis en esta propuesta de investigación:

1. Primero, la calidad en uso, además de otras características comúnmente tratadas, como sería la usabilidad, precisa contemplar el logro de facilidades para mejorar la adaptación para ser utilizada en diferentes contextos. Al menos así lo sugieren las versiones más recientes de los estándares internacionales relacionados con calidad, tales como la ISO/IEC 9126-1:2001 o el ISO/IEC 25010, éste último actualmente en desarrollo. El grupo LoUISE dispone de propuestas metodológicas, fruto del trabajo de investigación realizado hasta el momento, para dotar de facilidades de adaptación a interfaces de usuario.
2. Segunda, la calidad de un producto software es el efecto combinado de diferentes factores y atributos de calidad. En esta propuesta de tesis doctoral identificamos que dichos factores determinan o influyen decisivamente en el awareness [7,46,47] o conciencia que el usuario tiene del producto software. Dicho concepto lo asociamos, en primera instancia, a características del propio producto software y en función de ello estará influenciado por los atributos tanto internos como externos del mismo. En esta tesis se analizarán las contribuciones que los factores y criterios de calidad tienen sobre el citado concepto de awareness.
3. Tercero y último, y no por ello menos importante, el origen de la calidad de un producto software o de sus efectos se encuentra en la propia calidad del proceso que conduce al desarrollo de ese producto software. Existen diferentes propuestas para el desarrollo de productos groupware, por ejemplo AMENITIES [48], CIAM [49] y TOUCHE [50], pero en dichas propuestas hemos observado que se presta especial atención a las actividades de diseño y desarrollo basado en modelos y no tanta a las primeras fases de ese mismo proceso de desarrollo. La última aportación que identificamos en esta propuesta de tesis doctoral pasa por identificar y refinar técnicas de especificación en el trabajo en grupo asistido por computador (CSCW) que conduzcan a un desarrollo más preciso y con la intención de contribuir, en última instancia, a la automatización del proceso de desarrollo.

Seguidamente, profundizaremos en cada una de las anteriores hipótesis.

5.3.1. Calidad de proceso

Con el objetivo de contribuir a la calidad del proceso del desarrollo de un sistema colaborativo es necesaria la existencia de un proceso formalizado que guíe a los actores involucrados. Como se ha descrito anteriormente, existen una serie propuestas en este sentido, tales como AMENITIES [48], CIAM [49] o TOUCHE [50], donde cada una de las cuales presenta sus fortalezas y debilidades, todas ella especialmente centradas en la parte del proceso dedicada al diseño.

Sin embargo, una de las disciplinas más importantes en el proceso de desarrollo de cualquier producto software es la *Ingeniería de Requisitos* (IR), dado que ésta permite a todos los involucrados (stakeholders) conocer el alcance del proyecto, las necesidades de los usuarios, la identificación de factores de calidad clave, etc. Es por ello que su correcta aplicación es básica a fin de desarrollar aquel producto que demanda el cliente. Actualmente, en el ámbito de los sistemas groupware sólo una de las metodologías existentes, TOUCHE, presta atención a esta disciplina mediante la descripción de plantillas de Casos de Uso que han sido descritas extendiendo aquellas presentadas por Amador Durán [51]. Sin embargo, las extensiones planteadas están especialmente relacionadas con el modelo de usuario, especialmente, con la identificación de la interacción colaborativa en la realización de determinados servicios, olvidando otros requisitos no funcionales, tales como el rendimiento, la adaptación o el awareness, importantes para este tipo de sistemas. El mayor problema viene debido a las carencias expresivas que los Casos de Uso plantean para especificar los requisitos no funcionales. Sin embargo, existen otras técnicas de especificación, como las Orientadas a Objetivos (Goal-Oriented Requirements Engineering, [52]), que sí ofrecen un apropiado soporte para la especificación de éste tipo de requisitos.

Así, una de las contribuciones de esta tesis será estudiar qué técnicas de especificación de requisitos son más apropiadas para la especificación de sistemas groupware y adaptarlas para contemplar características relacionadas con el modelo de interacción necesario para este tipo de sistemas. Además, dado que tanto la calidad en uso como la calidad de producto son objetivos en sí mismo de esta propuesta de tesis, se plantea además que la técnica de especificación a desarrollar en el contexto de esta tesis se nutra en su descripción de estándares de calidad internacionales como la ISO/IEC 9126-1 y la norma ISO/IEC 25010.

Además, a fin de ofrecer soporte a dichas técnicas de especificación se plantea en esta tesis doctoral la incorporación de la aproximación del Desarrollo Dirigido por Modelos (MDD). La explotación de modelos en el desarrollo software no es una aproximación nueva sino que se ha venido utilizando desde principios de los ochenta con la aparición de la programación automática [53] o los compiladores de modelos [54].Sin embargo, ha sido durante los últimos años cuando un mayor esfuerzo se ha llevado a cabo en la explotación de modelos para el desarrollo software. Claros ejemplos han sido la aparición de términos como Desarrollo Dirigido por Modelos (DDM) [55], Ingeniería Dirigida por Modelos (IDM) [56], o Arquitectura Dirigida por Modelos (Model-Driven Architecture, MDA) [57]. Los tres términos han sido utilizados como sinónimos para describir el proceso de desarrollo como un problema de modelado. Una de dichas propuestas, concretamente la propuesta MDA, plantea el proceso de desarrollo (ver Figura 38) como la aplicación

sucesiva de un conjunto de transformaciones que permiten llegar desde un modelo independiente de la computación (CIM) hasta el desarrollo de un modelo específico de la implementación (ISM) que describe el código de la aplicación a desarrollar. La utilización de esta propuesta aporta como una de sus principales aportaciones el desarrollo de productos de calidad gracias a que la automatización permite evitar errores derivados de la programación tradicional.

Tal y como se puede apreciar en la Figura 38, uno de los principales modelos que aparecen en MDA es el CIM, dado que en él se describen los requisitos del sistema a desarrollar, los procesos, las reglas de negocio, etc. Dado que la mayoría de las metodologías de sistemas colaborativos siguen esta aproximación para el desarrollo, en el contexto de esta tesis doctoral se describirán a este nivel el modelo de requisitos identificado previamente, identificando las relaciones necesarias que permitan su trazabilidad al modelo PIM descrito en otras propuestas. Además, se desarrollarán aquellas transformaciones que sean necesarias a fin de facilitar la generación inicial de fragmentos de dicho modelo, automatizando así, tanto como sea posible el proceso de desarrollo y mejorando así la calidad de los productos a desarrollar.

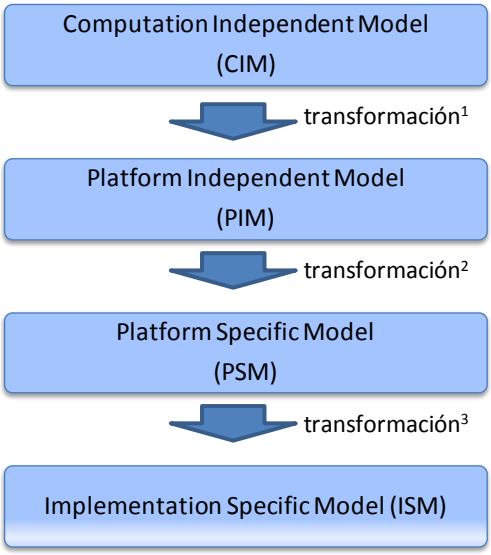

Figura 38: Modelos en MDA

5.3.2. Calidad del producto

Siendo el objetivo principal de esta propuesta de tesis doctoral contribuir al logro de la calidad en uso de un sistema groupware y habiendo propuesto considerar, en primera instancia, la calidad de proceso, los estándares internacionales identifican contribuciones de influencia y dependencia entre los distintos tipos de calidad mencionados utilizando el concepto de calidad del producto como nexo conector (véase la Figura 37). Sin embargo, dichas influencias y dependencias no están caracterizadas ni documentadas adecuadamente [58] de forma que permitan su explotación y traza en el proceso de desarrollo.

En esta tesis pretendemos contribuir a dar solución al escenario identificado mediante la propuesta y establecimiento de relaciones con una semántica apropiada entre la especificación de requisitos y los factores de calidad de un

producto software. Además, también se identificarán las relaciones que pueden establecerse entre esos factores y criterios de calidad, a nivel de producto, y cómo los percibe y utiliza el usuario/grupo de usuarios en el ámbito de la calidad en uso de los sistemas groupware.

5.3.3. Calidad en uso

De acuerdo a los últimos estándares relacionados con la calidad en uso [27] de un producto software, las facilidades de adaptación del mismo han cobrado especial relevancia. Dichas facilidades vienen marcadas desde dos puntos de vista: la adaptación al entorno en el cuál se produce la interacción, y especialmente destacable es la adaptación del producto software a los usuarios que lo emplean.

En el caso de las aplicaciones groupware dicho factor de adaptación cobra si cabe una mayor relevancia, ya que la misma aplicación es usada de forma síncrona o asíncrona, por distintos grupos de usuarios, cuyas características (cognitivas, preferencias, experiencia en el uso de ordenadores, físicas, etc.) puede ser variables, lo cual hace que el diseño de una única interfaz de usuario para un perfil general de usuario no sea suficiente, si no que aparece la necesidad de ser capaces de diseñar una interfaz de usuario capaz de acomodarse al máximo número de los usuarios que con ella interactúan.

Desafortunadamente, el diseño de las capacidades de adaptación ha sido acometido mediante técnicas totalmente ad-hoc, que distan mucho de las actuales tendencias más ingenieriles del diseño dirigido por modelos. En este sentido, dentro del grupo LoUISE, y en colaboración con grupo PRISME de la Université Catholique de Louvaine en Bélgica, se ha desarrollado un framework [59] que precisamente persigue la sistematización del diseño de las capacidades de adaptación de la interfaz de usuario de los productos software. Sin embargo, los trabajos realizados no han estado centrados en los sistemas CSCW, los cuáles, por su naturaleza colaborativa, presentan peculiaridades que suponen una ampliación y un refinamiento de los trabajos realizados para el diseño de sistema no colaborativos.

En esta tesis se realizará la identificación y descripción de técnicas ingenieriles para lograr que un producto software presente capacidades de adaptación que permitan amoldarlo a las preferencias y habilidades de los usuarios y, en general, a las características del contexto de uso de un sistema groupware. Con dicha aportación se estará dando un paso importante en la mejora de la calidad en uso de los sistemas groupware.

El último concepto a considerar en la presente tesis doctoral, el concepto de awareness, aparece de manera natural en los sistemas groupware de acuerdo a su definición [60]. El awareness, como conocimiento actualizado del entorno, es un aspecto vital para la interacción y la colaboración. Si un sistema no proporciona awareness, no hay una interacción efectiva y sin ella no puede haber colaboración [61]. Aunque dicho concepto no está incluido dentro de los estándares de calidad, sí que desempeña un papel determinante en la percepción de la calidad en uso que del sistema groupware tienen sus usuarios.

En este sentido, en esta tesis doctoral se trabajará el concepto de awareness integrándolo en la definición de calidad en uso e identificando influencias y dependencias entre los factores de calidad de producto y en uso.

5.4. Planificación

Para la realización de esta tesis se plantea la aplicación de la metodología Investigación-Acción ([62], IA). Para su correcta aplicación se han identificado los cuatros tipos de roles que intervienen y que se describen a continuación (Figura 39):

- El *equipo investigador* que lleva a cabo el proceso de investigación. Este rol va a ser llevado a cabo tanto por el doctorando como por el grupo de investigación LoUISE en el que se integra (UCLM).
- El *objeto* bajo investigación, es decir, el problema que debe ser resuelto. En este caso, el objeto de investigación es el desarrollo de sistemas colaborativos desde una perspectiva de calidad.
- El *grupo crítico* de referencia, el cual recibe los resultados de la investigación y participan en el proceso de investigación (aunque menos activamente que el investigador). Este rol va a ser llevado a cabo por la empresa SymbiaIT a fin de validar los resultados aplicables de la investigación.
- El *beneficiario* de la investigación es quien espera explotar los resultados de la investigación aunque no tome parte en el proceso. En este caso, este papel es llevado a cabo por cualquier colectivo involucrado en el desarrollo de sistemas groupware.

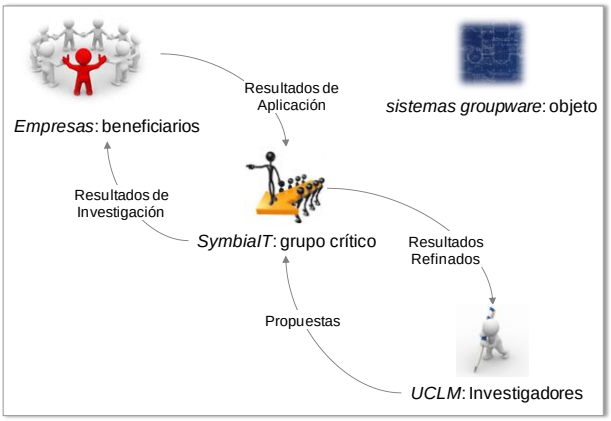

Figura 39: Actores en el proyecto de tesis

En el marco de esta tesis doctoral los diferentes actores detectados participaran en el desarrollo de las siguientes actividades, cuya distribución temporal se describe en la Figura 40:

- *Estado del arte*. se identificarán las cuestiones de investigación relativas a modelos de calidad software disponibles, técnicas de elicitación de requisitos, metodologías de desarrollo de sistemas colaborativos, definición de awareness, etc. Será está una actividad que se realice de forma reiterada a lo largo de toda la tesis doctoral.

- *Ingeniería de requisitos en sistemas groupware*. Elaboración de una propuesta integradora de técnicas de Ingeniería de Requisitos para la especificación de sistemas colaborativos y soportado por técnicas MDD. Además, se considerarán las modificaciones necesarias en los procesos descritos por las metodologías existentes para sistemas groupware.
- *Identificación de influencias y dependencias del proceso a la calidad del producto*. En los estándares internacionales relacionados con calidad, especialmente en los referidos en esta propuesta, se identifican influencias y dependencias de los conceptos de calidad, pero estas relaciones no están ni documentadas ni descritas con suficiente nivel de detalle. En esta tesis doctoral realizaremos aportaciones en este ámbito.
- *Identificación de influencias y dependencias del producto a la calidad en uso*. De manera similar a la actividad anterior, y para seguir considerando aspectos de trazabilidad, la calidad del producto tiene influencia en la calidad de uso del mismo. Este aspecto será considerado en esta tesis doctoral con el objetivo de constatar que los requisitos identificados en la primera de las actividades identificadas tienen correspondencia con los productos finalmente ofrecidos a los usuarios.
- *Soporte*. A lo largo de la presente tesis doctoral, se realizarán a su vez diferentes herramientas que ayuden a poner en práctica cada una de las aportaciones antes mencionadas.
- *Validación*. A fin de determinar la validez de las diferentes propuestas planteadas, se llevarán a cabo diferentes prototipos sobre diferentes dominios de aplicación como por ejemplo los sistemas de e-learning o sistemas de rehabilitación, contextos ambos en los que el grupo LoUISE tiene experiencia previa [63,64].
- *Difusión*. A lo largo de todo el proceso de realización de la presente tesis doctoral, se realizarán diferentes publicaciones que permitan obtener el feedback necesario tanto de foros nacionales como internacionales. Además, también como parte de dicho proceso de difusión se realizará la escritura del documento de tesis.

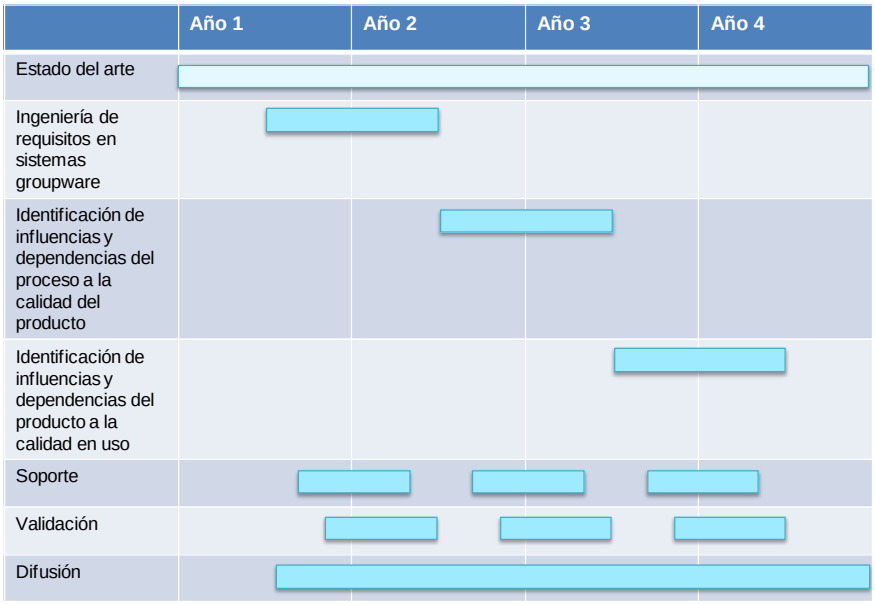

Figura 40: Distribución temporal de las actividades

Estas actividades se llevarán a cabo de forma iterativa, siguiendo el ciclo que recomiendan Padak & Padak [62] de *planificar – actuar – observar - reflexionar*, permitiendo así dar soluciones más refinadas a lo largo de la realización de la presente tesis doctoral.

Bibliografía

[1] M.A. Teruel, “Generación automática de instrumentos FM a partir de ejemplos,” 2010. Proyecto Final de Carrera. *Universidad de Castilla – La Mancha*

[2] C. Gutwin, “Workspace Awareness in Real-Time Distributed Groupware,” Calgary, Alberta, 1997.

[3] N. Fenton, M. Neil, and D. Marquez, “Using Bayesian networks to predict software defects and reliability,” *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, vol. 222, Dec. 2008, pp. 701-712.

[4] C. Van Koten and A. Gray, “An application of Bayesian network for predicting object-oriented software maintainability,” *Information and Software Technology*, vol. 48, 2006, p. 59–67.

[5] M.A. Teruel, E. Navarro, V. López-Jaquero, F. Montero, and P. González, “An Empirical Evaluation of Requirement Engineering Techniques for Collaborative Systems,” *15th International Conference on Evaluation and Assessment in Software Engineering*, Durham, UK: 2011.

[6] J.L. Garrido, “AMENITIES: una Metodología para el Desarrollo de Sistemas Cooperativos Basada en Modelos de Comportamiento y Tareas,” University of Granada, 2003.

[7] C. Gutwin and S. Greenberg, “A Descriptive Framework of Workspace Awareness for Real-Time Groupware,” *Computer Supported Cooperative Work*, vol. 11, 2002, pp. 411-446.

[8] H. Hochmuller, “Towards the Proper Integration of Extra-Functional Requirements,” *Australasian Journal of Information Systems*, vol. 6, 1999, pp. 98-117.

[9] A. Cockburn, *Writting Effective Use Cases*, Addison-Wesley, 2000.

[10] A. Finkelsetin, J. Kramer, B. Nuseibeh, L. Finkelstein, and M. Goedicke, “Viewpoints: A Framework for Integrating Multiple Perspectives in System Development,” *International Journal of Software Engineering and Knowledge Engineering*, vol. 2, 1992, pp. 31-57.

[11] L.M. Cysneiros and E. Yu, *Non-Functional Requirements Elicitation (Perspectives on Software Requirements)*, Springer, 2003.

[12] Google, “Google Docs,” 2001.

[13] G. Booch, J. Rumbaugh, and I. Jacobson, *The Unified Modeling Language User Guide*, Addison Wesley, 2005.

[14] B. Nuseibeh, J. Kramer, and A. Finkelstein, “A Framework for Expressing the Relationships Between Multiple Views in Requirements Specification,” *IEEE Transactions on Software Engineering*, vol. 20, 1994, pp. 760-773.

[15] A. Finkelsetin, J. Kramer, B. Nuseibeh, L. Finkelstein, and M. Goedicke, “Viewpoints: A Framework for Integrating Multiple Perspectives in System Development,” *International Journal of Software Engineering and Knowledge Engineering*, vol. 2, 1992, pp. 31-57.

[16] B. Nuseibeh, J. Kramer, and A. Finkelstein, “A Framework for Expressing the Relationships Between Multiple Views in Requirements Specification,” *IEEE Transactions on Software Engineering*, vol. 20, 1994, pp. 760-773.

[17] A. van Lamsweerde, “Goal-Oriented Requirements Engineering: A Guided Tour,” *Proceedings of the Fifth IEEE International Symposium on Requirements Engineering*, IEEE Computer Society, 2001, pp. 249-263.

[18] E. Kavakli and P. Loucopoulos, “Goal Modeling in Requirements Engineering: Analysis and Critique of Current Methods,” *Information Modeling Methods and Methodologies*, 2004, pp. 102-124.

[19] P. Giorgini, J. Mylopoulos, E. Nicchiarelli, and R. Sebastiani, “Formal Reasoning Techniques for Goal Models,” *Journal on Data Semantics*, vol. 2800/2003, 2004, pp. 1-20.

[20] L.M. Cysneiros and E. Yu, *Non-Functional Requirements Elicitation (Perspectives on Software Requirements)*, Springer, 2003.

[21] J.C. Sampaio do Prado Leite and A.P.M. Franco, “A Strategy for Conceptual Model Acquisition,” *First International Symposium on Requirements Engineering*, 1993, pp. 243-246.

[22] K. Pohl, *Requirements Engeneering: Fundamentals, Principles, and Techniques*, Springer, 2010.

[23] J. Castro, M. Kolp, and J. Mylopoulos, “A requirements-driven development methodology,” *In Proc. of the 13th Int. Conf. On Advanced Information Systems Engineering (CAiSE’01)*, 2001, pp. 108-123.

[24] A. Cockburn, *Writting Effective Use Cases*, Addison-Wesley, 2000.

[25] C. Gutwin, S. Greenberg, and M. Roseman, “Workspace Awareness in Real-Time Distributed Groupware: Framework, Widgets, and Evaluation,” *Proceedings of HCI on People and Computers XI*, Springer-Verlag, 1996, pp. 281-298.

[26] T. Schümmer and S. Lukosch, *Patterns for Computer-Mediated Interaction*, John Wiley & Sons Ltd, 2007.

[27] ISO/IEC JTC1/SC7, “Software engineering - Software product Quality Requirements and Evaluation (SQuaRE) Quality model,” 2008.

[28] A. Moreira, A. Rashid, and J. Araújo, “Multi-Dimensional Separation of Concerns in Requirements Engineering,” *13th IEEE International Conference on Requirements Engineering (RE’05)*, IEEE Computer Society, 2005, pp. 285-296.

[29] X. Franch, “On the Lightweight Use of Goal-Oriented Models for Software Package Selection,” *CAiSE*, 2005, pp. 551-566.

[30] A. Moreira, J. Araújo, and I. Brito, “Crosscutting Quality Attributes for Requirements Engineering,” *Proceedings of the 14th international conference on Software engineering and knowledge engineering*, ACM, 2002, pp. 167-174.

[31] B. Kitchenham, “A methodology for evaluating software engineering methods and tools,” *Experimental Software Engineering Issues: Critical Assessment and Future Directions*, H. Rombach, V. Basili, and R. Selby, eds., Springer Berlin / Heidelberg, 1993, pp. 121-124.

[32] P. Kruchten, *The Rational Unified Process: An Introduction (2nd Edition)*, Addison-Wesley Professional, 2000.

[33] J. Castro, M. Kolp, and J. Mylopoulos, “A requirements-driven development methodology,” *In Proc. of the 13th Int. Conf. On Advanced Information Systems Engineering (CAiSE’01)*, 2001, pp. 108-123.

[34] M. Noguera, M. González, J.L. Garrido, V. Hurtado, and M. Rodríguez, “System Modeling for Systematic Development of Groupware Applications,” *International Conference on Software Engineering Research and Practice*, 2006, pp. 750-756.

[35] ISO/IEC JTC1/SC7, “Software engineering - Software product Quality Requirements and Evaluation (SQuaRE) Quality model,” 2008.

[36] B. a Kitchenham, S.L. Pfleeger, L.M. Pickard, P.W. Jones, D.C. Hoaglin, K. El Emam, and J. Rosenberg, "Preliminary guidelines for empirical research in software engineering," *IEEE Transactions on Software Engineering*, vol. 28, Aug. 2002, pp. 721-734.

[37] C. Bereiter, *Education and Mind in the Knowledge Age*, Routledge, 2002.

[38] V.R. Basili, G. Caldiera, and H.D. Rombach, "The Goal Question Metric Approach," *Encyclopedia of Software Engineering*, vol. 2, 1994, p. 528–532.

[39] B.J. Winer, D.R. Brown, and K.M. Michels, *Statistical Principles in Experimental Design*, McGraw-Hill, 1991.

[40] C. Wohlin, P. Runeson, M. Höst, M.C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering: An Introduction*, 2000.

[41] R.E. Kirk, *Experimental Design: Procedures for Behavioral Sciences*, Pacific Grove, CA: Brooks/Cole, 1995.

[42] C.M. Grinstead and L.S. Snell, *Introduction to Probability*, 2006.

[43] R.E. Mayer, *Multimedia Learning*, Cambridge University Press, 2001.

[44] F. Montero, V. López-Jaquero, M.D. Lozano, and P. González, "Usability and Web Site Evaluation: Quality Models and User Testing Evaluations," *ICEIS*, 2003, pp. 525-528.

[45] F. Montero, M.D. Lozano, and P. González, "Usability-Oriented Quality Model Based on Ergonomic Criteria," *Handbook of Research on Web Information Systems Quality*, 2008, pp. 220-233.

[46] C. Gutwin and S. Greenberg, "The effects of workspace awareness support on the usability of real-time distributed groupware," *ACM Transactions on Computer-Human Interaction*, vol. 6, 1999, pp. 243-281.

[47] C. Gutwin and S. Greenberg, "Effects of awareness support on groupware usability," *Interface*, 1998, pp. 511-518.

[48] J.L. Garrido, P. Paderewski, M.L. Rodríguez-Almendros, M.J. Hornos, and M. Noguera, "A Software Architecture Intended to Design High Quality Groupware Applications," *Software Engineering Research and Practice*, 2005, pp. 59-65.

[49] A.I. Molina, M.A. Redondo, M. Ortega, and U. Hoppe, "CIAM: A methodology for the development of groupware user interfaces," *Journal of Universal Computer Science*, vol. 14, 2008, p. 1435–1446.

[50] V.M.R. Penichet, M.D. Lozano, J. a Gallud, and R. Tesoriero, "User interface analysis for groupware applications in the TOUCHE process model," *Advances in Engineering Software*, vol. 40, Dec. 2009, pp. 1212-1222.

[51] A. Durán, "Un Entorno Metodológico de Ingeniería de Requisitos para Sistemas de Información," Universidad de Sevilla, 2000.

[52] A. van Lamsweerde, "Goal-Oriented Requirements Engineering: A Guided Tour," *Proceedings of the Fifth IEEE International Symposium on Requirements Engineering*, IEEE Computer Society, 2001, pp. 249-263.

[53] R. Balzer, "A 15 Year Perspective on Automatic Programming," *IEEE Transactions on Software Engineering*, vol. SE-11, 1985, pp. 1257-1268.

[54] N.S. Prywes and A. Pnueli, "Compilation of Non-Procedural Specifications into Computer Programs," *IEEE Transactions Software Engineering*, vol. 3, 1983, pp. 267-279.

[55] B. Selic, “The pragmatics of model-driven development,” *IEEE Software*, vol. 20, 2003, pp. 19-25.

[56] S. Kent, “Model Driven Engineering,” *Integrated Formal Methods*, vol. 2335, 2002, pp. 286-298.

[57] OMG Architecture Board ORMSC, “Model Driven Engineering,” *OMG document number ormsc/2001-07-01*, 2001.

[58] L. Sorokin, F. Montero, and C. M\"artin, “Flex RIA development and usability evaluation,” *Proceedings of the 2007 international conference on Web information systems engineering*, Springer-Verlag, 2007, p. 447–452.

[59] V. López-Jaquero, J. Vanderdonckt, and F. Montero, “Towards an Extended Model of User Interface Adaptation: The ISATINE Framework,” *Ifip International Federation For Information Processing*, 2008, pp. 374-392.

[60] C. Gutwin and S. Greenberg, “Effects of awareness support on groupware usability,” *Interface*, 1998, pp. 511-518.

[61] J. Carroll, M. Rosson, G. Convertino, and C. Ganoe, “Awareness and teamwork in computer-supported collaborations,” *Interacting with Computers*, vol. 18, 2006, pp. 21-46.

[62] N. Padak and G. Padak, “Guidelines for Planning Action Research Projects. Research to Practice,” 1994.

[63] H. Fardoun, F. Montero, and V. López-Jaquero, “eLearniXML: Towards a model-based approach for the development of e-Learning systems considering quality,” *Advances in Engineering Software*, vol. 40, 2009, pp. 1297-1305.

[64] F. Montero, V. López-Jaquero, E. Navarro, and E. Sánchez, “Computer-aided relearning activity patterns for people with acquired brain injury,” *Computers & Education*, vol. 57, Aug. 2011, pp. 1149-1159.