UNIVERSIDAD DE DEUSTO

Facultad de Ingeniería

Elaboración con Vistas de los Requisitos de una Línea de Productos Software

---

Tesis doctoral presentada por Ana Monserrat Ereño Incera

Dentro del Programa de Doctorado en Ciencias de la Computación

Dirigida por la Dra. Rebeca Cortázar y la Dra. Asunción Barredo

La Directora La Directora La Doctoranda

Bilbao, Agosto de 2013

RESUMEN

La ingeniería de requisitos (IR) y en particular la ingeniería de requisitos para líneas de producto software (IR_LPS) son campos de investigación prioritarios dentro de la ingeniería de software.

El interés creciente respecto a la ingeniería de requisitos se encuentra plenamente justificado por ser esta la etapa en la que se encuentran los problemas más graves y con mayor impacto sobre el desarrollo de productos software. El interés por las Líneas de Producto Software se encuentra justificado por su propia filosofía de desarrollo basada en la flexibilidad de los productos y la adaptabilidad a las necesidades concretas de cada cliente. En la IR_LPS el principal problema está en capturar las expectativas de todo un sector del mercado, y al mismo tiempo conseguir que el producto final satisfaga las necesidades individuales de cada uno de los usuarios, clientes, etc. El problema básico es la comunicación. La comunicación entre la empresa desarrolladora y el cliente/mercado, no resulta sencilla y de este modo integrar todas las perspectivas sobre el producto se convierte en un área de investigación crítica. Así, dentro de esta corriente de orientación al cliente, la ingeniería de requisitos debe adaptarse. No sirven las fórmulas tradicionales.

El presente trabajo aporta un método que da soporte a las relaciones entre las necesidades del mercado y las características de los productos integrantes de una línea de productos. Este método se basa en los enfoques de ingeniería de requisitos basados en vistas que han demostrado ser útiles para extraer, definir y estructurar las especificaciones de LPS. Para mostrar la validez de la propuesta se ha utilizado la técnica de experimentos formales. La validación se ha realizado desde una doble perspectiva. Por un lado se ha validado la filosofía fundamental del método y por otro se ha validado la utilidad del método en sí mismo.

ABSTRACT

Requirements Engineering, and specially Requirements Engineering for Software Product Lines (RE_SPL) are priority research areas nowadays in Software Engineering.

The growing interest regarding Requirements Engineering is fully justified as this is the stage where relevant difficulties, that have a deep impact on the development of software products, are discovered. In addition, the interest in Software Product Lines is justified by its own development philosophy, based on product flexibility and adaptability to the specific needs of each client.

It is widely believed that the main problem in RE_SPL is to gather the expectations of an entire sector in the market and at the same time, make the final product meet the individual needs of the users, customers, etc. In true, the basic problem is communication. Communication between developers and customers/markets is not easy, so integrating all perspectives about the product becomes a critical research area. In consequence, given this trend of orientation towards the customer, Requirements Engineering activities must be adapted. The traditional ways do not work anymore.

This thesis provides a method for supporting the relationship between market needs and product features in a product line. This method is based on approaches related to View Oriented Requirements Engineering, which have proven useful for eliciting, identifying and structuring SPL specifications.

In order to show the validity of this proposal, formal experiments were conducted. The validation was performed from two different perspectives. On one hand, the fundamental philosophy of the method was validated; on the other hand, the usefulness of the method itself was assessed.

AGRADECIMIENTOS

Agradezco a Rebeca Cortázar y a Asunción Barredo su dedicación e inestimable ayuda en todos los aspectos. Sin ellas esto no hubiera sido posible.

A mis compañeros de Mondragón Unibertsitatea que durante todos los años que he invertido en la realización de esta tesis me han apoyado económica y anímicamente.

A los alumnos de Mondragón Unibertsitatea que durante este tiempo me han ayudado de forma desinteresada en todo tipo de tareas relacionadas con el trabajo de tesis.

A mi familia, a los que han aparecido durante este trabajo y a aquellos que ya no están pero que tuvieron una participación importantísima.

Finalmente, a las distintas administraciones públicas, instituciones y empresas que, de una u otra manera, han apoyado con su financiación el desarrollo de ciertas partes y actividades de esta tesis.

A todos, mi más sincero agradecimiento.

Indicé de contenidos

Índice de figuras

Índice de tablas

CAPÍTULO 1- INTRODUCCIÓN

La presente investigación doctoral se enmarca en el área de Ingeniería de Requisitos para Líneas de Producto Software (IR_LPS). En este capítulo presentamos el contexto actual del problema planteado: identificamos las cuestiones a cubrir, que representan las motivaciones de la investigación, y explicamos la relevancia del tema. A continuación, planteamos la hipótesis de trabajo, los objetivos que guían su desarrollo y las aportaciones previstas. Finalmente se proporciona una visión general de la estructura de esta memoria junto con un breve resumen del contenido de cada uno de sus capítulos.

1.1 CONTEXTO DE TRABAJO

Está ampliamente reconocido [Barlas96], [GAO79], [Gibbs94], [America00] que los problemas en las fases iniciales del desarrollo de software, concretamente durante la especificación de requisitos, son probablemente la principal razón de los fallos en los proyectos: los plazos se alargan, los sistemas desarrollados no cumplen las expectativas reales de sus usuarios, su ejecución es insatisfactoria, etc.

Existen diversos estudios sobre los factores de éxito y fracaso en proyectos de desarrollo de software. Entre ellos destaca el informe CHAOS, que realiza desde 1995 el Standish Group, y cuya última edición es de abril de 2012 [Standishgroup95], [Standishgroup12]. En el realizado en 1995 se concluye que solo un 16% de los proyectos analizados terminó correctamente, mientras que un 31% fueron cancelados y un 52% se terminaron pero los clientes no quedaron satisfechos. En el que se realizó en 2012 se concluye que el 44% de los proyectos

fueron deficientes (con retraso, por encima del presupuesto y/o con menos requisitos de los esperados) mientras que el 24% fracasaron (se cancelaron o se finalizaron pero el producto nunca se usó). Los jefes de proyecto de los proyectos analizados en estos estudios coincidieron en señalar como factores de éxito los siguientes: la participación activa de los usuarios, el apoyo de los directivos y unos requisitos claros. Los tres principales factores de fracaso eran: la falta de información por parte de los usuarios, unas especificaciones incompletas y unos requisitos cambiantes. Otro estudio realizado en 1996 el proyecto ESPITI (European Software Improvement Training Initiative) cuyo objetivo era analizar los principales problemas de desarrollo de software a nivel europeo, obtuvo resultados muy similares al de CHAOS. Éste reveló que los mayores problemas estaban relacionados con la especificación de requisitos y gestión de los requisitos [PM96]. Algo que corroboran estudios posteriores como [Kamsties98], [Nikula00].

Todos coinciden en que una ingeniería de requisitos deficiente es uno de los principales factores del fracaso software. Unos “requisitos incompletos” son la principal causa de cancelación de proyectos de desarrollo de software, mientras que unos “requisitos claros” conforman el factor de éxito más importante dentro de las prácticas de ingeniería de software.

1.2 PLANTEAMIENTO DEL PROBLEMA

El éxito de los productos de software se mide en base al grado de cumplimiento de las expectativas de los usuarios. Desafortunadamente, resulta difícil encontrar un usuario completamente satisfecho. Como dice F. Brooks, la parte más dura y complicada de construir un sistema software es precisamente decidir qué construir [Brooks87]. Podemos afirmar que la parte más complicada del trabajo conceptual es establecer los requisitos. Ninguna otra parte del trabajo conceptual es tan

complicada como establecer los requisitos ni tiene tanta influencia en la obtención de un producto inexacto.

La disciplina, dentro de la ingeniería de software, encargada de lidiar con los deseos y necesidades de los usuarios, es la ingeniería de requisitos (IR) [Sommerville97], [Davis03].

Si nos centramos en el desarrollo de software a medida, este problema se puede solucionar de una forma más o menos sencilla, mediante la utilización de diferentes técnicas, métodos y/o herramientas. Ésta ha sido un área de investigación muy trabajada por la comunidad de ingenieros del software durante los últimos años. Sin embargo, si nos centramos en el desarrollo de productos software orientados a mercado, la cuestión se complica. Bajo este enfoque, ya no existe un cliente con unas necesidades concretas sobre un producto, todo el mercado es cliente potencial de los productos. Se hace referencia al mercado como el "Mercado del Comprador" en el que las empresas deben satisfacer los requisitos para cada cliente. Así, si se pretende que las estrategias tengan éxito deben estar orientadas al consumidor. Las empresas han reaccionado a este cambio en el mercado con una modificación en la estrategia de desarrollo de software, y la industrialización del software se configura como una de las nuevas vías de desarrollo en el sector de las tecnologías de la información.

Ante un desarrollo orientado a mercado, o un desarrollo de líneas de producto software, se deben capturar las expectativas de todo un sector del mercado, y al mismo tiempo el producto final debe satisfacer las necesidades individuales de cada uno de los usuarios, clientes, etc. Integrar todas las perspectivas sobre el producto se convierte en un punto crítico. Así, dentro de esta corriente de orientación al cliente, la ingeniería de requisitos debe adaptarse, las fórmulas tradicionales no sirven.

1.3 IMPORTANCIA DEL PROBLEMA

La línea de investigación propuesta en esta memoria combina dos cuestiones: la ingeniería de requisitos y las líneas de productos software.

Los estudios mencionados en el primer apartado de este capítulo convencieron a la comunidad de la ingeniería de software de la necesidad de desarrollar la ingeniería de requisitos (IR). Actualmente, se puede afirmar que se ha consolidado como un área de investigación en desarrollo:

- Existen numerosos trabajos sobre ingeniería de requisitos tras 35 años de existencia [Nuseibeh00], [Lamsweerde09], [Xie09], [Withall07].
- Se organizan congresos específicos como:
    - El IEEE International Symposium on Requirements Engineering (RE), que se celebra los años impares desde 1993 organizado por IEEE, ACM, IFIP y varias asociaciones más y la IEEE International Conference on Requirements Engineering (ICRE), que se celebra los años pares desde 1994 organizado por el IEEE. Estas dos conferencias están actualmente unificadas.
    - El Workshop em Engenharia de Requisitos (WER), que se celebra anualmente desde 1998.
- Se publican revistas especializadas como el Requirements Engineering Journal, o monográficos sobre ingeniería de requisitos en IEEE Software.
- Se financian proyectos europeos y nacionales como:
    - NATURE (Novel Approaches to Theories Underlying Requirements Engineering).
    - REAIMS (Requirements Engineering Adaptation and Improvement for Safety and dependability).

  - CREWS (Cooperative Requirements Engineering With Scenarios).
  - CICYT MENHIR (Metodologías, Entornos y Nuevas Herramientas para la ingeniería de requisitos).
- Se ha creado una red europea RENOIR (Requirements Engineering Network of International cooperating Research groups).

Los esfuerzos investigadores en este área han conducido a propuestas de diferentes enfoques, que incluyen estándares como IEEE/EIA 12207.0-1996 [IEEE96**],** ANSI/IEEE Standard 830-1998 [IEEE98], procesos genéricos [Jacobson97], [Robertson99], guías de buenas prácticas [Sommerville97], [Wiegers99], etc.

En cuanto a las LPS, se puede afirmar que están adquiriendo gran importancia tanto en el ámbito investigador, como de aplicación real. Su origen se puede remontar a los años 60. En una conferencia titulada "*Mass produced software components*" realizada en 1968, [Mcllroy68] se introdujo el concepto de ***reutilización*** como la clave para el diseño eficiente de nuevos productos de software. A partir del año 2000, el interés de las LPS crece sustancialmente. Esto se refleja tanto en las crecientes publicaciones sobre el tema, entre otras:[Bosch00], [Jayazeri99], [Atkinson02], [Barlas96], [Yoshimura08], [Jenesen09], [Bell07], [Babar10], así como las conferencias y workshops dedicadas exclusivamente al área, tales como *Software Product Line Conference (SPLC), Component-Based Software Engineering (CBSE), International Conference on Component Deployment (CD), International Conference on Design Theory and Methodology (DTM), Formal Foundations of Embedded Software and Component-Based Software Architectures (ETAPS), International Conference on Software Reuse (ICSE), IEEE/IFIP Working Conference on Software Architecture (WICSA),* etc.

1.4 LA INVESTIGACIÓN

La habilidad de una organización para crear productos software con éxito, depende en gran medida de cómo obtenga, analice y utilice la información asociada a los productos que crea. Para desarrollar un nuevo producto o servicio, es esencial tener en cuenta "la voz del consumidor", sin olvidar al resto de personas implicadas. Hay que conseguir que el consumidor articule, estructure y priorice lo que quiere y necesita de un producto o servicio. Una comunicación directa con los consumidores permite aprender de ellos y ajustar los productos a sus necesidades.

En el desarrollo de software a medida, donde los desarrollos se centran en un único producto, el problema básico siempre ha sido la comunicación, y en el enfoque LPS el problema se agudiza. La comunicación entre la empresa desarrolladora y el cliente/mercado, no resulta sencilla por dos motivos principales [Burchil97]:

- El cliente final de una línea de productos no es una única persona/entidad, es todo el mercado. Esta situación presenta un escenario donde:
    - Los consumidores tienen necesidades múltiples y contrapuestas.
    - Las necesidades de los consumidores varían (segmentos).
    - Las necesidades se miden mediante escalas no monetarias.
    - Es complicado evaluar los beneficios (necesidad de predecir la credibilidad).
- El factor tiempo es crítico. Bajo el enfoque LPS no sólo se tienen en cuenta las necesidades actuales, sino también las futuras. Es necesario predecir las necesidades futuras del mercado, para ser capaces de integrarlas en los productos de la línea.

Para conseguir una descripción completa y coherente del sistema, debemos tener en cuenta las diferentes visiones o perspectivas de los usuarios del mismo. Cada una

de estas perspectivas, por sí sola, resulta incompleta, es una descripción parcial del sistema final. Una de las claves de éxito del desarrollo de nuevos productos es sintetizar el conocimiento poseído por los participantes en el proceso. Los enfoques de ingeniería de requisitos basados en vistas o perspectivas, han demostrado ser útiles para extraer, definir y estructurar las especificaciones de LPS [Troya99]. Así, en este trabajo de tesis abordamos las relaciones entre las necesidades del mercado y las características de los productos integrantes de una línea de productos.

En relación con esta problemática nos planteamos la hipótesis siguiente: "*Es posible integrar y utilizar el conocimiento de fuentes diversas en un método sistemático para la definición de una línea de productos software que dé valor al usuario, en las fases iniciales del desarrollo de software".*

Para su ratificación nos marcamos como objetivo el desarrollo un método formal para estructurar una familia de productos en función a las demandas y preferencias del mercado objetivo, y en consonancia los siguientes objetivos específicos:

1. Realizar un estudio en profundidad del estado del arte relativo al enfoque de desarrollo de líneas de producto software, de la ingeniería de requisitos, y del enfoque de vistas o perspectivas.
2. Proponer, desarrollar y validar un método que proporcione soporte al proceso de IR _LPS mediante la integración de vistas o perspectivas.

El resultado de esta investigación pretende realizar una aportación novedosa, innovadora y con impacto en un área de interés creciente como es la Mass Customization. Se trata de optimizar el Proceso de Creación de Nuevos Productos de Software Intensivos, gracias a la integración en el mismo de las necesidades latentes en el mercado.

1.5 ESTRUCTURA DE LA TESIS

Este trabajo está estructurado en 8 capítulos que se describen a continuación.

El presente capítulo enmarca el contexto de la tesis. En él se justifica el estudio en relación a la hipótesis establecida y se exponen los objetivos generales de la disertación.

El capítulo 2 recoge la metodología de investigación.

El capítulo 3 muestra un recorrido por los antecedentes del desarrollo de software para líneas de producto y la ingeniería de requisitos, finalizando con una comparativa de los mismos.

El capítulo 4 expone la técnica IROP (Ingeniería de Requisitos Orientada a Perspectivas) y analiza su aplicabilidad como solución al problema planteado.

Los capítulos 5 y 6 forman el grueso del trabajo de tesis, materializado en el desarrollo de un método para la Elaboración de Requisitos para Líneas de Producto Software. En el 5 se muestra una visión general del método propuesto y en el 6 se describe en detalle el método desarrollado.

El capítulo 7 muestra los métodos empleados para la validación de la hipótesis planteada en esta tesis y los resultados obtenidos.

Finalmente, el capítulo 8 está dedicado a las conclusiones y resumen de las aportaciones derivadas de la investigación desarrollada, contrastadas con los objetivos planteados. Además, se incluyen las publicaciones en congresos internacionales realizadas durante la investigación de la tesis doctoral. Este capítulo finaliza con la propuesta de algunas líneas de investigación que quedan abiertas.

La memoria se completa con 3 anexos que muestran varios de los documentos utilizados durante las actividades de validación.

CAPÍTULO 2- METODOLOGÍA DE LA INVESTIGACIÓN

En este capítulo se expone la metodología de trabajo seguida para cumplir con los objetivos marcados en esta tesis. Dicha metodología, aglutina dos de los métodos de investigación más utilizados y recogidos por la literatura especializada para la validación y contrastación de la hipótesis de investigación. Estamos hablando del método Investigación-acción y de los experimentos.

De este modo, el presente capítulo se encuentra dividido en cinco secciones. En primer lugar se plantea la creciente necesidad de realizar actividades de experimentación en el ámbito de la ingeniería de software. A continuación se detallan los métodos más utilizados en este ámbito. Para continuar se explican en detalle los fundamentos teóricos de los métodos aplicados en el contexto de esta tesis: el método investigación-acción y los experimentos. El capítulo concluye con un resumen.

2.1 NECESIDAD DE EXPERIMENTACIÓN EN INGENIERÍA DE SOFTWARE

La investigación en el campo de la ingeniería difiere sustancialmente, tanto en objeto de estudio como en método, de la investigación en las tradicionalmente llamadas “ciencias”. Mientras las ciencias se ocupan del estudio de objetos y fenómenos existentes (física, metafísica, etc.), las ingenierías basan sus estudios en cómo hacer y/o cómo crear nuevos objetos. Es por esto que los métodos de investigación científicos no son siempre directamente aplicables a problemas de investigación de ingeniería [Marcos08]. Por ello, del mismo modo que en el siglo XVI surgieron nuevos métodos de investigación adecuados a la ciencia de la época, es

necesario ahora definir otros que sean aplicables a los problemas concretos de la ingeniería de software.

Dependiendo del tipo de ciencia, se utilizan unos u otros métodos de investigación, sin embargo, tal y como se propone en [America00], las ingenierías no encajan totalmente en las clasificaciones propuestas en la literatura, si bien están relacionadas con la mayor parte de las disciplinas que en ellas aparecen. Por este motivo, la búsqueda de un método apropiado para la investigación en la ingeniería de software, y su aplicación al desarrollo e implantación de sistemas de información, se ha convertido en un tema de investigación en sí mismo [Glass02], [Gregg01], [Dieste07], [Myers02], [Dieste09], [Dieste12], [Fernandez09], [Fernandez10].

2.2 MÉTODOS DE INVESTIGACIÓN EN INGENIERÍA DE SOFTWARE

En la ingeniería de software existen distintos métodos de trabajo, adecuados en función del tipo de investigación que se está realizando. Aunque los métodos de investigación pueden categorizarse de diversas formas, una clasificación ampliamente aceptada es la que los divide en métodos *cuantitativos* o *deductivos* y métodos *cualitativos o inductivos*.

Los métodos cuantitativos y cualitativos tienen objetivos diferentes que podrían ser resumidos como “el desarrollo de la teoría” y “el análisis de la teoría” respectivamente [Mendoza06]. En la *tabla 1* se muestran las diferencias entre estos métodos, pero cabe destacar como principal diferencia entre la investigación cualitativa y cuantitativa, la naturaleza de los datos y el análisis realizado. En la investigación cualitativa los datos son poco estructurados, obtenidos mediante entrevistas, observaciones y discusiones en grupo y no se suelen analizar de forma

estadística. Mientras, en la investigación cuantitativa los datos están más estructurados y se utilizan técnicas estadísticas para su análisis.

Investigación Cuantitativa	Investigación Cualitativa
Basada en la inducción probabilística	Centrada en la fenomenología y comprensión
Medición controlada	Observación sin control
Objetiva	Subjetiva
Inferencias más allá de los datos	Inferencias de sus datos
Confirmatoria, inferencial, deductiva	Exploratoria, inductiva y descriptiva
Orientada al resultado	Orientada al proceso
Datos sólidos y repetibles	Datos ricos y profundos
Generalizable	No generalizable

Tabla 1: Diferencias entre investigación cualitativa y cuantitativa.

Los métodos de investigación cuantitativos son especialmente adecuados para el estudio de fenómenos u objetos naturales, y en este tipo de métodos se podrían encuadrar los métodos deductivos y empíricos, siendo algunos ejemplos los estudios y los experimentos de campo, los experimentos de laboratorio, la simulación experimental, el encuestado, la investigación de archivo, la de opinión o los experimentos adaptativos [Straub04]. El estudio de fenómenos culturales y sociales requiere otro tipo de métodos, que no se basen en experimentos ni teorías formales, sino en observaciones de los participantes, entrevistas, cuestionarios, documentos y textos (informes, artículos etc.), impresiones y reacciones del investigador, etc. Estos métodos se encuadran dentro de lo que se conoce como métodos cualitativos. Ejemplos de estos métodos son el método de investigación-acción, la investigación basada en casos de estudio, o la etnografía [Myers97].

En general los métodos cuantitativos son muy potentes en términos de validez externa ya que con una muestra representativa de la población hacen inferencia a dicha población a partir de una muestra con una seguridad y precisión definida. Por el contrario una limitación de los métodos cualitativos es su dificultad para generalizar.

Las ventajas e inconvenientes de los métodos cuantitativos vs los cualitativos se pueden ver en la *tabla 2.*

Métodos Cuantitativos	Métodos Cualitativos
Propensión a "servirse de" los sujetos del estudio	Propensión a "comunicarse con" los sujetos del estudio
Se limita a responder	Se limita a preguntar
Son débiles en términos de validez interna (nunca sabemos si miden lo que quieren medir), pero son fuertes en términos de validez externa (lo que encuentran es generalizable a la población)	Son fuertes en términos de validez interna, pero son debilidades en validez externa, no es generalizable a la población.

Tabla 2: Ventajas e inconvenientes entre métodos.

Hoy en día hay un predominio claro de la investigación cuantitativa en relación a la cualitativa. En el ámbito de la ingeniería de software, por ejemplo, los métodos cuantitativos han centrado la atención de la comunidad científica durante los últimos años. Mediante los métodos cuantitativos es posible evaluar nuevas aportaciones en la Ingeniería de software antes de que sean introducidas en los procesos software de las empresas [Host02], [Glass02]. A pesar de esto, los métodos cualitativos son totalmente aceptados como métodos de investigación válidos, y esta aceptación está creciendo en los últimos años. Ya desde finales de los años 90, los métodos cualitativos y en especial el de investigación-acción ha

recibido la atención y aceptación por parte de la comunidad investigadora en sistemas de información y su uso ha comenzado a ser habitual [Avison99].

Ambas corrientes, la cualitativa y la cuantitativa no deben considerarse extremos opuestos o paradigmas encontrados. La metodología cualitativa no es incompatible con la cuantitativa, y de hecho el empleo de ambos procedimientos, cuantitativos y cualitativos, en una investigación probablemente podría ayudar a corregir los sesgos propios de cada método. Éste es el caso del trabajo de investigación de esta tesis donde se empleará un método cualitativo, el de *investigación-acción*, como método conductor durante todo el ciclo de vida de la investigación, es decir, la identificación del problema, el planteamiento y refinamiento sucesivo de la solución y la experimentación no formal, y un método cuantitativo, los *experimentos en laboratorio*, para la fase de experimentación más formal.

2.3 INVESTIGACIÓN-ACCIÓN

El método Investigación-Acción es un método de investigación cualitativa. El término "Investigación-Acción" proviene del autor Kart Lewin [Lewin47], que lo utilizaba para describir una forma de investigación que podía enlazar el enfoque experimental de las ciencias sociales con programas de acción social que respondieran a los problemas sociales principales de entonces. Mediante la Investigación-Acción, Lewin argumentaba que se podían lograr en forma simultánea avances teóricos y cambios sociales.

Otra definición, más actual y por lo tanto más cercana a nuestro contexto de investigación es la de [Mctaggart91], que define este método de investigación como *"la forma que tienen los grupos de personas de preparar las condiciones necesarias*

para aprender de sus propias experiencias, y hacer estas experiencias accesibles a otros".

Este método de investigación obtuvo la aceptación y atención por parte de la comunidad investigadora en sistemas de información hace tiempo [Avison99], [Seaman99] y actualmente sigue siendo un método ampliamente aceptado [Kock01], [Wadsworth09].

La Investigación-Acción tiene una doble finalidad: generar un beneficio al "cliente" de la investigación y, al mismo tiempo, generar "conocimiento de investigación" relevante [Kock01]. Por tanto, Investigación-Acción es una forma de investigar de carácter colaborativo, que busca unir teoría y práctica entre investigadores y profesionales mediante un proceso de naturaleza cíclica. Esto se consigue gracias a la intervención de un investigador en la realidad del mencionado grupo. Los resultados de esta experiencia deben ser beneficiosos tanto para el investigador como para los profesionales.

2.3.1 PARTICIPANTES

En el campo de los sistemas de información, el cliente de una investigación es normalmente una organización a la cual el investigador suministra servicios tales como consultoría o desarrollo de software, a cambio de tener acceso a datos de interés para la investigación y, en muchos casos, de recibir financiación [Kock01]. En cualquier caso, el investigador que utiliza Investigación-Acción en sistemas de información sirve a dos entidades diferentes: el cliente de la investigación y la comunidad científica del área de sistemas de información. Las necesidades de ambos suelen ser muy diferentes y, a veces, opuestas entre sí. Intentar satisfacer ambas demandas es el principal desafío. Sirviendo tanto a las necesidades de los

profesionales como las del conocimiento científico, se añaden nuevos elementos a la investigación que hace que ésta sea más beneficiosa.

En la situación descrita, se pueden identificar diferentes roles en la Investigación-Acción como son el investigador, el objeto investigado, el grupo crítico de referencia (GCR) y los beneficiarios (véase la *tabla 3*) [Wadsworth09].

Nombre del Rol	Descripción
Investigador	El individuo o grupo que lleva a cabo de forma activa el proceso investigador
Objeto investigado	El problema a resolver
Grupo Crítico de Referencia (GCR)	Aquel para quien se investiga en el sentido de que tiene un problema que necesita ser resuelto y que también participa en el proceso de investigación (aunque menos activamente que el investigador)
Beneficiarios	Aquel para quien se investiga en el sentido de que puede beneficiarse del resultado de la investigación, aunque no participa directamente en el proceso

Tabla 3: Roles en Investigación – Acción.

2.3.2 La espiral en ciclos de la Investigación-Acción

El método investigación-acción se estructura en ciclos de investigación en espiral, contando cada ciclo con fases claves: fase de planificación, fase de acción y fase de reflexión, generando esta última un nuevo ciclo de investigación (véase la *figura 1*).

La primera fase de la Investigación-Acción es determinar la temática sobre la que se va a investigar. No se trata de identificar problemas teóricos de interés para los investigadores, sino de problemas cotidianos vividos como tales por los desarrolladores software, que puedan ser resueltos a través de soluciones prácticas.

La segunda fase es la reflexión inicial o diagnóstico. En ella hay que cuestionarse sobre el origen y evolución de la situación, la posición de las personas implicadas en la investigación ante ese problema, los aspectos más conflictivos, qué formas adoptan tales conflictos, cuáles son las formas de respuesta, y qué correspondencia o falta de correspondencia existe entre la teoría y la práctica. Es muy importante, en esta fase, ser capaces de describir y comprender lo que realmente se está haciendo, así como los valores que sustentan las prácticas realizadas. Todas estas cuestiones permitirán identificar los obstáculos tanto subjetivos como objetivos a las propuestas de cambio.

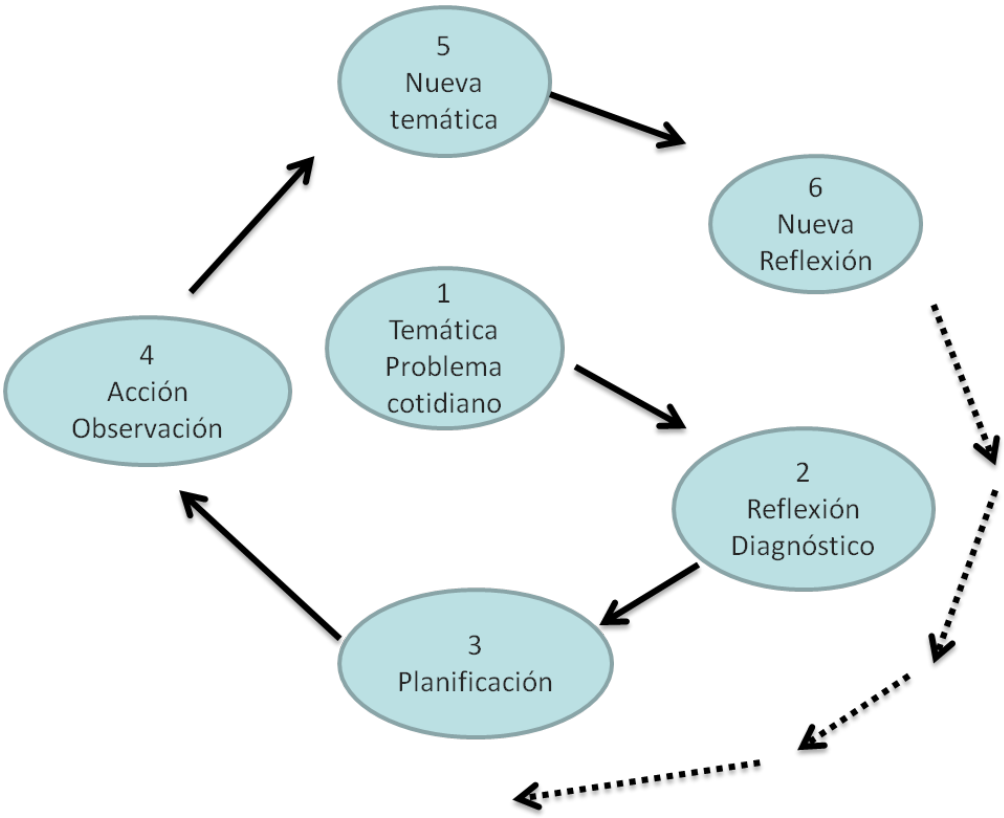


Figura 1: Espiral Investigación – Acción.

La tercera fase es la planificación. El plan general que se elabore debe ser flexible, para que pueda incorporar aspectos no previstos en el transcurso de la investigación. En este plan inicial de la investigación-acción se debe: 1) describir la preocupación temática, 2) presentar la estructura y las normas de funcionamiento del grupo de investigación, 3) delimitar los objetivos, atendiendo a los cambios que se pretenden conseguir en las ideas, las acciones y las relaciones sociales, 4) presentar, lo más desarrollado posible, un plan de acción, 5) describir cómo se va a relacionar el grupo de investigación con otras personas implicadas o interesadas en los cambios esperados, 6) describir cómo se van a controlar las mejoras generadas por la investigación.

La cuarta fase corresponde a la acción-observación. En esta fase, se pone en marcha el plan y se registran datos que serán utilizados en una reflexión posterior.

La espiral comienza de nuevo, debido a que en la fase de reflexión se produce un nuevo esclarecimiento de la situación problemática, y se repiten de nuevo todos los pasos.

2.3.3 APLICACIÓN DE INVESTIGACIÓN-ACCIÓN

A lo largo del desarrollo del trabajo de investigación, se ha aplicado el método Investigación-Acción con el objetivo de generar conocimiento colaborativo y aplicarlo en el desarrollo de un método para la elaboración basada en vistas de los requisitos de una línea de productos de software. Esto ha resultado posible mediante una subvención del programa INTEK durante los años 2005-2007. El proyecto subvencionado, denominado TRANSER, tenía como objetivo el desarrollo de una herramienta informática de gestión integral del transporte dirigida a los proveedores de servicios logísticos.

En este proyecto han participado como entidades dos empresas tecnológicas Adur Software Productions S.Coop. y A2, con el apoyo de la Universidad de Mondragón. El consorcio formado entre tres entidades, dos empresas y un centro de investigación universitario, ha proporcionado un equilibrio adecuado a la realización del proyecto. Las empresas tecnológicas han aportado el conocimiento de la actividad de Transporte, basado en 20 años de recorrido, más de 3.000 programas desarrollados en el área, y un mantenimiento actual de más de 100 instalaciones con cerca de 3.000 usuarios. La Universidad de Mondragón, agente de la RVCTI, ha prestado su conocimiento en constante renovación para dar al proyecto el enfoque adecuado, soportado por las tendencias más asentadas en el análisis y desarrollo de soluciones de software, de cara a obtener un producto basado en un análisis detallado de los requisitos del cliente. En la *tabla 4* se muestran los roles de los distintos participantes.

El rol jugado por la Universidad de Mondragón en este proyecto, como se muestra en la *tabla 4*, ha sido el de investigador, tanto en el enfoque como en las metodologías a utilizar en las fases iniciales de desarrollo. De este modo, a lo largo de las distintas etapas del proyecto, se fueron presentando las diferentes alternativas (tanto metodológicas como tecnológicas) necesarias para abordar dichas etapas. La elección de una alternativa y su adaptación al proyecto corrió siempre a cargo de las dos empresas participantes en el mismo. Esta situación ha permitido a la Universidad de Mondragón sumergirse en un entorno real y al mismo tiempo realizar un estudio observacional analizando los comportamientos en cada etapa. Puede decirse por lo tanto, que se ha aplicado un método de investigación descriptivo donde el objetivo ha sido identificar aspectos relevantes de la realidad. Concretamente, se han investigado aquellos aspectos relacionados con los conocimientos, las actitudes y las prácticas de ingeniería de requisitos para líneas de producto software y la interpretación que el equipo de desarrollo tenía de las mismas.

La posibilidad de observar en un contexto real aspectos particulares a las fases iniciales de un desarrollo de software orientado al mercado, y contrastar estos aspectos con los expuestos en la literatura relacionada ha permitido enriquecer el método y acercarlo a la investigación aplicada. El trabajo realizado y las conclusiones obtenidas han sido publicadas en el congreso WER 2008 –Workshop on Requirements Engineering [Ereño08].

Nombre del Rol	Descripción
Investigador	Universidad de Mondragón
Objeto investigado	Fases iniciales del desarrollo de software orientado a mercado
Grupo Crítico de Referencia (GCR)	Adur Software Productions S.Coop. y A2
Beneficiarios	El mercado

Tabla 4: Roles utilizados con el método investigación-acción.

2.4 EXPERIMENTACIÓN

Los experimentos son una técnica cuantitativa utilizada para comprobar el comportamiento de un determinado fenómeno bajo unas condiciones establecidas. Se llevan a cabo en un entorno artificial, creado por el investigador, y donde las condiciones pueden ser controladas [Straub04], [Boudreau01], [Dix93].

La experimentación es un método de investigación en el que uno o más factores (variables independientes) sistemáticamente cambian, para determinar si tales variaciones afectan a uno o más factores (variables dependientes).

El principal objetivo de un experimento es determinar las situaciones en las que ciertas afirmaciones son verdaderas. Los experimentos pueden proporcionar el

contexto en el que ciertos estándares, métodos y herramientas son adecuados, y permiten sacar conclusiones acerca de las relaciones entre la causa y el efecto para la cual se formuló la hipótesis.

En este apartado se describen los factores esenciales que hay que considerar a la hora de llevar a cabo una buena planificación y realización de experimentos controlados, con el fin de obtener resultados que sean creíbles [Juristo01], [Wohlin00], [Perry00], [Briand00], [Kitchenham02]. En el *Capítulo 7 - Validación*, se describe en detalle el experimento realizado para validar la hipótesis planteada.

2.4.1 PARTICIPANTES

El tipo de personas involucradas en un experimento de laboratorio depende mucho de la modalidad del mismo y de sus objetivos. En los experimentos con un carácter social, refiriéndose con este término a aquellos en los que intervienen personas como objeto de estudio (bien de su comportamiento en sí mismo o bien de los datos o resultados que éstos produzcan), pueden distinguirse dos roles:

- Conductor. Quien guiará el experimento, y su participación o influencia sobre los participantes podrá ser mayor o menor en función del tipo de experimento, las premisas o condiciones de partida y su objetivo.
- Sujeto. Son los objetos del estudio (habitualmente los usuarios). Operarán en las condiciones establecidas por el experimento y su comportamiento y/o resultados generados serán evaluados con el fin de verificar una teoría o deducir nuevo conocimiento.

2.4.2 EL PROCESO EXPERIMENTAL

Los experimentos necesitan ser planificados de forma cuidadosa, si se pretende obtener de ellos resultados útiles y significativos. Por ello es necesario seguir un proceso experimental como el que se propone en [Wohlin00]. Este proceso consta de las siguientes etapas (véase la *figura 2*):

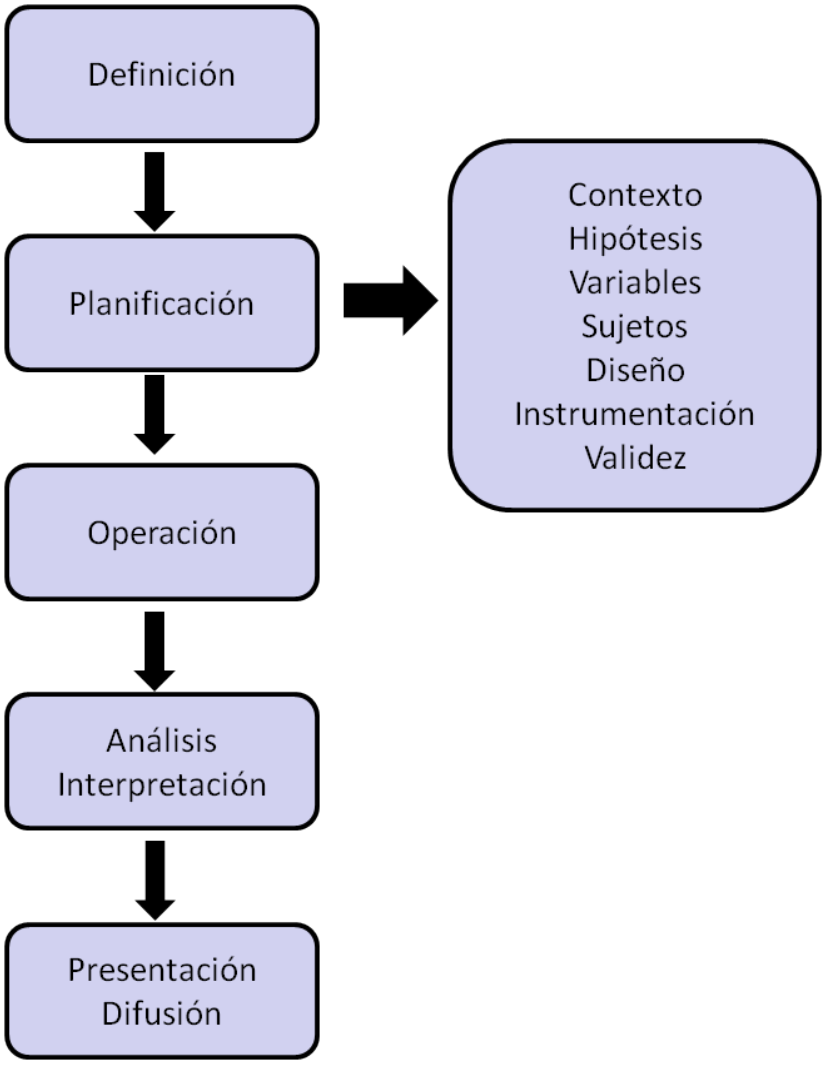


Figura 2: Proceso de Experimentación

1. Definición. Se define el experimento en términos del problema y los objetivos.
2. Planificación. Se determina "como" se va a conducir el experimento, es decir se realiza el diseño del experimento. La planificación consta de los 7 pasos siguientes:
   2.1. Selección del contexto: se elige el entorno en el cual se ejecutará el experimento (entorno real, ficticio, simulado, etc.).
   2.2. Formulación de hipótesis: se formulan las hipótesis a validar con el experimento.
   2.3. Selección de variables dependientes e independientes.
      - Variable dependiente: es aquella variable que se mide en un experimento. Las variables dependientes deben derivar directamente de la hipótesis.
      - Variable independiente: es aquel factor en un experimento que sistemáticamente cambia el investigador.

   La elección de las variables lleva asociada la elección de la escala de medición y el rango.

   2.4. Selección de sujetos participantes: hay que tener en cuenta que deben ser representativos de toda la población de sujetos.
   2.5. Diseño del experimento en función de las hipótesis y las variables seleccionadas. Un experimento bien diseñado permitirá la réplica del mismo. Este paso está relacionado con la etapa de Interpretación. Así el análisis estadístico que apliquemos dependerá en gran medida del diseño elegido y viceversa.
   2.6. Instrumentación: se prepara la implementación práctica del experimento. Se preparan los documentos, guías, procesos, sesiones de formación, etc. necesarios para la ejecución del experimento.

2.7. Evaluación de la validez: En esta etapa se evalúan los aspectos que pueden amenazar a la validez del experimento. Para ello se consideran cuatro tipos de validez:
   - Validez de constructo. Define hasta qué punto las medidas utilizadas para medir las variables independientes y dependientes, miden fielmente los conceptos que intentan medir.
   - Validez interna. Se refiere al grado de confiabilidad con el que podemos asegurar la relación causa-efecto entre las variables independientes y dependientes.
   - Validez externa. Consiste en el poder de generalización de los resultados, no sólo para el entorno donde se ha desarrollado el experimento sino también en otros entornos.
   - Validez de la conclusión. Define hasta qué punto las conclusiones obtenidas son estadísticamente válidas.

3. Operación. Se lleva a cabo el experimento y se recogen los datos empíricos.
4. Análisis e interpretación. Los datos recogidos se analizan utilizando técnicas estadísticas. Además se interpretan los resultados obtenidos.
5. Presentación y difusión. Se elabora un informe sobre los resultados para que los mismos sean difundidos con el objetivo de otros investigadores puedan replicar el experimento.

2.5 RESUMEN

En este capítulo se ha profundizado en los fundamentos teóricos de los métodos de investigación utilizados en esta tesis, concretamente los métodos investigación-acción y experimentos.

El método investigación-acción nos ha permitido unir teoría y práctica además de ir aprendiendo y generando conocimiento gracias a las experiencias realizadas.

Los experimentos han sido la técnica utilizada para validar la hipótesis planteada. En el capítulo 7 se explica en detalle la definición, preparación y ejecución del experimento, el cual se ha realizado siguiendo el proceso propuesto por Wohlin en [Wohlin00]. Esta información también se puede consultar en los artículos presentados en los congresos JISBD 2007 y VASOP 2010 respectivamente [Ereño07], [Ereño10].

CAPÍTULO 3- ESTADO DEL ARTE

En este capítulo contextualizamos la participación humana en las fases iniciales de un proceso de desarrollo de software, bajo el enfoque de desarrollo de líneas de producto software. En primer lugar profundizamos en el enfoque de desarrollo de software para líneas de producto; así se explica su origen, se detalla el proceso que lo soporta y se presentan los estudios más relevantes realizados. Posteriormente profundizamos en la fase inicial del desarrollo de software: la ingeniería de requisitos. Llegados a este punto tenemos el conocimiento suficiente para entender la casuística específica de la ingeniería de requisitos bajo el enfoque de líneas de producto software, y así señalar las diferencias más apreciables respecto al desarrollo de software a medida. El capitulo concluye con el planteamiento del problema.

3.1 DESARROLLO DE SOFTWARE PARA LÍNEAS DE PRODUCTO

En los siguientes apartados se analiza la aparición de este enfoque de desarrollo, se explica en detalle el proceso que soporta su uso y se muestran los antecedentes y la situación de la práctica del mismo.

3.1.1 ORIGEN DEL DESARROLLO DE SOFTWARE PARA LÍNEAS DE PRODUCTO

En la ingeniería de software, la aparición de un nuevo paradigma no sucede de forma arbitraria. Surge de necesidades concretas que permiten generar una

evolución en la forma de programar, diseñar y modelar. Esto es lo que ha sucedido en el caso de las líneas de producto software.

El mercado actual, tan competitivo y cambiante, ha redefinido la forma de trabajar que tienen las compañías. Los clientes, hasta ahora considerados como un gran mercado homogéneo, se han transformado en individuos cuyos deseos y necesidades individuales, pueden y deben ser determinadas y colmadas para cada uno de ellos, es decir, el "Mercado del Comprador" en el que las empresas deben satisfacer los requisitos de cada cliente. Esta situación de mercado, es radicalmente distinta al mercado dominado por el vendedor de los años 60, 70 y 80, que estaba caracterizado por una alta demanda y una relativa escasez de suministros. En aquel momento, las empresas produjeron grandes volúmenes de productos idénticos, basándose en técnicas de producción de masa. Sin embargo, el mercado dominado por el comprador, está forzando a las empresas, a entregar mejores productos, con mayor rapidez y a menor coste; productos ajustados a las necesidades de cada cliente individual.

Este nuevo mercado se rige por el concepto de ***"Comercialización en Masa".*** Este término hace referencia a la fabricación de productos, utilizando técnicas de fabricación en masa, pero dotando a la línea de producción de ciertos grados de libertad, que permita la variabilidad del producto, para generar toda una gama o familia de productos.

Ante este nuevo escenario, los distintos sectores se han visto obligados a reaccionar buscando "soluciones" que traten de abordar esta problemática. Si nos centramos en el sector del desarrollo de software, vemos que esa "solución" se basa en la *Reutilización*. Esta no es una solución novedosa. Echando la vista atrás, vemos que la historia de la programación es como una espiral creciente donde el elemento clave que origina cada vuelta de espiral o cada salto es la *Reutilización* (véase la *figura 3*). Ya en los últimos años de la década de los sesenta, la idea de construir

sistemas mediante la composición de *Componentes Software* fue presentada como solución a la afamada *Crisis del Software* por M. D. McIlroy [America00]. Durante la década de los setenta se propugnó la reutilización de los *Módulos*, mientras que en los años ochenta la influencia del *Paradigma Orientado a Objetos* hizo que la *Clase* se convirtiera en la unidad de reutilización. Sin embargo, todas estas tendencias fallaban en conseguir un enfoque sistemático de reutilización. Todas ellas daban lugar a iniciativas individuales, frecuentemente realizadas a pequeña escala. Para solucionar este problema surgen los *Frameworks* [Wirfsbrock90] y la *Programación Orientada a Componentes* [Szyperski98].

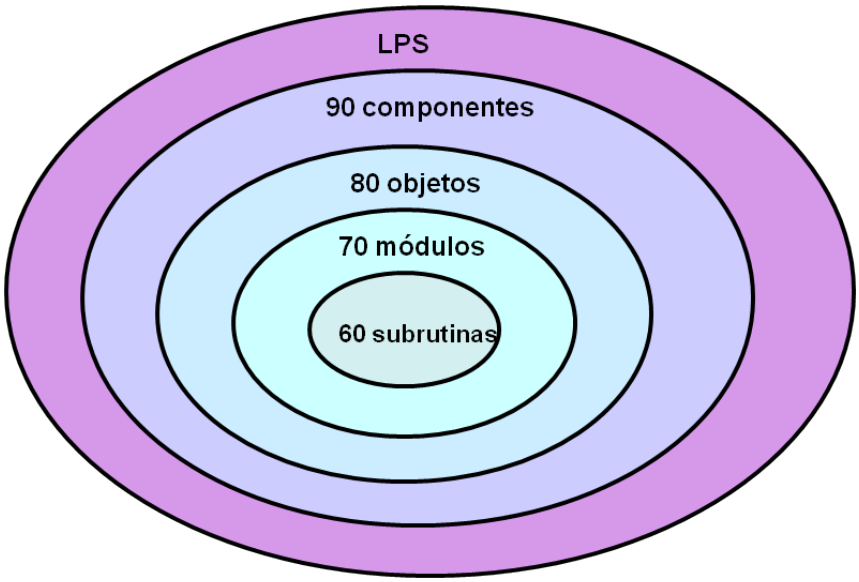


Figura 3: Evolución en las formas de desarrollo de software

La Reutilización del Software es uno de los objetivos fundamentales dentro de la ingeniería de software, ya desde sus inicios. Actualmente, cada vez son más los contextos empresariales donde el desarrollo de software es visto como una actividad de re-desarrollo. Estamos hablando de un entorno donde las empresas

venden sus productos para un uso masivo y permiten la adaptación de éstos a las necesidades cambiantes del mercado. Por ejemplo, sectores como el de telecomunicaciones, aplicaciones para el hogar, sector automovilístico, etc. donde se construyen productos iguales o muy similares una y otra vez (y donde la mayoría de los productos han sido ya desarrollados anteriormente).

Así, las LPS representan el siguiente escalón o vuelta de espiral. Las LPS se entienden como una de las formas más prometedoras para llevar a buen término un Plan de Reutilización, consiguiéndose así un incremento en la productividad, en la competitividad en el mercado y en la calidad de los productos de software finales [Bosch10]. Las LPS prometen convertirse en el paradigma dominante de desarrollo de software de este siglo. La flexibilidad de los productos es el nuevo "himno" en el mercado, y el paradigma de LPS proporciona sistemas adaptables a las necesidades concretas de cada cliente. La clave del éxito en este nuevo enfoque radica en explotar las características comunes compartidas por todos los productos dentro de la línea, para lograr beneficios económicos. [Clements02], [Easterman01], [Svahnberg00], [SEI01], [America00].

Vamos a utilizar la definición de LPS de Clements [Clements02] que define una línea de producto software como un "conjunto de sistemas software intensivo que comparten un conjunto de características comunes las cuales satisfacen las necesidades específicas de un segmento del mercado y son desarrolladas a partir de un conjunto de activos comunes de una forma preestablecida".

3.1.2 PROCESO DE DESARROLLO

Cuando se aplica el enfoque LPS, se está produciendo un cambio de un desarrollo orientado a un único producto software, a un desarrollo centrado en varios

productos que comparten unas características formando una familia. Como consecuencia, el proceso tradicional de desarrollo de software no sirve bajo este enfoque. El enfoque de LPS provoca una reestructuración del proceso de desarrollo; se pasa de una vista de "proyecto" a una vista de "dominio" [SEI01], [Steffen00], surgiendo dos niveles denominados (véase la *figura 4*):

- Ingeniería de Dominio o proceso de desarrollo para reutilización
- Ingeniería de Aplicación o proceso de desarrollo con reutilización.
-

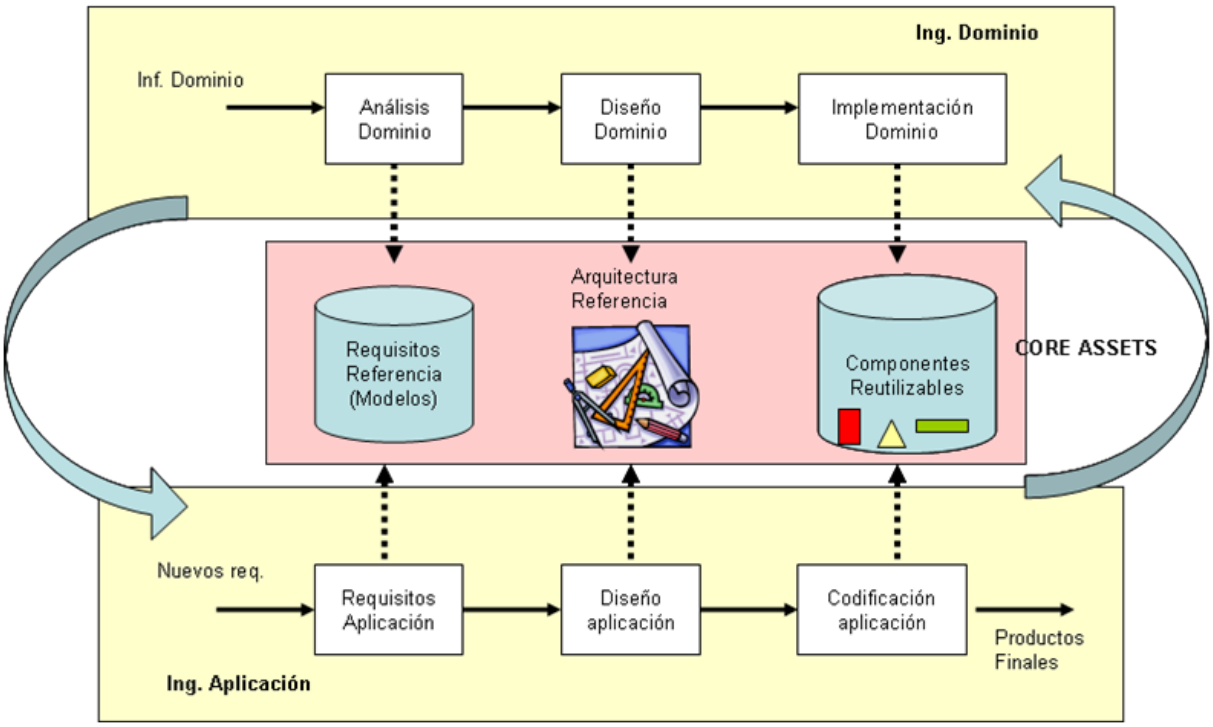


Figura 4: Proceso de desarrollo para el enfoque LPS.

La ingeniería de dominio o desarrollo para la reutilización, estudia cómo los productos miembros del dominio se parecen y diferencian entre sí. Esta fase se

centra en el desarrollo de elementos reutilizables (activos) que formarán la familia de productos.

La ingeniería de aplicación o desarrollo con reutilización se orienta hacia la construcción de productos individuales, pertenecientes a la familia de productos, reutilizando e integrando los elementos reutilizables (activos) existentes, producidos en la ingeniería de dominio.

Existe realimentación entre la ingeniería de dominio y la ingeniería de aplicación. De hecho, los productos desarrollados y los activos comunes de la LPS influyen los unos a los otros y evolucionan en paralelo durante su ciclo de vida. Los activos comunes son la base utilizada para producir nuevos productos, pero también es cierto que se definen y evolucionan según se desarrollan nuevos productos.

Este nuevo proceso de desarrollo origina además una nueva forma de trabajar, nuevos perfiles o roles dentro de la ingeniería de software. Así, en la ingeniería de dominio, surge la figura del "Ingeniero de Dominio". Esta figura se puede equiparar al responsable de producto en el desarrollo tradicional, pero ampliando el concepto de producto a dominio. El ingeniero de dominio es el responsable de la definición del dominio y de su estructuración o modularización en elementos comunes. Es el encargado de determinar los elementos comunes y variables del dominio, así como de la evolución de dichos elementos (elementos variables que con el tiempo pasan a ser comunes, nuevo elementos, etc.).

Mientras tanto, en el dominio de aplicación, los ingenieros de software realizarán las tareas, vamos a decir "tradicionales", del desarrollo de software: análisis, diseño, desarrollo y pruebas de cada producto desarrollado. La diferencia está en que deben hacerlo a partir de los elementos comunes identificados. Así, ante un nuevo producto a desarrollar, el ingeniero de software de aplicación, identificará los requisitos del cliente, y teniendo en cuenta los requisitos de la LPS, determinará sus necesidades reales, a partir de la determinación de esas necesidades se realiza el

diseño del producto, utilizando la arquitectura de referencia y se codifica utilizando los componentes existentes.

3.1.3 LPS. ESTADO DEL ARTE Y DE LA PRÁCTICA

A finales de los años noventa aparecieron los primeros trabajos sobre LPS. En ellos se describían las posibilidades de este nuevo enfoque de desarrollo de software [Jacobson97], [Bass98]. A partir de este momento los esfuerzos realizados en la definición de metodologías para el desarrollo de LPS han sido considerables: PuLSE [DeBaud98], KobrA [Atkinson00], COPA [America00], FAST [Weiss99], FORM [Kang98], FODA [Kang90], SREB + FODA [Griss98], SPLIT [Coriat00], Featured-Oriented PLE [Kang02], etc.

A partir del año 2000, el interés de las LPS crece sustancialmente. Esto se refleja tanto en las crecientes publicaciones sobre el tema [Bosch00], [Atkinson02], así como las conferencias y workshops dedicadas exclusivamente al área, tales como *Software Product Line Conference* (SPLC), *Component-Based Software Engineering* (CBSE), International *Conference on Component Deployment* (CD), *International Conference on Software Reuse* (ICSE), etc.

Desde el punto de vista de aplicación, el enfoque de LPS ha sido adoptado por organizaciones de distintos sectores. [Bosch00], [Clements02]. Stefen Thiel y Andreas Hein lo aplican en el sector de la automoción [Thiel02], Ari Jaaksi lo aplican en el desarrollo de navegadores para dispositivos móviles [Jaaksi02], Hewlett-Packard [Toft00], [Toft04] una de las mayores empresas en tecnologías de la información comenzó con la adopción del enfoque de LPS en 1997, Hitachi lo aplicó en el desarrollo de dispositivos médicos [Barlas96] y desarrolló un método para la evolución de las LPS [Yoshimura08], [Jenesen09], SystemForge [Bell07] adopta el

enfoque LPS para el desarrollo de diversas aplicaciones web como comercios electrónicos y gestión de contenidos.

Es destacable el trabajo realizado por el Software Engineering Institute (SEI), tanto en propuestas de adopción del enfoque LPS (Product Line Technical Probe PLTP) [Clements02], como en diversos estudios comparativos sobre la aplicación de este enfoque: [Northrop02], [Schmid02], [Linden02].

Existen multitud de líneas de investigación abiertas, pero profundizando en temas más concretos, quizás la más investigada sea la gestión de la variabilidad (la característica que diferencia a este modelo de otros) [Schmid04], [Bosch02], [Sinnema07], [Chen09], [Babar10].

En los últimos tiempos, está creciendo como corriente de investigación la integración de las LPS y el movimiento Ágil [Mohan10].

La base de esta área de conocimiento, es que si bien las LPS prometen y consiguen cubrir el mercado de forma personalizada y controlar costos, lo cierto es que por detrás está soportada por una ingeniería de dominio y de aplicación que en ocasiones resulta pesada, sobre todo desde el punto de vista de evolución y adaptación a cambios.

El movimiento ágil apoya la improvisación frente a los métodos de desarrollo tradicionales; se apoya en las personas más que en la documentación y promueve un desarrollo rápido en entornos no del todo conocidos. Integrar estos dos movimientos podría resultar muy beneficioso y ya hay bastantes investigaciones al respecto [Diaz11], [Freitas11], [Kakarontzas10], [Mohan10], [Hanssen08], [Faegri08], [Tian06], [Noor08].

3.2 INGENIERÍA DE REQUISITOS

En los siguientes apartados mostramos una visión completa de la fase de ingeniería de requisitos. Así se introducen los conceptos básicos utilizados en esta etapa inicial del desarrollo de software, se presentan las actividades que la constituyen y se muestran los procesos definidos para abordarla de una manera sistemática.

3.2.1 INTRODUCCIÓN A LA INGENIERÍA DE REQUISITOS_ CONCEPTOS BÁSICOS

Tradicionalmente, la ingeniería de requisitos (IR) se ha entendido como una parte poco tangible del ciclo de vida software, en la que se obtiene una especificación formal de unas ideas informales. Sin embargo, desde mediados de los años setenta esta disciplina ha cobrado una especial importancia, y actualmente se considera la etapa clave en el desarrollo de software. Fue en 1977 cuando los términos *ingeniería* y r*equisitos* se utilizaron juntos por primera vez en el desarrollo de SREM (Software Requirements Engineering Method) [Alford77]. El uso del término *ingeniería* indica la posibilidad de utilizar técnicas sistemáticas y repetibles, para alcanzar los objetivos propuestos [Sommerville97].

Existen diversas definiciones de IR que enfatizan aspectos diferentes. Así, para Aurum y Wohlin [Aurum03], la IR es un proceso de resolución de problemas complejos que involucra a clientes e implica decisiones. Según Ebert&Wieringa [Ebert05], la IR es la rama de la ingeniería de software relacionada con las propiedades esperadas y las limitaciones de los sistemas intensivos software, los objetivos a lograr en el entorno software y la concepción del entorno [Ebert05]. Para Lausen, es el proceso de identificar, analizar, documentar, validar y gestionar las propiedades del software [Lausen02]. Para Pohl es un proceso de construcción

de una especificación de requisitos en el que se avanza desde especificaciones iniciales, que no poseen las propiedades oportunas, hasta especificaciones finales completas, formales y acordadas entre todos los participantes [Pohl94], [Pohl97].

Aunque no existe una definición aceptada por toda la comunidad de la Ingeniería de software, en términos generales se puede decir que la ingeniería de requisitos (IR) es la etapa inicial del proceso de desarrollo de software, cuyo principal objetivo es descubrir y capturar las necesidades de los interesados (*stakeholders*, en inglés) para definir las características que debe satisfacer un determinado sistema software [IEEE98], [IEEE04], [Jiang07], [Maiden07], [Maiden04], [Solheim05], [Nuseibeh00a]. Estas características se registran en un documento formal denominado "Especificación de Requisitos del software o del sistema (ERS)".

El concepto esencial en la IR es el *Requisito*. El término "Requisito" ha sido definido de diferente forma por diferentes autores. Idealmente, los requisitos expresan las características QUE un sistema debe mostrar, en lugar de expresar CÓMO funciona el sistema. Es decir, los requisitos deben expresar el problema y no la solución [Kotonya98]. Sutcliffe indica que los requisitos implican encontrar lo que las personas esperan de un sistema y entender estos deseos en términos de diseño [Sutcliffe02]. Esta definición nos muestra como los requisitos transmiten el propósito del sistema y además dirigen el desarrollo del mismo. En la misma línea está la definición de Ferdinardi que define requisito como la necesidad o deseo a satisfacer por el producto o servicio [Ferdinardi02]. Otras definiciones explican lo que incluyen los requisitos, por ejemplo Sommerville define requisito como una declaración, en lenguaje natural y con diagramas, de los servicios esperados de un sistema y de las limitaciones bajo las cuales funcionará [Sommerville97]. Esta declaración se genera a partir de la información proporcionada por los clientes. Otro ejemplo de definición es la de Kotonya que dicen que los requisitos definen los servicios que el sistema debe proporcionar y establecen el funcionamiento del sistema [Kotonya98]. Otros autores siguen esta línea y definen requisito como la

propiedad que un sistema debería tener para resolver problemas del mundo real [SWEBOK04], [Sawyer01]. Una definición más extensa es la proporcionada por el estándar IEEE 610 [Machado05], [IEEE90] que ven un requisito como: a) una condición o capacidad que un usuario necesita para resolver un problema o lograr un objetivo. b) una condición o capacidad que debe tener un sistema o un componente de un sistema para satisfacer un contrato, una norma, especificación u otro documento formal. c) Una representación en forma de documento de una condición o capacidad como las expresadas en a) o b).

Como podemos observar no existe una única definición de *ingeniería de requisitos* ni sobre el propio concepto de *Requisito*. Sin embargo todas las definiciones integran la misma esencia:

- Resolución de problemas complejos
- Participación de múltiples actores
- Toma de decisiones

3.2.2 ACTIVIDADES DE LA INGENIERÍA DE REQUISITOS

El proceso de IR implica una serie de actividades, relacionadas e iterativas, mediante las cuales distintas clases de información fluyen, y se incrementa el conocimiento relativo a los requisitos (véase la *figura 5*).

Generalizando, podemos decir que el proceso IR está formado por las actividades de obtención, análisis, negociación, especificación-modelado, verificación y validación y gestión de requisitos [SWEBOK04], [Wieger00], [Faulk97].

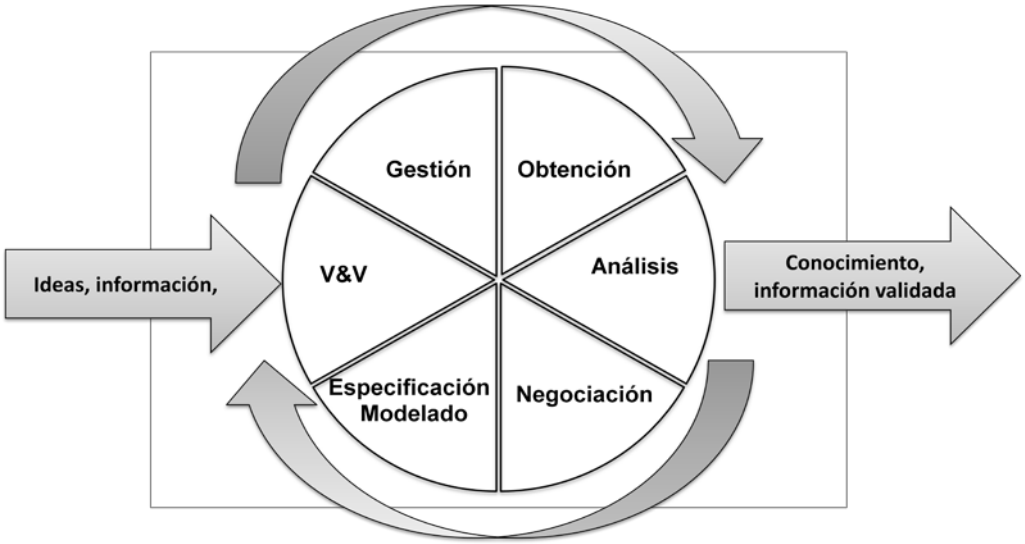


Figura 5: Proceso Iterativo de Ingeniería de requisitos

En los siguientes párrafos, explicamos estas actividades de forma independiente y desde un punto de vista genérico, aunque se sobreentiende que en la práctica están interrelacionadas, son iterativas y pueden ejecutarse a lo largo de todo el proceso de desarrollo.

Obtención de Requisitos

Es el proceso de identificación y descubrimiento de las necesidades de los stakeholders. En esta actividad, los ingenieros de requisitos consultan distintas fuentes de información como clientes, usuarios, expertos en el dominio, etc., es decir, stakeholders, para entender el dominio del problema y establecer los requisitos del sistema a desarrollar [Kotonya98], [Zowghi05], [Goguen93]. Aunque los stakeholders tengan una idea muy clara de lo que esperan del sistema les suele resultar difícil articular sus deseos, por lo que generalmente en esta fase se obtiene un conjunto de requisitos incompleto, expresados de forma vaga o no estructurada

[Sommerville97], [Kotonya98]. Por este motivo el ingeniero de requisitos tiene un trabajo complicado.

En la obtención de requisitos predominan los aspectos sociales sobre los técnicos. Es una actividad fundamentalmente "humana" [Bourque99].

Utilizando las palabras de [Zowghi05] la obtención de requisitos debe permitir la comunicación, priorización, negociación y colaboración con todos los stakeholders relevantes. Además debe proporcionar una base sólida para la aparición, descubrimiento y/o invención de las necesidades como parte de un proceso totalmente iterativo.

Análisis de Requisitos

Una vez descubiertos los requisitos, es necesario organizarlos de una manera coherente y eliminar conflictos e inconsistencias [Sommerville97], [Kotonya98].

Así el objetivo del Análisis es obtener un conjunto consistente y completo de requisitos, y para ello es necesario:

- Detectar los conflictos en los requisitos de los stakeholders; normalmente los requisitos se originan en distintas fuentes y normalmente presentan contradicciones o ambigüedades, debido a su naturaleza informal.
- Profundizar en el modelo del dominio del problema; generalmente se estructuran los requisitos y se refinan de manera que los stakeholders tengan un entendimiento completo de las especificaciones del sistema [Wiegers99]. También es muy común transformar un conjunto de requisitos informales en una representación semi-formal o formal, es decir en un modelo.
- Establecer las bases para el diseño.

Negociación de Requisitos

Esta actividad se suele realizar durante el análisis, pero debido a su importancia en el enfoque de desarrollo para líneas de producto software se presenta en un apartado diferente.

Es más que probable que los distintos stakeholders presenten distintas necesidades que sean contradictorias entre sí. En estos casos, es necesario negociar entre los distintos participantes hasta obtener una visión común de los requisitos. [Boehm94], [Kotonya98], [Sommerville97], [Sutcliffe02], [Grunbacher05]. De este modo se puede afirmar que el propósito más importante de la negociación es obtener un acuerdo común sobre un conjunto de requisitos [Sutcliffe02]. Según [Grunbacher05] otro resultado importante de la negociación de requisitos es el entendimiento sobre el por qué existe un desacuerdo entre stakeholders.

En esta actividad, al igual que pasaba en la actividad de obtención, predominan los aspectos sociales sobre los técnicos. Sin embargo, el proceso de negociación debería basarse en argumento lógicos (necesidades técnicas, organizativas, etc.) [Kotonya98].

Especificación/Modelado de Requisitos

El término Especificación de Requisitos tiene un doble significado. Por una parte se refiere a la actividad de especificar los requisitos de forma que sean entendidos por todos los stakeholders; y por otra es el nombre del documento resultado de esta actividad [Matulevicius04a].

En la actividad de especificación de requisitos, se documentan los requisitos solicitados, analizados y negociados, utilizando alguna notación que todos los participantes entiendan. Según [Pressman01b] una documentación de requisitos

debe contener una descripción detallada desde un punto de vista funcional, una representación del comportamiento del sistema, y las limitaciones desde un aspecto de diseño y de comportamiento. Hay autores que dicen que resulta más efectivo utilizar métodos formales en la especificación de requisitos, ya que el lenguaje natural se presta a la ambigüedad y las malas interpretaciones [Sutcliffe02], [Kotonya98]. Sin embargo, el lenguaje natural es el principal medio de comunicación por lo tanto resulta simple y entendible por todos los stakeholders. Además, es útil para la validación de los requisitos y no es específico de dominio en un nivel de abstracción determinado. También, resulta más flexible que un lenguaje formal [Natt05].

La especificación de requisitos permite utilizar tantas notaciones como sea necesario para que todos los participantes las entiendan. [Pohl97], [Sawyer99a], [Davis95]. Incluso hay autores proponen que los requisitos de los clientes y los requisitos de los desarrolladores se especifiquen en documentos separados [Bourque99].

Los modelos describen un aspecto particular del sistema, complementando las descripciones de los requisitos en lenguaje natural. Se les puede considerar como una especificación detallada de aspectos concretos del sistema.

Verificación y Validación de Requisitos (V&V)

Por un lado se debe comprobar que los requisitos obtenidos, una vez analizados y resueltos los posibles conflictos, corresponden realmente a las necesidades de los stakeholders [Sutcliffe02]. Esta actividad debe ser realizada por los propios stakeholders, con la ayuda si es necesario de los ingenieros de requisitos. Por otro lado la V&V se centra en el propio documento de especificación “per se” y tiene

como objetivo comprobar que el documento que refleja los requisitos es consistente, completo y veraz Kotonya98], [Bray02].

Gestión de Requisitos

El objetivo de esta actividad es mantener los requisitos gestionando los cambios producidos en los mismos con el objeto de que representen el sistema en desarrollo o que ya se ha desarrollado [Leffingell03]. La gestión de requisitos es la estructuración y administración de la información creada durante las actividades de obtención, análisis, negociación, especificación y V&V del proceso de IR y a lo largo del ciclo de vida del producto [Hoffmann04]. Ambas definiciones muestran cómo la actividad de gestión se mantiene activa durante todo el proceso de ingeniería de requisitos, durante todo el proceso de desarrollo, e incluso durante todo el ciclo de vida del producto.

3.2.3 PROCESO DE INGENIERÍA DE REQUISITOS

En el apartado anterior hemos mostrado las actividades genéricas que componen la fase de IR, sin embargo, no existe un consenso en la secuenciación y dependencia de estas actividades y por ese motivo existen diferentes modelos de procesos para la ingeniería de requisitos, donde se combina de diversas maneras la realización de las actividades. Así, por ejemplo, Davis [Davis03] estructura el proceso en dos fases (Análisis y Especificación), mientras que [Locopoulos95] lo hace en tres fases (Obtención, Modelización y Validación).

Se pueden identificar casi tanto modelos de proceso de ingeniería de requisitos, como contextos de desarrollo y organizacionales donde aplicarlos [Sutcliffe02],

[Bray02], [Eriksson07], [Kotonya98], [Sawyer05]. Aun así, hay que destacar que todos ellos combinan las actividades mencionadas en el apartado anterior, y todos ellos parten de la base que:

- La ingeniería de requisitos es un proceso iterativo: es básicamente un proceso de descubrimiento y comunicación, y no puede realizarse de forma lineal. Las actividades que conforman este proceso pueden repetirse múltiples veces mientras se adquiere el conocimiento sobre el sistema deseado [GarciaDuque09].
- Los límites de las actividades son difíciles de establecer: aunque disponer de un modelo de procesos es siempre beneficioso, hay que tener en cuenta que los límites de las actividades no son claros en ingeniería de requisitos.
- Los productos a obtener como resultado del proceso de ingeniería de requisitos no están claramente definidos: la mayoría de los modelos no definen claramente los productos que se deben obtener, exceptuando, claro, un documento de especificaciones del sistema a desarrollar. Casi todos los modelos excluyen otro tipo de documentación como podría ser una descripción de los stakeholders participantes, unas reglas explícitas para la negociación de requisitos, así como el resultado de su aplicación, etc.

3.3 LA INGENIERÍA DE REQUISITOS PARA LÍNEAS DE PRODUCTO SOFTWARE

En este punto podemos afirmar que la ingeniería de requisitos bajo el enfoque de desarrollo orientado a mercado implica unos cambios importantes respecto a la ingeniería de requisitos bajo un enfoque tradicional o a medida. El interés de este área viene demostrado por el tema de la decimocuarta conferencia internacional *IEEE International Requirements Engineering conference* "Entendiendo los deseos y necesidades de los stakeholders", y el número de Marzo/abril del 2007 de la revista

IEEE Software titulado "Stakeholders en la ingeniería de requisitos". Este interés queda demostrado también por los diversos trabajos de investigación realizados, centrados todos ellos en mostrar las diferencias en las fases iniciales de ambos enfoques de desarrollo. Uno de los primeros trabajos en los que se analizaron estas diferencias fue el de Lubars [Lubars93], pero después han sido más los autores en explicar las características del desarrollo orientado a mercado [Lubars93], [Potts95], [Sawyer00].En la *tabla 5* se recogen estas diferencias. Esta tabla está basada en los trabajos realizados por autores como [Bowman96], [Carlshamre02], [Kamsties98], [Keil95], [Lubars93], [Novorita96], [Potts95], [Yeh92] y [Robertson99].

Existen otras investigaciones, que van más allá del mero análisis de diferencias y proponen soluciones a problemas identificados. Así, [Kuusela05], [Kuusela00] propone la organización en jerarquías de los requisitos como soporte para las familias de productos, [McPhee02] y [Kuloor02] analizan la aplicabilidad de técnicas de IR tradicionales en el contexto de líneas de producto software, [Chastek01] trabaja sobre todo la modelización (basada en casos de uso y modelos de características) proponiendo diferentes estrategias para su realización, [Laguna05] proponen la aplicación de Objetivos (Goals) y MDA (Model Driven Architecture) para la gestión y representación de la variabilidad, y [Halmans03] propone el uso de casos de uso como medio de comunicación con los clientes.

3.4 RESUMEN

Como se afirma en el apartado *3.3 La ingeniería de requisitos para líneas de producto software*, la ingeniería de requisitos bajo el enfoque de desarrollo orientado a mercado implica cambios notables respecto a la ingeniería de requisitos bajo un enfoque tradicional o a medida (*véase tabla 5*) [Ereño08].

Aspectos	Desarrollo a medida	Desarrollo LPS
Objetivo principal	Cumplimiento de la especificación de requisitos	Tiempo de lanzamiento al mercado
Medición del éxito	Cumplimiento de las especificaciones. Satisfacción. Aceptación	Ventas. Porción de mercado alcanzada
Ciclo de vida	Una versión del producto y luego mantenimiento del mismo	Varias versiones del producto. Tantas como sea necesario mientras exista demanda en el mercado
Proceso de desarrollo	En cascada o similar	Proceso desdoblado: Ingeniería de Dominio e Ingeniería de Aplicación
Stakeholders principales	Organización del cliente	Organización de desarrollo
Conocimiento stakeholder	Usuarios conocidos o al menos identificables	Usuarios inicialmente desconocidos y difíciles de identificar
Comunicación stakeholder	La dificultad radica en identificar con precisión los deseos de ese usuario	La dificultad radica en identificar a los usuarios y en conseguir información sobre sus deseos
Distancia stakeholders	Normalmente pequeña	Normalmente grande
Stakeholders - producto	Relación a corto plazo (hasta que termina el proyecto)	Relación a largo plazo (mantenimiento y evolución del producto)
Cliente	La persona que encarga el software o la que negocia el contrato	Sectores del mercado. Producto a medida
Actividades IR	Obtención, modelado, validación y resolución de conflictos	Identificación gradual de requisitos, priorización, estimación de costes, planificación de versiones
Especificación de requisitos	Utilizada como contrato entre el cliente y el desarrollador	Existe en pocas ocasiones y si existe es poco formal
Métodos IR	Existen aunque en la práctica cuesta utilizarlos	Apenas existen o son muy generales

Tabla5: Comparación de las características de desarrollo del software a medida y del software orientado a mercado

Aunque son muchas las diferencias detectadas, podemos observar que la mayoría de ellas están relacionadas con la participación de los stakeholders [Ballejo11], [Solaimani12]. Hay un factor que no cambia y es que, la integración de stakeholders en el proceso de desarrollo (y más concretamente en las fases iniciales) es vital, juegan un papel importante y decisivo en la ingeniería de requisitos. En el enfoque de desarrollo orientado a mercado, esta tarea es mucho más complicada.

El problema básico es la comunicación. La IR es una actividad que por naturaleza conlleva grandes dosis de comunicación y colaboración. Esta característica se potencia bajo el enfoque LPS. Mientras en el desarrollo a medida, el cliente es una persona o grupo de personas identificadas, bajo el enfoque SPL el cliente es una entidad abstracta. El desarrollo se dirige a mercados globales, los stakeholders (incluyendo clientes, usuarios y desarrolladores) pueden ser numerosos y estar físicamente distribuidos.

Es clara la necesidad de un método que integre a los stakeholders implicados y proporcione un soporte a la actividad de ingeniería de requisitos [GarciaDuque09], [Letier02], [Liu02]. Siendo esta una necesidad clara, los trabajo de investigación existentes en el contexto de las líneas de producto software, no abordan las fases iniciales del desarrollo o lo hacen sin profundizar y manteniendo un enfoque dirigido al desarrollador. Ninguno de estos métodos se centra en la fase inicial de requisitos, ni la aborda formalmente desde el punto de vista del cliente [Ereño07].

Podemos concluir, por lo tanto, que existe un problema a resolver en el contexto planteado. En el siguiente capítulo (*Capítulo 4 – Ingeniería de requisitos orientada a perspectivas*) se analiza la posible aplicación de una técnica tradicional de ingeniería de requisitos, como es el uso de Perspectivas, como solución al problema planteado.

CAPÍTULO 4- INGENIERÍA DE REQUISITOS ORIENTADA A PERSPECTIVAS – IROP

El proceso de la IR implica la captura, análisis y resolución de múltiples ideas, perspectivas y relaciones con diferente grado de abstracción [Kotonya98]. Los métodos de IR tradicionales, centrados en el desarrollo y no en el usuario, y basados en esquemas de estructuración rígidos no proporcionan el marco necesario para extraer y estructurar la diversidad de conocimiento requerida. El enfoque de ingeniería de requisitos orientada a perspectivas (IROP) surge para cubrir esta deficiencia. En este capítulo profundizamos sobre la filosofía de la ingeniería de requisitos basada en perspectivas y analizamos la aplicabilidad de esta filosofía como solución al problema identificado en el capítulo 3.

4.1 INTRODUCCIÓN

La ingeniería de requisitos orientada a perspectivas (IROP) considera el origen de la información relacionada con el problema disperso entre diferentes agentes (stakeholders), y cada uno/a de estos *agentes* tiene diferentes *visiones* (todas ellas igualmente válidas pero incompletas) del problema. Estas visiones surgen como consecuencia de las diferentes responsabilidades, roles, objetivos o interpretaciones de las fuentes de información. A la combinación de "*stakeholders* " más su "*punto de vista"* u opinión sobre el problema se le denomina "*vista"* o "perspectiva" [Finkelstein96]. Así, el enfoque IROP reconoce que toda la información de un sistema no se puede descubrir considerando a dicho sistema

desde una única perspectiva; al contrario, es necesario recoger y organizar la información desde perspectivas diferentes [Easterbrook05]. De este modo, la información completa sobre el sistema se obtiene gracias a la integración de las vistas o perspectivas (véase la *figura 6*).

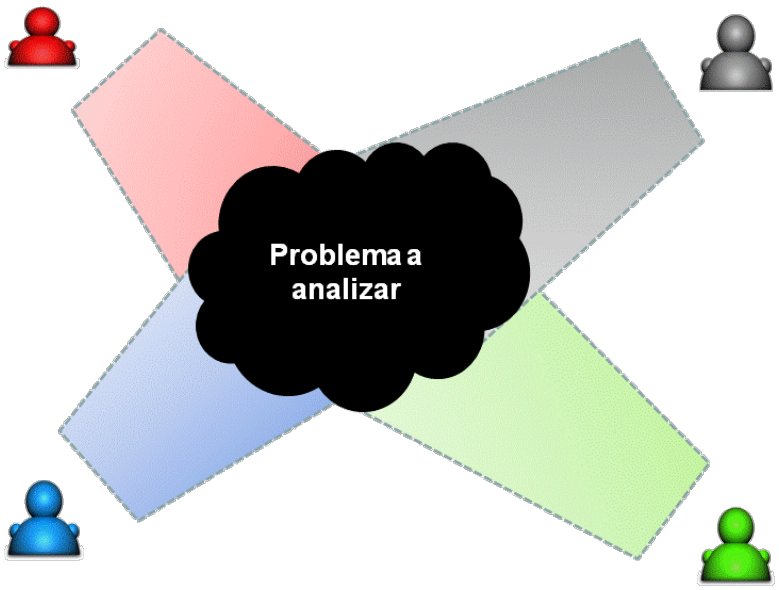


Figura 6: Vistas\Perspectivas sobre un sistema.

Los modelos basados en puntos de vista consideran a éstos (a los puntos de vista) como elementos de primera clase en su concepción del sistema.

El enfoque basado en puntos de vista ayuda a organizar la información atendiendo a las distintas perspectivas y fuentes del sistema y también ayuda a poner de manifiesto, rápidamente, la existencia de conflictos entre los agentes. Esto implica la realización de un proceso de resolución de conflictos que cada uno de los métodos de este grupo resuelve de distinto modo.

En los siguientes apartados de este capítulo se explica el origen del enfoque IROP basado en la evolución de la participación en el proceso de desarrollo de software. Así se explicará el origen del término *stakeholders* que hemos venido utilizando desde el comienzo de esta investigación. Posteriormente, profundizamos en los dos principios básicos de este enfoque: la identificación de perspectivas y el tratamiento de inconsistencias. A continuación mostramos los métodos IROP más relevantes desarrollados y el capítulo finaliza planteando la solución al problema a resolver mediante el uso de un método IROP.

4.2 ORIGEN DEL ENFOQUE IROP

La filosofía que subyace al enfoque IROP no es nueva dentro de la literatura de ingeniería de requisitos; de hecho, se puede observar una evolución en la participación cada vez mayor de diferentes roles en el desarrollo de productos de software. Si miramos hacia atrás en la historia de la ingeniería de software, podemos apreciar cómo esta participación va creciendo, pero no de forma explícita ni procedimentada. El enfoque de IROP es el primero en proponer una integración explícita no sólo de los roles involucrados, sino también del punto de vista de cada rol. Así surge el concepto de *Perspectiva* o *Vista*.

En los inicios del desarrollo de software, cuando los productos se realizaban "a medida", la única participación externa permitida era la del cliente/usuario; en aquellos momentos no se diferenciaba entre cliente y usuario. Analizando la literatura sacamos la conclusión que no existía una clara definición sobre la participación del usuario en el desarrollo de software. Se han utilizado conceptos como "participación del usuario", "foco en el usuario" "consultando al usuario" y "contactando con el usuario" para explicar la forma de colaboración con el usuario [Kujala05].

Con el correr de los tiempos y viendo los problemas de no involucrar al usuario, se detecta la necesidad de la participación del mismo [Kujala03]. Analizando la literatura son muchas las fuentes que identifican las ventajas en la participación del usuario en el proceso de desarrollo de software: [Gallivan03] explica cómo la participación del usuario mejora el proceso de IR, consigue aumentar las compras, y mantiene a los usuarios informados del progreso del proyecto, permitiendo así incrementar los niveles de satisfacción del usuario, la calidad del sistema y el uso del sistema. En [Khan89] afirman que aquellos usuarios involucrados en las primeras etapas del proyecto son más proclives a la aceptación del sistema, [Keil95] indica que la participación del usuario se transforma en ideas para nuevos productos y en mejoras del producto existente.

Según evolucionan mercado y tecnología, la participación de otros roles se va transformando en esencial. Con el objetivo de conseguir un producto de calidad se introducen en el proceso los roles de *revisores*, *inspectores, testadores* o resumiendo, el rol de *calidad*. Con el objetivo de soportar la evolución y mantenimiento de los productos, surge el rol de *mantenimiento*. Aparecen, o al menos se distinguen, los roles de *analista, diseñador, arquitecto, jefe de producto, jefe de proceso*, etc. Todos estos roles trabajan conjuntamente en el desarrollo del producto. Sin embargo, también se detecta la necesidad de involucrar en el desarrollo a todas aquellas personas que de alguna forma se ven afectadas por el resultado del mismo: clientes, usuarios finales, etc. Para poder manejar todos estos roles participantes surge el concepto de *stakeholder*.

El término *stakeholders* proviene de la frase "*The holder of the best in a game*", que informalmente se puede explicar como toda persona o grupo de personas que tienen un interés o parte en un negocio o empresa.

El término stakeholder tiene su origen en los ámbitos de gestión y estrategia empresarial [Freeman84] donde se define como cualquier grupo o individuo que

puede afectar o verse afectado por el logro de los objetivos de la empresa. Sin embargo este término ha sido adoptado también por otros ámbitos, así en el área de los Sistemas de Información, por ejemplo, stakeholders son los participantes en el proceso de desarrollo, incluyendo individuos, grupos u organizaciones cuyas acciones pueden influenciar o ser influenciadas por el desarrollo y uso del sistema de forma directa o indirecta [Pouloudi97].

En el área de la ingeniería de software, se pueden encontrar diversas definiciones para el término stakeholders: en [Conger94] se define stakeholders como aquellas personas y organizaciones afectadas por la aplicación; para [Kotonya98], los stakeholders son personas u organizaciones que se verán afectadas por el sistema y que tienen influencia directa o indirecta en los requisitos del sistema; [Cotterell95] los define como aquellas personas que tienen un interés en el proyecto; [Dix93] los define como cualquier persona cuyo trabajo se vea alterado, que proporcione u obtenga información del mismo, o cuyo poder o influencia en la organizaron pueda aumentar o disminuir por el sistema; [Glinz07] define stakeholder como una persona u organización que tiene influencia en los requisitos de un sistema o que es impactado por el sistema; en el trabajo de [Alex07] afirman que los stakeholders de un proyecto no incluyen solo a los usuarios resultantes de la creación de un sistema, sino también a aquellos que originaron el proyecto, aquellos cuyo trabajo o actividades cambiaron por la creación del sistema, aquellos que lo construyeron y aquellos que pueden influir en el éxito del proyecto; los autores [Poloudi97], [Kaler03], [Woolridge07] proponen una definición más amplia donde incluyen como stakeholder a cualquiera que pueda influenciar o ser influenciado por el proyecto o por el sistema resultante. En la ingeniería de requisitos el término stakeholder apareció en los años 90 cuando se consideró que los términos cliente y usuario eran demasiado específicos. El glosario estándar de ingeniería de software de 1990 [IEEE90] no incluye todavía el término stakeholder. La primera aparición de este

término de la que se tiene constancia es en 1993 en un artículo de Linda Macaulay [Macauly93].

Tras analizar todas estas definiciones podemos afirmar que en el marco de la ingeniería de software, por stakeholder se entiende cualquier persona afectada por el desarrollo de un producto software, que se beneficia de la existencia del mismo.

4.3 PRINCIPIOS DEL ENFOQUE IROP

El enfoque IROP y todos los métodos desarrollados dentro de su marco, encarnan una serie de características propias de la ingeniería de software más alguna nueva. En esta apartado vamos a desarrollar dos principios básicos de este enfoque, como son: la identificación de perspectivas y el tratamiento de inconsistencias.

4.3.1. IDENTIFICACIÓN DE PERSPECTIVAS

La actividad fundamental del enfoque IROP que precede a cualquier otra actividad en la fase de IR es la identificación de perspectivas. Lo primero es determinar quiénes son los stakeholders y cuál es su importancia respecto a la definición del producto [Glinz07]. La identificación de los stakeholders es necesaria para asegurar una especificación de requisitos completa y correcta [Woolridge07], [Pi12].

Esta actividad esencial no es fácil de llevar a cabo. En caso de encontrar muchas perspectivas, resultará difícil gestionar toda la información que generan y priorizar requisitos. Por ello, se trata de seleccionar solo las perspectivas críticas en el análisis. El ingeniero de requisitos debe realizar un balance entre la mayor

cobertura ofrecida por un número mayor de perspectivas, las dificultades de gestión de la información, y el costo asociado.

Cualquiera de los métodos desarrollados bajo este enfoque (y que se comentarán en detalle en el siguiente apartado) aborda esta actividad formalmente, a diferencia de los procesos ad-hoc anteriores. Analizando la literatura relacionada, la gran mayoría de las propuestas existentes para la realización de esta tarea, se basan en la experiencia y en el sentido común para su realización. Existen trabajos, en los cuales se sugieren los stakeholders "típicos", y proponen elegir entre ellos. Por ejemplo [Cotterell95] sugiere que los stakeholders pueden pertenecer a tres categorías, internos al equipo de proyecto, externos al equipo de proyecto pero internos a la organización, y externos al equipo de proyecto y a la organización. En [Newman95] dividen a los stakeholders en dos grupos, los que utilizarán el sistema y los que están involucrados en su desarrollo. Esta última clasificación es la misma que la propuesta por [Pouloudi97a] para el desarrollo de sistemas basados en el conocimiento. En [Sharp99] proponen un enfoque de identificación de stakeholders basado en redes donde se comienza con un grupo base donde se van incluyendo proveedores, clientes y grupos cercanos. Como limitación a este enfoque se puede decir que se centran exclusivamente en el dominio del proyecto ignorando el dominio del problema. Cabe destacar el trabajo de Glinz, que proporciona un conjunto de directrices para acertar en la elección de los stakeholders relevantes [Glinz07], así, estos roles deben cumplir alguna de las siguientes características:

- Tener un interés en el sistema: lo usan actualmente o están directamente relacionadas con procesos que el sistema a definir cambiará.
- Gestionar, operar o mantener el sistema después de su desarrollo.
- Estar involucrados en el desarrollo del sistema como arquitecto, desarrollador, testador, ingeniero de calidad o jefe de proyecto.
- Ser responsables del negocio o proceso que el sistema soporta o automatiza.

- Tener intereses financieros: han solicitado el sistema, lo han pagado o son responsables de su venta.
- Limitar el sistema de alguna manera.
- Estar afectados negativamente por el sistema: stakeholders negativos.

La identificación de perspectivas sigue resultando un área de investigación de interés hoy en día. Así lo demuestran trabajos como el de [Ballejos11] donde proponen una forma de modelar stakeholders para desarrollos a medida. Esto incluye la identificación de stakeholders y su priorización.

4.3.2 TRATAMIENTO DE INCONSISTENCIAS

Una característica clave del enfoque IROP es la tolerancia de inconsistencias [Easterbrook96], [Finkelstein94]. En este enfoque se permite la construcción y modificación de muchos modelos parciales del sistema, que se superponen entre ellos, sin necesidad de eliminar la inconsistencia de forma inmediata. Esto es un cambio radical frente a los enfoques tradicionales de ingeniería de requisitos: se acepta este grado de inconsistencia, fundamentalmente porque resulta natural. Es inevitable, que extrayendo información de diversas fuentes se produzcan solapamientos. Los solapamientos son previos a las inconsistencias. Sin embargo las inconsistencias se ven como una posible fuente de alternativas que promueve una forma de pensar innovadora.

Existen muchos trabajos que reconocen la necesidad de un enfoque más tolerante hacia la inconsistencia [Balzer91], [Gabbay91], [Narayanaswamy92], [Schwanke88]. Estos trabajos permiten el desarrollo ante la presencia de inconsistencias y ofrecen diferentes formas de analizar y resolver estas inconsistencias. Un paso más es el

trabajo de [Easterbrook96a] que además afirma que de las inconsistencias se puede aprender.

Se han propuesto diversos marcos que proporcionan un soporte explícito para la identificación, seguimiento y resolución de inconsistencias entre perspectivas [Easterbrook96], [Grundy98], [Robinson99], [Easterbrook91], [Nuseibeh97]. No existe un consenso sobre el momento en el cual eliminar la inconsistencia, por ejemplo [Lamsweerde98] se centra en resolver la inconsistencia en etapas muy tempranas mediante la resolución de divergencias entre los objetivos de los stakeholders, mientras que [Nuseibeh00a] argumenta que ciertas inconsistencias no se resolverán nunca.

Los trabajos más recientes se han centrado en la gestión de la inconsistencia entre perspectivas y han propuesto diversos esquemas de representación para capturar y gestionar las relaciones de consistencia en lenguajes de modelado. Por ejemplo, el trabajo de [Nentwich03] que propone la lógica de primer orden para chequear documentos XML, un enfoque de reglas para chequear modelos UML [Liu02], y una técnica para notaciones gráficas [Sabetzade03]. Otros trabajo han profundizado en técnicas formales de razonamiento que toleran la inconsistencia [Hunter98], [Easterbrook01].

4.4 LAS PERSPECTIVAS EN LA INGENIERÍA DEL SOFTWARE

Son numerosos los intentos de resolver el problema de las múltiples perspectivas en el ámbito de la ingeniería de requisitos. Como consecuencia han surgido diversos métodos con distintas propuestas de solución. El objetivo de todos los métodos IROP definidos es conseguir un equilibrio entre la aportación de cada perspectiva y la creación de un sistema consistente [Kotonya98]. Aunque los métodos son muy

diferentes entre sí, todos ellos proporcionan una manera de representar y gestionar las diversas perspectivas que aparecen durante el desarrollo de un sistema y también proponen marcos y/o técnicas para la integración de dichas perspectivas.

En este aparado se presentan los trabajos más importantes desarrollados en el ámbito IROP.

4.4.1 MÉTODOS INICIALES

SADT (Structured Analysis and Design technique) es un técnica de análisis y diseño que en su época fue un paso más allá y sugirió que el análisis debía realizarse desde diferentes puntos de vista. El método SADT fue desarrollado a finales de los 70 por Ross [Ross77] para el tratamiento de sistemas complejos mediante la construcción de modelos. El método recurre a la semántica gráfica para ayudar a estructurar y precisar la semántica del lenguaje natural que se emplea al construir un "modelo". La estructura del modelo obedece a los principios de descomposición de complejidad en árboles con configuración jerárquica-arborescente: aparece tanto mayor detalle cuanto más se avanza en el despliegue de los niveles jerárquicos del modelo.

SADT es el primer método que toma en consideración la existencia de perspectivas, sin embargo, no las utiliza de forma explícita. No existe ninguna etapa dentro del método donde se identifiquen, definan y estructuren las perspectivas. Hace uso de las perspectivas de forma intuitiva como si fuera una extensión de la técnica de modelado. Considera a las perspectivas como las fuentes, los orígenes de los datos. Tampoco trata las inconsistencias entre las diferentes perspectivas.

Los métodos estructurados como JSD [Jackson83] o las diferentes corrientes de análisis orientado a objetos [Rumbaugh91], [Alford77], [Jacobson93], [Booch94], ya

incorporan una noción implícita del concepto de perspectiva. Estos métodos sugieren que se pueden desarrollar diferentes modelos del sistema, donde cada uno de ellos proporciona una vista del sistema.

4.4.2 CORE

CORE (Controlled Requirements Expression) fue desarrollado a finales de los años setenta [Mullery79]. Este método ha sido ampliamente utilizado en la industria aeroespacial europea, en proyectos reseñables como *Experimental Aircraft Programme (EAP)* a mediados de los 80 en el que se utilizó para la definición del sistema software y más recientemente el European Fighter Aircraft (EFA) donde fue seleccionado como el método de análisis de requisitos a utilizar.

CORE promueve la participación del usuario. De hecho es uno de los primeros métodos donde se indica de forma explícita la intervención del usuario. En la filosofía CORE se identifican dos roles principales: por un lado el rol usuario, al cual considera imprescindible para proporcionar información, y por otro lado el rol analista, que será el encargado de traducir la información del usuario al equipo de desarrollo.

Al igual que SADT, CORE se basa en la descomposición funcional y está soportado por una notación basada en diagramas.

CORE utiliza las perspectivas explícitamente. Este método define dos niveles de perspectivas. El primer nivel, que comprende todas las entidades que interactúan con o se ven afectadas por el sistema en cuestión. En este nivel CORE proporciona unas guías para la identificación de perspectivas funcionales y no funcionales. En el segundo nivel diferencia entre “Defining Viewpoints” y “Bounding Viewpoints”,

donde los primeros son subprocesos del sistema y los segundos entidades externas que intercambian información o interactúan con el sistema.

CORE utiliza las perspectivas para estructurar la especificación de requisitos. Para este método, las perspectivas representan los diferentes componentes del sistema y su entorno, pudiendo así ser perspectivas de la organización, humanas, software o hardware. CORE no permite que las perspectivas se solapen entre ellas.

4.4.3 VOSE

VOSE (Viewpoint-Oriented System Engineering) fue desarrollado en el Imperial College de Londres a principios de los 90 [Easterbrook96], [Nuseibeh96], [Nuseibeh94]. Es un marco para integrar métodos de desarrollo, cuyo objetivo es abarcar todo el ciclo de desarrollo de un sistema, como su propio nombre indica.

VOSE proporciona una infraestructura para capturar y organizar el conocimiento durante el desarrollo de software [Nuseibeh03]. La filosofía que subyace bajo este marco es el hecho de que el desarrollo de software implica la participación de muchos expertos en las diferentes actividades del desarrollo. Cada participante puede tener responsabilidades y visiones que pueden cambiar durante el desarrollo. Estos métodos emplean diferentes notaciones y estrategias de desarrollo que deben integrarse para producir las especificaciones del sistema. VOSE proporciona el marco para realizarlo. Así, admite la inevitable existencia de múltiples puntos de vista singulares inconsistentes entre sí, promueve la separación de dichos puntos de vista y alienta la especificación no centralizada, además de proporcionar soporte para la integración y composición final de todos los puntos de vista.

VOSE utiliza la noción de perspectivas para hacer particiones y distribuir las actividades y el conocimiento de los participantes. Las perspectivas aglutinan el rol y la responsabilidad de un participante en una etapa determinada del proceso de desarrollo de software. Los autores definen una perspectiva como *"un objeto que encapsula conocimiento parcial acerca no sólo del (sub) dominio, sino también del proceso de desarrollo"* [Nuseibeh96].

Las perspectivas se representan mediante una plantilla donde se aglutina toda la información de la misma.

La fortaleza de este método es que, como marco de trabajo, sirve de guía para orientar un desarrollo basado en la multiplicidad de puntos de vista. Este modelo es realmente flexible y se puede utilizar como base para automatizar parte de la IR. Sin embargo, no proporciona guías concretas para la elección de perspectivas, elección de métodos, elaboración de reglas de integración; se tratan los temas de discrepancias desde un punto de vista genérico.

4.4.4 VORD

VORD (Viewpoint-Oriented Requirements Definition Method) es un método de ingeniería de requisitos que cubre este proceso desde el descubrimiento inicial de requisitos hasta el modelado del sistema. Fue desarrollado por Gerald Kotonya e Ian Sommerville [Kotonya 96].

El método VORD en sus inicios [Kotonya92] consideraba las perspectivas como receptores de servicios de un sistema y proveedores de datos e información de control. El modelo era similar a un modelo cliente-servidor, donde las perspectivas eran los clientes, el sistema desarrollaba servicios para los clientes (perspectivas) y las perspectivas pasaban información de control y parámetros al sistema. Las

perspectivas se podían mapear con clases de usuarios finales de un sistema o con otros sistemas que interactúan con él. Este enfoque tan centrado en el usuario final tenía el peligro de centrarse solo en tareas del usuario y dejar de lado los aspectos organizacionales, de manera que se obtuvieran unos requisitos incompletos. Por ese motivo la noción de perspectiva de este modelo evolucionó y se extendió [Kotonya96] distinguiendo dos clases de perspectivas: directas e indirectas.

Las perspectivas directas forman el corazón del modelo. Corresponden directamente con clientes que reciben servicios del sistema y pasan información de control y datos al sistema. Las perspectivas directas son tanto operadores/usuarios del sistema como otros subsistemas que interactúan con el sistema en análisis.

Las perspectivas indirectas tienen "interés" en alguno o todos los servicios que proporciona el sistema pero no interactúan directamente con él. Estas perspectivas pueden generar requisitos que limiten los servicios desarrollados para las perspectivas directas. Las perspectivas indirectas pueden ser: "perspectivas de ingeniería" (aquellas relacionadas con el diseño o implementación del sistema); "perspectivas organizacionales (aquellas relacionadas con la influencia del sistema en la organización); "perspectivas externas" (aquellas relacionadas con la influencia del sistema en el entorno), etc.

El método VORD consta de tres pasos iterativos:

- Paso 1: Identificación y estructuración de las perspectivas. El objetivo es identificar las perspectivas y los servicios que reciben del sistema. Para ello, se parte de una serie de perspectivas abstractas generales, y se intenta definir una jerarquía de clases de perspectivas basadas en las necesidades y en el dominio de aplicación del sistema concreto que se está desarrollando.
- Paso 2: Documentación de las perspectivas. VORD propone una estructura de información para las perspectivas soportada mediante un conjunto de plantillas.

- Paso 3: Análisis, especificación y validación de requisitos de las perspectivas. El propósito del análisis es establecer que los requisitos de cada perspectiva son correctos y completos. Esta actividad se realiza en dos pasos: el chequeo de la documentación, para asegurar que es consistente y no se ha omitido ninguna sección, y el análisis de conflictos, en donde se analizan todas las limitaciones de un servicio y confrontan los requisitos de todas las perspectivas.

VORD incluye un esquema de pesos simple para representar tres factores: la importancia, los recursos y el riesgo. Así a cada requisito se le asigna un peso en una escala de tres valores: alto, medio o bajo.

En cuanto a la especificación de requisitos, VORD permite especificar los requisitos utilizando diferentes notaciones. Consideran esto un punto fuerte ya que ayuda a la comunicación entre el ingeniero de requisitos y el usuario potencial. Afirma además que esto resulta necesario al no existir una notación universal que permita articular de forma adecuada todas las necesidades del sistema. En este punto, este método se parece al método VOSE (Apartado 4.4.3).

4.4.5 PREVIEW

El método Preview, desarrollado en el Departamento de Informática de la Universidad de Lancaster, surgió dentro del desarrollo del proyecto REAIMS [Sommerville97]. Sus autores son Ian Sommerville y Steve Viller.

PREview sigue un proceso de IR que va desde la actividad de identificación de perspectivas hasta la definición de requisitos, pasando por el análisis de inconsistencias.

Proponen dos clases de perspectivas:

- Perspectivas asociadas con "stakeholders del sistema: en esta categoría entrarían usuarios finales del sistema, gestores de las organizaciones donde se instalará el sistema, personal y sistemas dentro del organización donde se instalará el sistema, entidades externas con alguna clase de interés en el sistema (leyes, clientes de la organización que ha instalado el sistema, etc.), ingenieros involucrados en el diseño, desarrollo y mantenimiento del sistema, etc.
- Perspectivas asociadas con conocimiento del dominio y de la organización: el conocimiento del dominio de aplicación y de la organización suponen el conocimiento de las limitaciones (constraints) del sistema. Las limitaciones pueden ser físicas (p.e: funcionamiento de la red), organizacionales (p.e: hardware incompatible utilizado en diferentes divisiones de la compañía), humanos (p.e: tasa media de errores del operador), o pueden reflejar leyes, regulaciones o estándares locales, nacionales o internacionales. Estos tipos de perspectivas no pueden ser asociados a una clase de perspectiva determinada pero incluyen información recogida de diferentes fuentes (personas, documento y otros sistemas).

4.4.6 MÉTODO DE LEITE

En [Leite89], [Leite91] describen un modelo para la obtención de requisitos que se basa en recoger información de diferentes perspectivas y construir "vistas". Este método hace una mayor distinción entre puntos de vista y perspectivas. Así, una vista es una integración de diferentes tipos de información (como información de procesamiento de datos, de estructura de datos, etc.) de la misma perspectiva.

Una característica interesante de este método es que existe una relación 1-N entre las fuentes de información y las perspectivas. Una perspectiva es una posición

mental, por lo cual un mismo individuo puede proporcionar información desde varias y diferentes perspectivas. Esto refleja también el hecho de que la gente a menudo tiene un número diferente de roles dentro de la organización. Por ejemplo, una misma persona puede ser un usuario final del sistema y el responsable de la seguridad del sistema en general.

Un punto fuerte de este método es que proporciona un chequeo automático y una detección de problemas desde las etapas más tempranas del proceso de obtención de requisitos, es decir, proponen una técnica para la validación temprana de los requisitos basado en perspectivas.

4.4.7 VIM

VIM (Viewpoint Inconsistences Management) [Easterbrook95] es un enfoque que propone utilizar las perspectivas como mecanismo para manejar las inconsistencias entre especificaciones de requisitos parciales. En esencia, es similar al método VOSE, pero hace especial hincapié en el tratamiento de inconsistencias.

Según VIM, una perspectiva es una combinación gestionada localmente de un objeto que almacena conocimiento parcial del sistema y su dominio, y de un esquema de representación. De esta manera, el marco permite una especificación del sistema desde diversas perspectivas sin tener que predefinir una notación.

Las perspectivas que describen el sistema deben ser integrables para permitir completar la representación del sistema en su totalidad. Para ello, el marco proporciona un conjunto de reglas de consistencia.

VIM sigue el mismo mecanismo de generación de plantillas que VOSE.

4.4.8 MÉTODOS BASADOS EN ASPECTOS

Existe un enfoque más moderno para abordar la ingeniería de requisitos, que es el enfoque basado en aspectos. Los aspectos son aquellos objetivos organizativos y restricciones que restringen el sistema o el proceso a analizar. Se basa en extrapolar ideas y conceptos tratados en programación orientada a aspectos (AOP, Aspect Oriented Programming) a los requisitos "early aspects". Con este enfoque se define una estrategia de obtención de requisitos basada en aspectos que facilite la traducción de los mismos a un entorno de programación AOP y que pueda ser aplicable también a otros entornos más generales [Czarnecki00], [Clarke01], [Elrad01], [Lamsweerde04].

AORE (Aspects Oriented Requirements Engineering) [Grundy99], [Dinwal02] es uno de los primeros enfoques que introduce los conceptos de orientación a aspectos a nivel de requisitos. El método AORE propone una técnica para separar los requisitos aspectuales (no funcionales) y no aspectuales (funcionales), según sus reglas de composición.

AORE tiene como objetivo modularizar y componer los aspectos a nivel de requisitos, no sólo generando un documento de especificación de requisitos, sino también asegurando su consistencia. Esto se consigue mediante un mecanismo de detección y manejo de conflictos entre perspectivas, aspectos y requisitos.

La mayor aportación proporcionada por este método es la capacidad de separar y después componer requisitos funcionales y no funcionales, mediante un soporte de composición simple pero poderoso y a la vez flexible. Este soporte es un soporte de identificación de conflictos, resolución y validación de los requisitos a lo largo de todo el ciclo de vida del desarrollo de software.

GORE (Goal-oriented requirements engineering) se basa en el uso de objetivos para extraer, estructurar, elaborar, analizar, y documentar los requisitos [Lamsweerde04]. También es un modelo basado en vistas, donde las vistas muestran la forma en la cual los objetivos, objetos, agentes, escenarios, operaciones y propiedades del dominio están interrelacionados.

4.5 RESUMEN

Como ya hemos dicho en apartados anteriores, el éxito de los proyectos de software depende en gran medida de las especificaciones de requisitos, las cuales deben representar los deseos de una gran variedad de personas, cada una de ellas con su perspectiva. Esto resulta complejo en el enfoque de LPS, ya que los métodos existentes no abordan la actividad de identificar e integrar las perspectivas, sino que se basan en mantener una descripción consistente, y que normalmente representa un único punto de vista, (generalmente el del departamento de marketing). Ante este escenario, el enfoque IROP se posiciona como una posible solución.

De esta forma, el objetivo de esta investigación es desarrollar y validar un método basado en perspectivas que proporciona un soporte procedimental para la obtención, análisis y especificación de requisitos bajo un enfoque de desarrollo de LPS.

CAPÍTULO 5- BASES DEL MÉTODO

En los capítulos anteriores hemos presentado tanto los conceptos clave de la ingeniería de requisitos para productos a medida y /o para líneas de productos software, como los trabajos existentes en el área de la IR basada en perspectivas. Llegados a este punto, el lector dispone de la información necesaria para entender las motivaciones que han propiciado este trabajo, y las aportaciones que este trabajo proporciona.

En este capítulo y en el siguiente se describe el grueso del trabajo materializado en el desarrollo de un método para la Elaboración de Requisitos para Líneas de Producto Software, al cual denominaremos ELVIRA (acrónimo que proviene de ELaboración con VIstas de los Requisitos de una líneA de productos software)

En este capítulo se presenta una visión general de ELVIRA como solución al problema planteado en la primera parte del documento y se desarrollan los fundamentos del ámbito de la ingeniería de software que soportan la misma: la integración de stakeholders, QFD, la toma de decisiones, el tratamiento de inconsistencias y la priorización.

5.1 VISIÓN GENERAL DE LA SOLUCIÓN

En este apartado se proporciona una visión general de ELVIRA; así, en los próximos puntos, se identifica el momento dentro del proceso de desarrollo de software donde se aplica y de forma más concreta, se presenta el método integrado en cada fase del proceso de ingeniería de requisitos bajo el enfoque LPS. También se dedica

un apartado para explicar la utilización del lenguaje natural como notación utilizada en la aplicación del método, y la utilización de la técnica de matrices como soporte de toda la información.

5.1.1 ELVIRA_EN QUÉ MOMENTO SE APLICA

La idea de construir sistemas de software bajo el enfoque de LPS, está focalizada en la mayoría de las ocasiones hacia la derivación de productos o hacia la producción. Así, como hemos visto en el capítulo 3, son muchos los trabajos realizados en este sentido. Pocos se plantean la estructuración de productos en base a las demandas del mercado, es decir, no tienen en cuenta las *actividades de pre-desarrollo.*

Sin embargo, son numerosos los trabajos que afirman que afrontar adecuadamente la fase de pre-desarrollo de los nuevos productos contribuye directamente al éxito de los mismos. A pesar de todo, en la práctica, se observa que los recursos de las empresas se invierten en mayor grado en actividades posteriores al pre-desarrollo. Cooper y Kleinschmidt [Cooper88] encontraron que el 54 % del total de gastos se destinaba al lanzamiento, frente a un 39 % destinado al desarrollo del producto y a un 7% para actividades de pre-desarrollo. [Page98] advierte que las actividades de iniciación apenas ocupan una tercera parte del tiempo total de desarrollo, recomendando a las empresas un mayor esfuerzo en su realización. [Urban93] afirman que la última etapa, de lanzamiento, es la que absorbe el mayor compromiso monetario y la mayor dedicación de la dirección. También [Khurana97] aprecian serias deficiencias en la ejecución de las actividades de pre-desarrollo por parte de las empresas analizadas.

Por otro lado, está claro, que la habilidad de una organización para crear productos exitosos depende en gran medida de cómo obtenga, analice y utilice la información

asociada al mismo. Para desarrollar un nuevo producto o servicio, es esencial tener en cuenta "la voz del consumidor". Hay que conseguir que el "cliente" articule, estructure y priorice lo que quiere y necesita de un producto o servicio. Una comunicación directa con los clientes permite aprender de ellos y ajustar los productos a sus necesidades. Esto nos da pistas para considerar que debe existir una relación entre la ingeniería de requisitos y el diseño o estructuración de productos, de tal forma que la estructuración de un producto puede y debe ser una consecuencia directa de las necesidades de mercado [Ereño05a], [Ereño05], [Ereño08]. Es en este punto del proceso de desarrollo donde encaja el método ELVIRA. Hay que aclarar, sin embargo, que en este momento del proceso de desarrollo (entre el análisis y el diseño), no se tienen en consideración cuestiones de esfuerzos o coste. Posiblemente este sea el paso inmediatamente posterior.

5.1.2 ELVIRA Y EL PROCESO DE INGENIERÍA DE REQUISITOS

ELVIRA es un método general de ingeniería de requisitos orientado al enfoque de desarrollo de líneas de productos software. Como todo método de ingeniería del software, ELVIRA establece un marco de trabajo común, mediante la definición de una serie ordenada de actividades, que son aplicables a todos los proyectos de definición de LPS, con independencia de su tamaño o complejidad. En la *figura 7*, se puede ver el paralelismo entre un proceso general de IR-LPS y el método ELVIRA. ELVIRA añade una actividad inicial, la actividad de identificación de perspectivas, pero las etapas siguientes tienen una trazabilidad clara con cualquier proceso genérico, como se explica a continuación.

La obtención de requisitos\perspectivas, es la actividad mediante la cual los analistas extraen las necesidades de los stakeholders clave. Realizar esta captura de información no resulta sencillo principalmente por dos motivos:

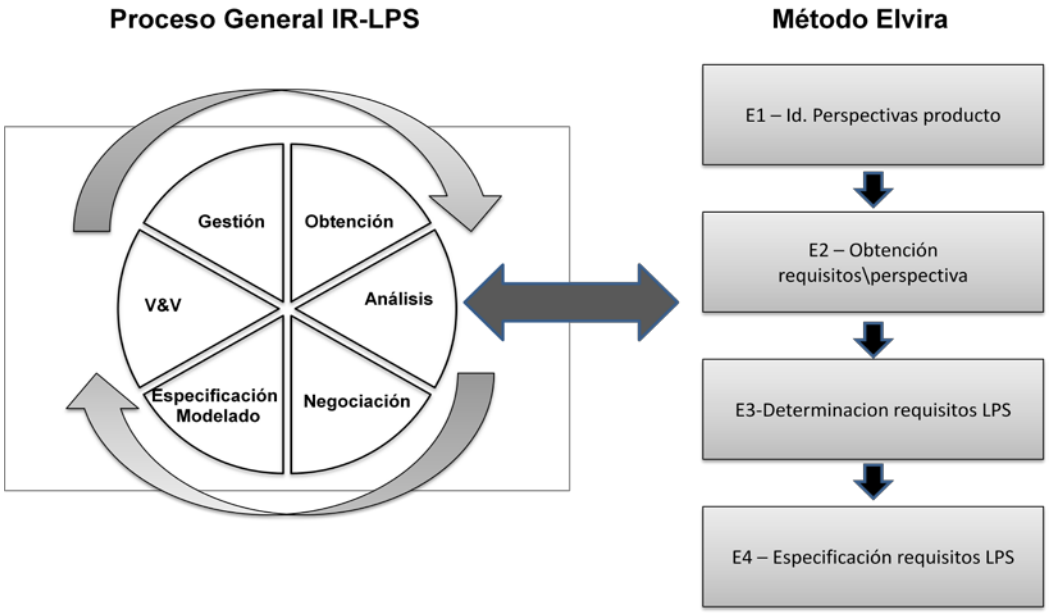


Figura 7: Paralelismo entre un proceso general IRLPS y ELVIRA

- Los deseos, implícitos o explícitos, de los consumidores, están flotando en el ambiente; los consumidores están predispuestos a expresar su opinión pero normalmente no tienen medios para ello y así solamente llegan los reflejos negativos en forma de reclamaciones, quejas, descontento, etc.
- Identificar y enumerar las expectativas de los stakeholders son dos operaciones extremadamente complejas y que pueden resultar costosas. Ambas actividades necesitan competencia, método, rigor y tiempo. El presupuesto que se suele reservar para este tipo de trabajo suele ser bajo, sin embargo también las inversiones y gastos dedicados al conocimiento del cliente son con diferencia mucho más rentables que otras inversiones.

La determinación de requisitos de la LPS es la actividad mediante la cual se analiza la información obtenida en la actividad anterior. En un enfoque de LPS el objetivo es identificar los requisitos comunes y variables de la familia de productos. ELVIRA utiliza técnicas de priorización de stakeholders y de perspectivas en este análisis.

La especificación de Requisitos de la LPS produce como resultado el documento de especificación de requisitos. En este documento se describirán las partes comunes del sistema y los discriminantes o características que diferenciarán unos sistemas de otros dentro del dominio. Este documento debe seguir alguna estructura formal. ELVIRA proporciona la información necesaria para esta actividad, pero no impone el formato de este documento.

Hay que añadir también, que a pesar de no soportar directamente las actividades de Validación y Gestión de Requisitos, ELVIRA gracias a la estructura de matrices que utiliza, permite la realización de estas actividades de forma sencilla y coherente.

5.1.3 UTILIZACIÓN DE LENGUAJE NATURAL

ELVIRA propone la utilización del lenguaje natural a lo largo de su aplicación. El motivo es la participación tan diversa de stakeholders a lo largo del mismo. Como dice Sommerville, en su trabajo [Sommerville97], el documento de especificación de requisitos básicamente tiene dos lectores: el usuario y los desarrolladores del sistema. Generalmente la información dedicada al usuario se escribe en lenguaje natural, para facilitar su comprensión. Se trata de una definición del dominio del problema. Mientras que la información dirigida al equipo de desarrollo es más técnica y contiene detalles del dominio de la solución. Si trasladamos esta situación al contexto de las LPS y además utilizando un método como ELVIRA que precisamente potencia la integración de todas las perspectivas posibles, se considera que el lenguaje natural será la forma más sencilla de trabajar en la definición del producto.

5.2 FUNDAMENTOS DEL MÉTODO

ELVIRA se apoya en una serie de conceptos fundamentales dentro del ámbito de la ingeniería del software. Concretamente, guía las actividades de IR_LPS bajo las directrices de un proceso de *toma de decisiones*, donde la *priorización* de los *stakeholders* y sus preferencias, permiten una toma de decisiones cuantitativa, cuyo resultado determina los requisitos comunes y variables de la LPS. Para la realización de estas actividades se apoya en la herramienta QFD (Despliegue de la Función Calidad – Quality Function Deployment). En la *figura 8* se muestra la relación entre estos conceptos.

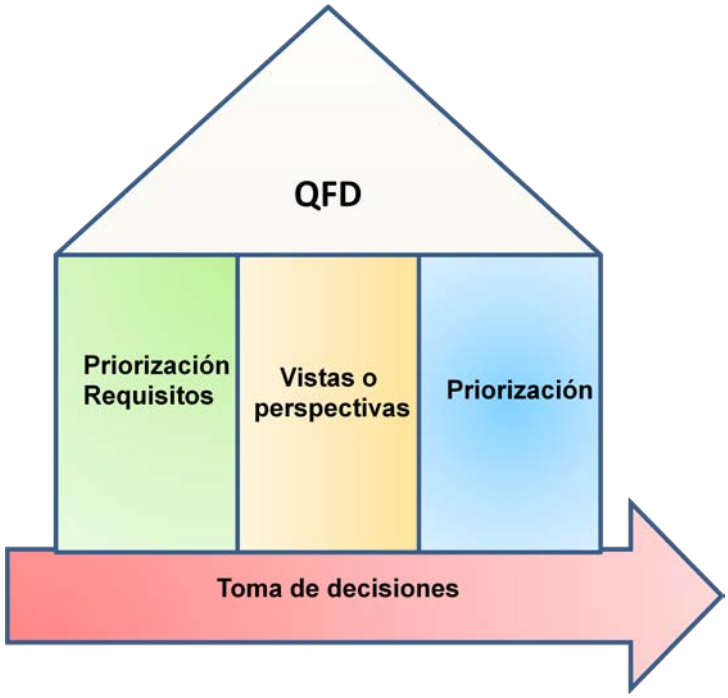


Figura 8: Fundamentos básicos de ELVIRA

En los siguientes apartados vamos a desarrollar en profundidad los fundamentos del método ELVIRA: la integración de stakeholders, QFD, la toma de decisiones, el tratamiento de las inconsistencias y la priorización de requisitos y de stakeholders.

5.2.1 LA INTEGRACIÓN DE STAKEHOLDERS

Cómo se ha comentado en los capítulos tercero y cuarto, los programadores no son el único rol imprescindible en el desarrollo de productos de software. Hablamos así de toda una lista de roles entre los que destacamos: analistas, diseñadores, arquitectos, testadores, jefes de proyecto, responsables de calidad, clientes, usuarios finales, etc. Todas estas figuras están dentro del paraguas "stakeholders".

Este método integra a los diversos stakeholders en la definición del producto. Así, define su trato con los stakeholders situándolos como el origen de los requisitos del producto.

ELVIRA adopta el enfoque de *vistas* para integrar todas las posibles perspectivas de producto. Está claro que cada stakeholder mantiene su propia y diferente vista del dominio del problema, lo cual origina diferentes requisitos del sistema a desarrollar. Todos ellos son correctos tomados individualmente, pero a menudo resultan inconsistentes tomados en su conjunto. Los enfoques basados en vistas permiten obtener una definición completa del sistema soportando las inconsistencias individuales.

ELVIRA integra stakeholders del dominio del problema y del dominio de la solución. De hecho esta diferenciación es crucial para el posterior tratamiento de todas estas perspectivas.

5.2.2 QFD Y LAS MATRICES DE CORRELACIÓN

QFD (Quality Function Deployment -Despliegue de la Función Calidad) se puede definir como un proceso estructurado y metódico para obtener la voz del cliente o usuario final y recogerla en todas las fases del análisis, diseño y desarrollo de un producto o servicio [Mizuno97]. El concepto básico de QFD es la transmisión de las necesidades del cliente o usuario final a través de todo el proceso de desarrollo de producto.

El concepto de QFD fue introducido en Japón por Yoji Akao [Akao90], [Mizuno94] en 1966, y se aplicó inicialmente en el Kobe Shipyard de Mitsubishi Heavy Industries Ltd. A partir de entonces QFD se convirtió en una metodología de desarrollo de productos y servicios ampliamente aceptada en Japón. A principios de los años ochenta se aplicó en Xerox y a partir de entonces también fue ampliamente utilizada en EEUU.

QFD sobre todo se ha aplicado en el desarrollo de productos tangibles; es decir, productos físicos. Sin embargo, no es exclusivo para este tipo de productos. Dentro de la industria de desarrollo de software, QFD se ha utilizado como una metodología de diseño y desarrollo [Erikkson93], [Thackery90]. En Haag et al. [Haag96], se reflexiona sobre la adaptación y uso de lo que ellos denominan SQFD (Software Quality Function Deployment) en el desarrollo de software de las grandes firmas de software como DEC, AT&T, Hewlett-Packard y Texas Instruments. En este trabajo utilizan SQFD como una técnica “front-end” de solicitud de requisitos adaptable a cualquier metodología de ingeniería del software y que permite la solicitud y definición de requisitos críticos de una forma cuantitativa. Lamia [Lamia95] aplica QFD al análisis y diseño orientado a objetos y muestra cómo QFD puede ser utilizado para la recopilación inicial de información de una forma sencilla,

permitiendo una estructuración natural sin la necesidad de una amplia formación en los conceptos y métodos de la orientación a objetos.

Para alcanzar sus objetivos, QFD se basa en la construcción de matrices o tablas de calidad donde se confrontan datos desde diferentes perspectivas.

No existe un número determinado de matrices a utilizar, depende de cada caso, pero la idea general es que en cada nueva matriz se desciende en el nivel de abstracción, de tal forma que las matrices iniciales utilizan un lenguaje más funcional, y las últimas entran en detalles más técnicos. Se supone que el número de matrices corresponde al número de fases del proceso de desarrollo, y el proceso de desarrollo es algo "a medida". Por ejemplo, el ASI (American Supplier Institute) sigue un enfoque de 4 fases, en el cual el equipo QFD despliega los requisitos de cliente en características de producto, las características de producto en características de partes, las características de partes en características de proceso y finalmente las características de proceso en características de producción. Otros enfoques podrían incluir más fases: de requisitos de cliente a requisitos de diseño, ingeniería de diseño, características de producto, operaciones de manufactura y finalmente controles de producción /calidad.

La primera y más utilizada de estas matrices se llama Casa de la Calidad ("House of Quality" - HOQ). La Casa de la Calidad (HOQ) consiste en la creación encadenada de una serie de matrices de correlación, que físicamente terminan adoptando la forma de una casa (véase la *figura 9*) [Cohen95].

La construcción de la Casa de la Calidad surge como consecuencia del flujo de informaciones a lo largo de una serie de matrices. En cada matriz se confronta una información determinada. Se comienza confrontando un conjunto de requisitos ("*qué*") contra posibles soluciones técnicas ("*cómo*"). Esto se traduce en la relación (*"qué-cómo)*". Para completar la casa se añade información sobre el por qué de la

existencia de los requisitos ("*por qué*"), las correlaciones entre las posibles soluciones técnicas (*cómo-cómo*) y objetivos de desarrollo cuantificados (*cuánto*).

Son varias las características que permiten utilizar QFD como herramienta de soporte del método ELVIRA:

- QFD permite la definición de las necesidades de los clientes y su traducción en planes de producción de productos que cubren dichas necesidades.

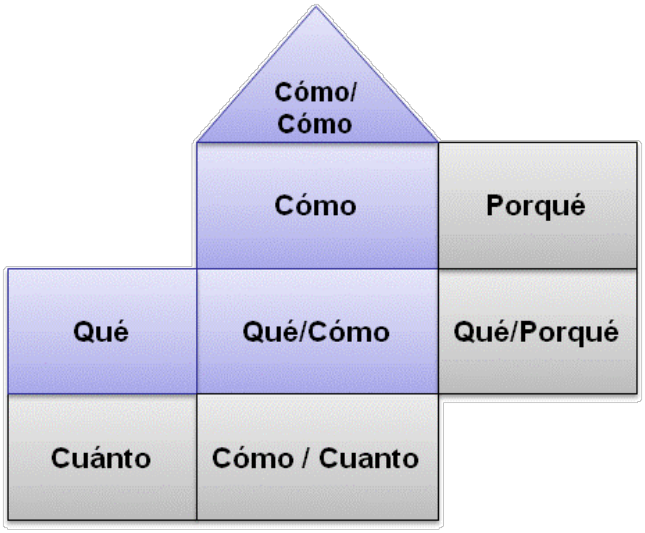


Figura 9: La casa de la calidad.

- QFD involucra a los clientes en el proceso de desarrollo de un producto desde las fases iniciales. Traduce lo que el cliente quiere en lo que la organización produce; esto permite a una organización priorizar las necesidades de los clientes, encontrar respuestas innovadoras a esas necesidades, y mejorar los procesos hasta una efectividad máxima.
- QFD está orientado a su aplicación en equipos multidisciplinares, equipos formados por personas de diferentes perfiles pero todas ellas involucradas

de alguna forma en el desarrollo de un producto: marketing, diseño, aseguramiento de la calidad, soporte final, etc. Esta participación multidisciplinar permite aumentar la variedad funcional, debido a las diferentes perspectivas y evita la posibilidad de que se eliminen funciones consideradas poco importantes desde una determinada perspectiva.

Además de estas características, otro de los motivos para elegir QFD son los beneficios tangibles y los beneficios intangibles. Entre los beneficios tangibles se pueden resaltar la reducción en el tiempo de desarrollo, la detección anticipada de posibles cambios, el incremento de la confianza en el diseño y el control de los factores económicos. Entre los beneficios intangibles hay que señalar la mejora de la satisfacción de los usuarios, la base de conocimiento adquirida para futuras planificaciones, y el resultado obtenido, lo cual supone una documentación completa.

Además de los motivos ya expuestos, la razón principal para la selección de QFD como herramienta de soporte en ELVIRA, es su capacidad de adaptación a cualquier contexto de desarrollo de productos. Como dice Mikel Sorli [Sorli08], [Sorli09]. "*Para aplicar QFD no hay un camino único sino que cada aplicación debe seguir sus propias directrices y crear su propia herramienta (matriz) en función de las peculiaridades intrínsecas de cada proceso".*

El uso de QFD en esta investigación se ha planteado de la siguiente forma:

- QFD está pensado para el desarrollo de nuevos productos. En este trabajo y viendo el desarrollo y situación del mercado, se propone su utilización para la definición de una LPS.
- QFD solo se utiliza para las fases iniciales del desarrollo de LPS, es decir la ingeniería de requisitos. Únicamente se aplican las dos primeras etapas de la metodología QFD. La etapa 1 concierne a la investigación de mercado. Es una etapa de recogida y análisis de las informaciones y por lo tanto está

orientada esencialmente hacia el exterior de la empresa. Su finalidad es conocer al cliente, escucharle y recoger sus expectativas. La etapa 2 parte de los resultados de la etapa 1, es decir, de la lista de las expectativas del cliente, y teniendo en cuenta factores como recursos tecnológicos, recursos humanos, recursos financieros, etc. decide cuál es el producto a desarrollar, es decir, da una respuesta a la oportunidad identificada y definida en la etapa anterior. Se trata, pues, de la etapa de definición del producto, en términos de funciones de servicio.

5.2.3 LA TOMA DE DECISIONES

La toma de decisiones es el proceso durante el cual se debe escoger entre dos o más alternativas. Con frecuencia se dice que las decisiones son algo así como el motor de los negocios y en efecto, de la adecuada selección de alternativas depende en gran parte el éxito de cualquier organización.

Las fases iniciales del proceso de desarrollo de productos de software son básicamente una toma de decisiones continua [Ereño05a]:

- ¿Qué espera el cliente de este producto?
- ¿Qué funcionalidades ofrece este producto de software?
- ¿Qué aspectos no funcionales son necesarios?

Así el proceso de IR se puede ver como un proceso de toma de decisiones [Aurum03], [Regnell01]. En [Alenljung05] afirman que un requisito puede entenderse como la representación de una decisión relativa a un aspecto particular de un sistema. Estas decisiones gobiernan el proceso de desarrollo, así como la naturaleza del producto. Si se toman las decisiones equivocadas, tanto el proceso de desarrollo como el producto se verán afectados de forma negativa.

Si en lugar de un único producto, nos enfrentamos a la definición y desarrollo de toda una línea de productos software, las decisiones se multiplican [Ereño05], [Ereño05a]:

- ¿Qué funcionalidades ofrece esta línea de productos software?
- ¿Cuáles de estas funcionalidades son comunes a todos los productos de la línea?
- ¿Cuáles son opcionales?
- ¿Cómo estructurar el producto para soportar de forma robusta esta comunalidad/variabilidad?

Esta toma de decisiones está lejos de resultar sencilla. Hay que tener en cuenta que las fases iniciales del proceso de desarrollo se caracterizan por ser parte de un ambiente incierto y dinámico, donde toman parte diferentes actores, con diferentes objetivos y bajo una restricción temporal [Orasanu93]. Tampoco se debe olvidar que la ingeniería de requisitos es la fase del desarrollo de productos de software donde la componente humana es mayor. Esto hace que el proceso de toma de decisiones resulte crítico.

Existe otra diferencia a considerar bajo ambos enfoques de desarrollo, y es el momento en el cual se toman las decisiones. En el desarrollo de productos de software específico para clientes concretos, el proceso de toma de decisiones se puede situar sobre todo en la fase de diseño. Este es el momento en el cual el diseñador debe decidir la forma en la cual traducir las necesidades de los clientes en requisitos de ingeniería o técnicos. En el enfoque de LPS sin embargo, la toma de decisiones se adelanta en el tiempo. Ya no existe un cliente con unas necesidades concretas sobre un producto. En el enfoque de LPS todo el mercado es cliente potencial de nuestros productos. El objetivo del equipo de desarrollo es diseñar un marco que represente a la LPS completa y a partir de él, personalizar los productos individuales.

Hemos desarrollado el método ELVIRA con el propósito de soportar el proceso de decisiones en la fase de IR_LPS. Esta premisa, se ve reforzada por otros trabajos, como el de Ngo The [NgoThe03] que afirma que las compañías exitosas en el futuro serán aquellas que tengan un enfoque integrado para la toma de decisiones y la gestión de requisitos. Las compañías de éxito serán aquellas que "utilicen su capital intelectual en el proceso de toma de decisiones y sean capaces de mantener una trazabilidad entre este proceso y el soporte de información".

ELVIRA soporta una toma de decisiones para la identificación de partes comunes y variables. Para ello, se apoya en una técnica de decisión como es la priorización y en métodos como QFD. En el punto anterior, ya se ha presentado QFD, sin embargo es importante resaltar el soporte de este método en la toma de decisiones. ELVIRA utiliza el método QFD (Quality Function Deployment) como soporte a la toma de decisiones en el desarrollo de una línea de productos software. La *tabla 6* resume los motivos por los cuales se considera que este método es adecuado [Ereño05], [Ereño05a].

5.2.4 EL TRATAMIENTO DE LAS INCONSISTENCIAS

De acuerdo con [Duran00], el análisis es la actividad de la ingeniería de requisitos en la que los ingenieros de software analizan los requisitos identificados previamente, para detectar posibles conflictos, profundizar en el conocimiento del problema y desarrollar los modelos conceptuales que serán la base del diseño. Aunque, tradicionalmente, el análisis ha supuesto un paso intermedio para aliviar la transición desde los requisitos hasta el diseño, no cabe duda que es durante el análisis, y al estudiar en profundidad los requisitos, cuando es posible detectar las inconsistencias en los requisitos, que habrán de tratarse.

Características de la toma de decisiones	Cómo aborda QDF este aspecto
No deben existir conflictos acerca del objetivo final	QFD permite determinar de forma cuantitativa los objetivos
El tomador de decisiones tiene que conocer los criterios, las alternativas y las posibles consecuencias de su determinación.	QFD se basa en la correlación entre características (Qué/Cómo/Cuánto). Esto permite conocer cómo una decisión afecta a dichas características
Preferencias claras: deberían poderse asignar valores numéricos y establecer un orden de preferencia para todos los criterios y alternativas posibles.	QFD trabaja con datos cuantitativos
No se debe desestimar el futuro.	QFD permite visualizar configuraciones futuras
Diferentes perspectivas ayudan a tomar la decisión	QFD permite el trabajo conjunto de equipos de trabajo multidisciplinares.

Tabla 6: QFD como técnica de soporte en la toma de decisiones

Utilizando términos más precisos, se hace necesario diferenciar los términos *solapamientos (u overlaps)* e *inconsistencias.* Según Finkelstein, los solapamientos son el prerequisito para las inconsistencias [Finkelstein96], por ese motivo se deben detectar los solapamientos y explicitarse en busca de inconsistencias. Spanoudakis, define solapamiento como un relación entre las interpretaciones de dos especificaciones [Spanoudakis99], y declara que la presencia de un solapamiento está directamente relacionada con la necesidad de determinar si las especificaciones que se solapan son consistentes entre sí. En el trabajo de Spanoudakis [Spanoudakis99], el objetivo era identificar los solapamientos, para luego identificar la posible existencia de inconsistencias y resolverlas. Como resultado de su investigación, identificó distintos tipos de solapamientos:

- Solapamiento total: si el conjunto de objetos tratado es el mismo.
- Solapamiento parcial: si hay ciertos objetos comunes entre los objetos tratados.

- Solapamiento inclusivo: si uno de los conjuntos tratados es un subconjunto del otro.

En este trabajo de tesis el enfoque es diferente. ELVIRA permite tratar las inconsistencias, no con el objetivo de eliminarlas, sino de utilizarlas en el análisis de partes comunes y variantes de una LPS. Así los solapamientos reflejarán el mismo deseo por parte de diferentes stakeholders, y las inconsistencias reflejarán deseos individuales (incluso en ocasiones contrarios) de los stakeholders. Esto, que en un desarrollo de software a medida no tiene sentido, resulta esencial tenerlo en consideración cuando el objetivo es desarrollar una familia de productos donde cada cliente final obtendrá el producto a su medida.

En nuestro enfoque, la identificación de los solapamientos cae en manos de los ingenieros de software, al igual que en otros trabajos como [Easterbrook91] y [Zave97].

ELVIRA soporta la identificación de solapamientos y su resolución. Así el método guía a los ingenieros de software en el tratamiento de los solapamientos de una manera estructurada.

5.2.5 LA PRIORIZACIÓN

En el diccionario de la Real Academia de la Lengua se define *priorizar* como *"dar prioridad a algo",* y define *prioridad* como *"anterioridad o procedencia de algo respecto a otra cosa que depende o procede de ello*".

La priorización es el soporte a la toma de decisiones. Priorizar un conjunto de elementos respecto a un criterio determinado, permite obtener un conjunto ordenado de esos elementos y así decidir con un criterio cuantitativo.

La priorización no es un término novedoso en el área de la ingeniería del software. La comunidad de la ingeniería de software, ha realizado actividades de priorización, más o menos formales, prácticamente desde sus orígenes [Lehtola04], [Glinz07], [Woolridge07].

La priorización, entendida en el contexto de la ingeniería de software, no tiene por qué ceñirse a momentos o etapas marcados, es un proceso iterativo y debería realizarse a diferentes niveles de abstracción y con diferente información de las diferentes fases del desarrollo de software. Sin embargo, es posible identificar dos etapas en las cuales la actividad de priorización tiene más sentido:

- En los inicios de la IR, cuando los requisitos son bastante vagos y poco definidos, las técnicas de priorización se utilizan con dos objetivos:
    - Planificar el trabajo de desarrollo intentando identificar qué requisitos desarrollar en primer lugar
    - Identificar los requisitos que tendrá el producto final, teniendo en cuenta los stakeholders implicados
- En la etapa de mantenimiento, donde se analizan las nuevas funcionalidades del producto y sus sucesivas versiones, las técnicas de priorización ayudan en la elección. Este enfoque tiene sentido sobre todo en el desarrollo de productos de software orientados a un mercado abierto.

Son numerosas las motivaciones de la priorización [Berander05]:

- Permite a los stakeholders decidir los requisitos fundamentales del sistema.
- Ayuda a la planificación ordenada de futuras implementaciones de conjuntos de requisitos, que conformarán nuevas versiones del producto software.

- Permite negociar entre los límites inicialmente definidos del proyecto y las limitaciones de calendario, presupuesto, recursos, tiempo de salida al mercado y calidad.
- Permite equilibrar los beneficios de cada requisito y su costo asociado.
- Permite determinar las implicaciones de los requisitos en la arquitectura software y cómo esto afecta a la evolución del producto y su costo.
- Permite la identificación de un subconjunto de los requisitos que produzcan un sistema que satisfaga al usuario.
- Ayuda a estimar la satisfacción esperada del cliente.
- Permite obtener una ventaja técnica y optimizar las oportunidades del mercado.
- Permite determinar la importancia relativa de cada requisito para proporcionar el mayor valor al menor costo.

Analizando las motivaciones anteriores deducimos que los elementos de priorización más habituales en el contexto de la ingeniería de software son dos: los requisitos y los stakeholders. En los siguientes apartados se explica en detalle ambos tipos de priorización y se presenta la forma en la cual ELVIRA lo soporta.

Priorización de Requisitos

La priorización de requisitos ayuda a identificar los requisitos más "valiosos" entre un conjunto de requisitos. Existen dos enfoques principales de priorización de requisitos:

- La priorización basada en el orden de implementación. Se prioriza el orden de implementación y entrega de los requisitos de cara a la definición de un plan de trabajo que satisfaga tanto al cliente como al desarrollador. El objetivo es la búsqueda de un equilibrio entre los costos y riesgos del

desarrollo y el tiempo que supone. El propósito final es el desarrollo completo del producto pero con la priorización se decide qué funcionalidades se desarrollarán antes y cuáles después [Wieggers99a]. Tradicionalmente es el enfoque de priorización más utilizado aunque sus detractores temen la posibilidad de que no se implementen los requisitos menos prioritarios [Wiegger99].

- La priorización basada en el grado de importancia de los requisitos. Se priorizan los requisitos en función de lo importantes que resulten para los stakeholders [Lehtola04]. La importancia se puede determinar en base a preferencias personales, valor de negocio, costo de implementación, riesgo, etc. Por ejemplo, en [Sommerville97] clasifican a los requisitos en:
    - Esenciales: Sin ellos el sistema no tiene sentido.
    - Útiles: en caso de no existir reducen la efectividad del sistema.
    - Deseables: hacen el producto más atractivo a ciertos stakeholders.

Los requisitos se pueden priorizar también teniendo en cuenta otros aspectos. Entre los aspectos más utilizados en las actividades de priorización se identifican los siguientes [Berander05]:

- Penalización: se trata de evaluar la penalización o pérdida en que se incurre si un requisito no es considerado.
- Costo de desarrollo: se ordenan los requisitos en función del esfuerzo en horas/persona que supone desarrollarlos. Generalmente son los desarrolladores quienes priorizan según este aspecto. Es un aspecto difícil de considerar ya que se ve afectado por otros factores como son: la complejidad de los requisitos, la habilidad para reutilizar código ya existente, la cantidad de testeo y documentación exigida, etc.

- Tiempo: se ordenan los requisitos en función del tiempo que supone desarrollarlos. Este factor también se ve influenciado por otros como el grado de paralelismo en el desarrollo, las necesidades de formación, la necesidad de desarrollar infraestructuras, etc.
- Riesgo: se ordenan los requisitos en función del riesgo asociado a cada uno de ellos (riesgos técnicos, de mercado, normas) y se puede llega a determinar el riesgo asociado al proyecto.
- Otros: volatilidad, beneficios financieros, beneficios estratégicos, etc.

Los aspectos anteriormente citados se pueden utilizar de forma individual en la priorización de requisitos, o se pueden combinar varios aspectos. Por ejemplo, en el enfoque Cost-Value, se priorizan los aspectos valor (importancia) y costo, con el objetivo de implementar aquellos requisitos que dan el mayor valor por un costo determinado [Karlsson97]. La técnica Planning Game (PG) para eXtreme Programming (XP) utiliza un enfoque similar y prioriza importancia, esfuerzo (costo) y riesgo [Beck99]. En el enfoque de Wiegers [Wieggers99] el valor relativo (importancia) se divide entre el costo relativo y el riesgo relativo para determinar los requisitos que tienen el balance más favorable de valor, costo y riesgo. Este enfoque además permite asignar diferentes pesos a los diferentes aspectos para así favorecer el aspecto más importante de los tres en situaciones específicas. Como se puede ver, son muchas las alternativas de combinación de aspectos. La elección depende la situación específica y de lo importante que resulte tener información sobre un aspecto determinado.

Priorización de Stakeholders

La priorización de stakeholders parte del hecho de que no todos los stakeholders son igual de importantes respecto al producto y por ese motivo se deben priorizar.

Se deben tener en cuenta los requisitos más importantes de los stakeholders más importantes.

Una forma de hacerlo es clasificar a los stakeholders en uno de los siguientes grupos: críticos, importantes, secundarios [Glinz07]. En este trabajo proponen realizar esta priorización basándose en el riesgo que supondría ignorar o desatender a un stakeholder:

- Si el hecho de desatender a un stakeholder supone el fin del proyecto o limita el uso del sistema, entonces el rol de stakeholder es crítico.
- Si el hecho de desatender a un stakeholder tiene un impacto negativo importante en el sistema, entonces el stakeholder tiene un rol prioritario
- Si el hecho de desatender a un stakeholder tiene un impacto marginal, entonces el stakeholder tiene un rol secundario.

Existen otros enfoques más globales, que integran varios de los aspectos de la clasificación anterior. Así, en [Woolridge07] proponen un modelo denominado OBSRAM para identificar los riesgos asociados a los stakeholders. Este modelo proporciona un enfoque para identificar stakeholders durante la fase de IR, identificar su influencia en el proyecto, identificar el impacto del proyecto en los stakeholders y evaluar el riesgo de un stakeholder negativo. Además en [Woolridge07] consideran importante identificar desde el inicio a aquellos stakeholders que tienen una percepción negativa del sistema, ya que solo originarán problemas. En este enfoque diferencian los ámbitos sobre los que actúan los stakeholders. Por un lado está el dominio del problema, de donde surgen los requisitos de dominio, y por otro el ámbito del proyecto de donde surgen los requisitos funcionales. Por eso proponen diferentes formas de identificar a los stakeholders del dominio del problema y a los stakeholders del proyecto. Una vez identificados se observa que algunos de ellos se superponen.

ELVIRA permite la priorización de stakeholders y de requisitos. Esto resulta esencial para el enfoque de líneas de producto software, donde no hay un único cliente, sino que el producto va dirigido a un segmento amplio de mercado, y lo que se busca es contentar al mayor número de clientes potenciales.

5.3 RESUMEN

En este capítulo hemos presentado una visión general de ELVIRA, explicando en profundidad la manera en la cual ELVIRA aborda actividades fundamentales de la IRLPS, tales como la integración de stakeholders, la toma de decisione,s el tratamiento de inconsistencias y la priorización. También hemos explicado el uso de QFD como soporte al método. Todo este conocimiento resulta esencial y debe ser previo a la presentación del detalle del método que se desarrolla en el próximo capítulo.

CAPÍTULO 6- ELVIRA

En este capítulo describimos el método ELVIRA en detalle e ilustramos su aplicación en un escenario. ELVIRA proporciona una guía durante el proceso de ingeniería de requisitos para líneas de producto software. En los siguientes apartados presentamos una visión general de ELVIRA para a continuación detallar su estructura en etapas y pasos. El capitulo finaliza con un análisis de los resultados que proporciona ELVIRA.

6.1 VISIÓN GENERAL DE ELVIRA

El método ELVIRA (**El**aboración con **vi**stas de los **r**equisitos de una line**a** de productos software) proporciona una manera formal y cuantitativa de integrar a los usuarios y sus preferencias, en la definición de una línea de productos software [Ereño07].

ELVIRA permite la elaboración del catálogo de requisitos de una LPS [Ereño05]. Este método involucra directamente a todos los stakeholders y permite una toma de decisiones cuantitativa cuyo resultado determina los requisitos comunes y variables de la LPS [Ereño05], [Ereño05a]. En la *figura 10* se muestra el método de forma general. En esta figura se aprecia cómo ELVIRA se apoya en el método QFD, ya que las matrices HOQ se distinguen fácilmente. Estas matrices soportan el flujo de información proporcionado por los stakeholders.

Una de las características de ELVIRA es la integración de los stakeholders desde las etapas más tempranas del desarrollo del producto. Esto lo hace de forma

sistemática proporcionando soporte a las diferentes vistas o perspectivas participantes. En la *figura 10*, se puede ver una leyenda que muestra gráficamente las diferentes perspectivas participantes: en color gris la perspectiva de mercado o cliente, en color verde la perspectiva de desarrollo de la solución y en color rojo la perspectiva de diseño. Todas estas perspectivas son stakeholders del sistema. La *figura 10* también muestra el momento y orden de participación de cada uno de los stakeholders. El flujo de información proporcionado por los diversos stakeholders va generando las diferentes matrices QFD. Como resultado se obtiene la información necesaria para el diseño de la arquitectura del sistema, información interesante para los diseñadores.

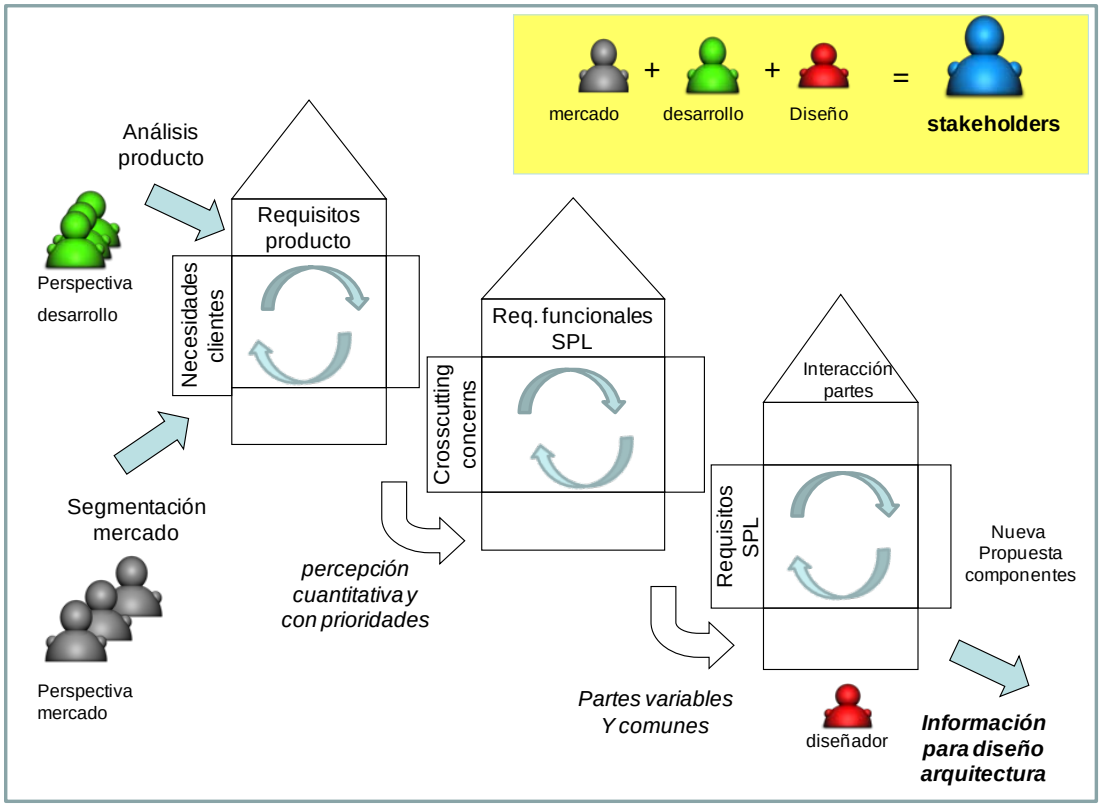


Figura10: Método ELVIRA. Visión general.

6.2 DESCRIPCIÓN DEL MÉTODO

ELVIRA consta de 3 etapas y 9 pasos [Ereño05], [Ereño05a]. En la *figura 11* se puede ver esta estructura de etapas y pasos. En los próximos apartados se explica detalladamente cada uno de ellos.

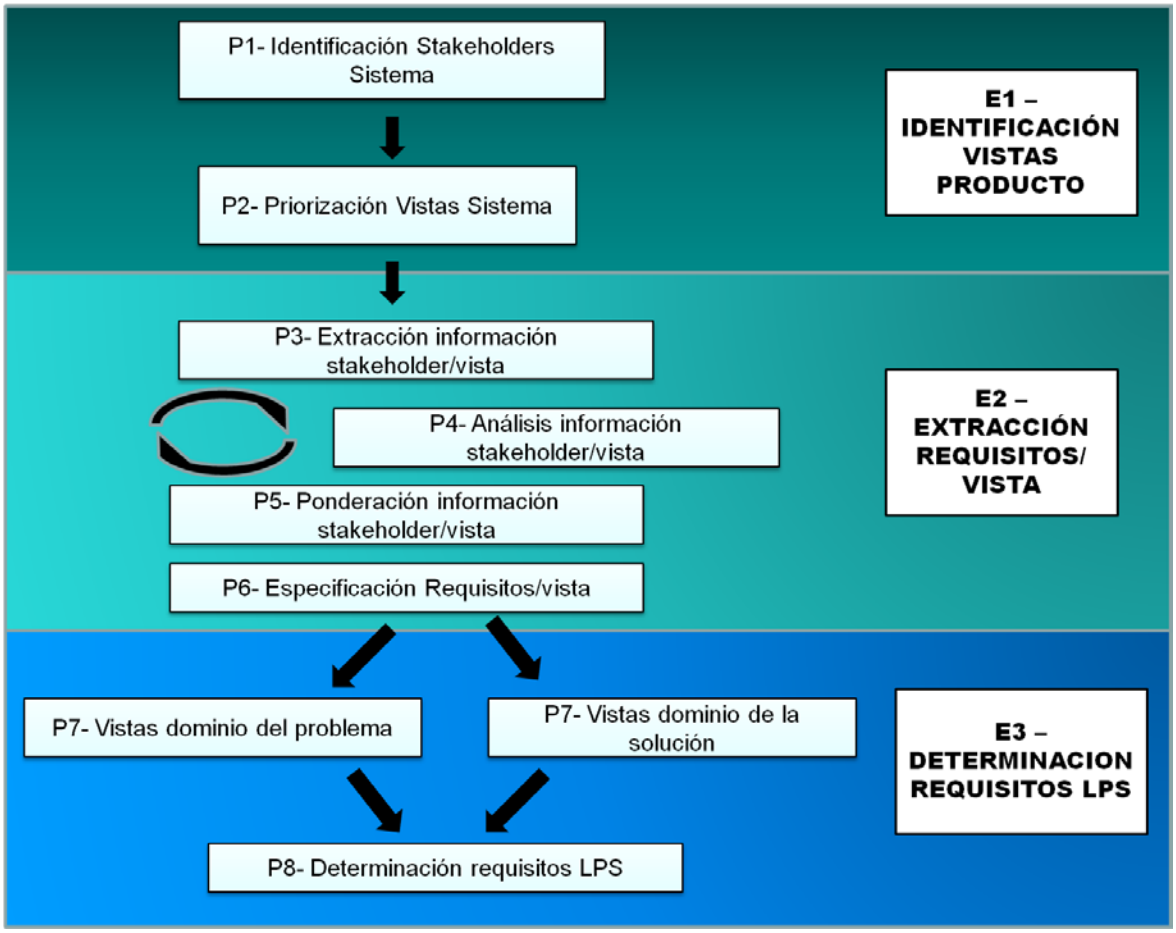


Figura11: Etapas y pasos del método ELVIRA

6.2.1 ETAPA 1: IDENTIFICACIÓN DE LAS VISTAS DEL PRODUCTO

Sobre un mismo producto existen diferentes vistas o perspectivas. Los diversos stakeholders del producto tendrán sus propias expectativas del mismo. El objetivo de esta etapa es identificar todas las vistas o perspectivas existentes sobre el producto y el grado de influencia de las mismas sobre el producto y su desarrollo.

El resultado de esta etapa se materializa en una lista que contendrá al menos dos columnas. En la primera de las columnas se anotan las vistas o perspectivas identificadas y en la segunda se indica el grado de importancia o el peso de cada clase de vista respecto al producto.

Esta etapa consta de dos pasos:

- La identificación de los stakeholders del sistema.
- La priorización de las vistas del sistema.

Paso 1: Identificación de los stakeholders del sistema

El primer paso para identificar las vistas o perspectivas de un producto es identificar los stakeholders asociados al desarrollo del mismo. Para la identificación de los stakeholders, ELVIRA, al igual que [Newman95], [Pouloudi97] propone la distinción de dos tipos de stakeholders.

- Los stakeholders del dominio del problema, donde se encuentran los diversos clientes y usuarios del sistema, así como los afectados por la puesta en marcha del mismo. Los tipos de stakeholders varían de acuerdo con la

naturaleza del sistema, por ejemplo no es lo mismo un producto de consumo, que un servicio como un control de tráfico.

- Los stakeholders del dominio de la solución donde se encuentran todos los roles relacionados con el desarrollo del producto, desde los roles de marketing, hasta los de mantenimiento, pasando por el analista, el diseñador, etc.

En la *figura 12* se puede ver la clasificación de stakeholders en función del dominio para un escenario de una aplicación de gestión de biblioteca.

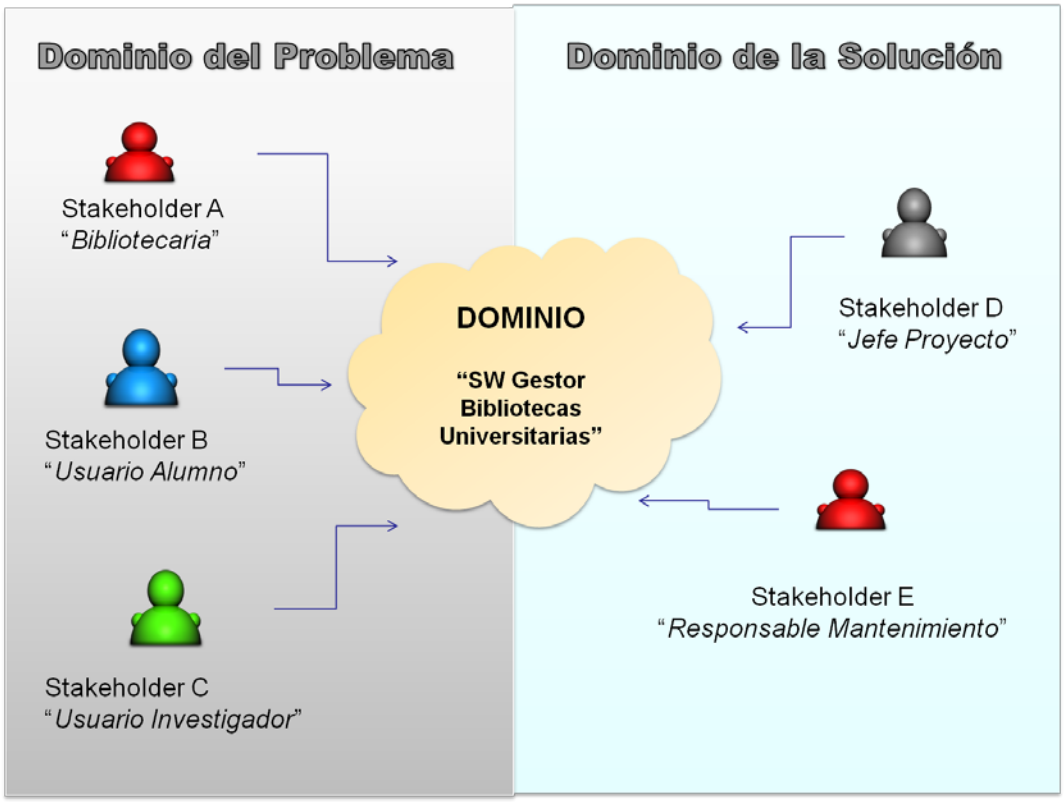


Figura12: Clasificación de stakeholders.

Como ELVIRA es un método enfocado al desarrollo de LPS, va un poco más allá en la definición de stakeholder. ELVIRA toma en consideración las preferencias de los diferentes stakeholders en diferentes entornos. En la *figura 13* se plantea esta situación. En la parte izquierda de la figura, se muestra la situación de un desarrollo de software a medida para un dominio concreto (una biblioteca universitaria) y los stakeholders identificados. En la parte derecha de la figura se muestra la situación de un desarrollo de LPS para el mismo dominio. En este caso, un mismo stakeholder en un entorno diferente puede tener una vista o perspectiva diferente del sistema.

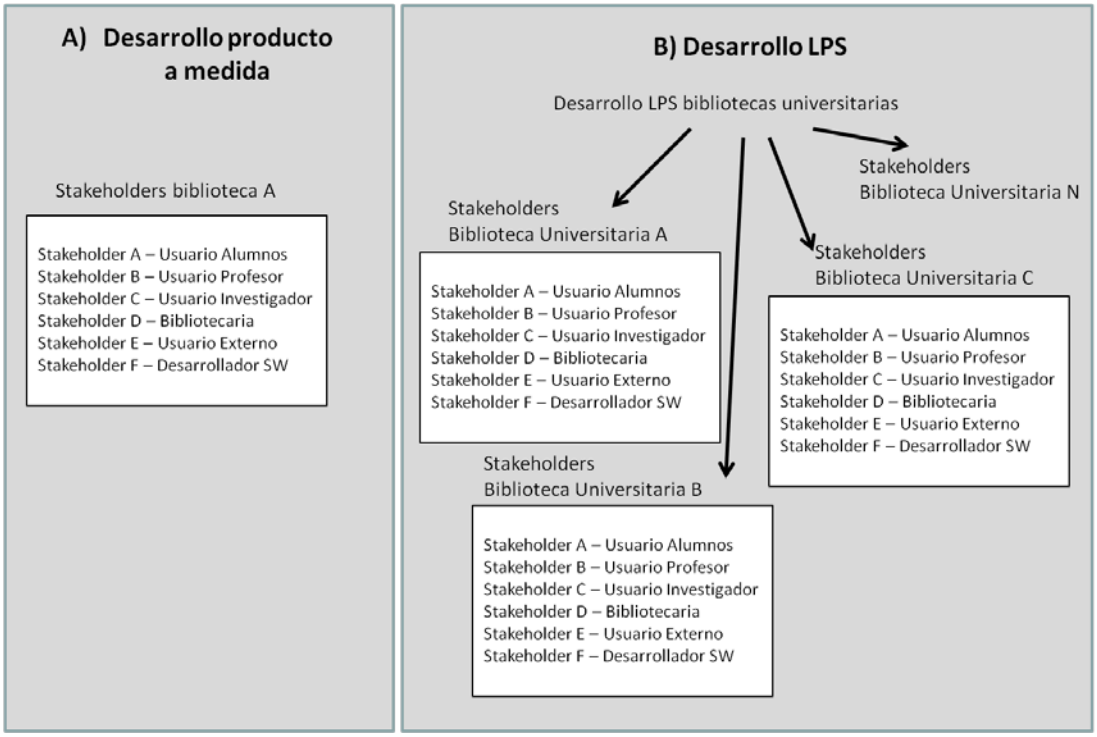


Figura13: Ejemplo de stakeholders y entornos

Así, ELVIRA, identifica N entornos y todos los stakeholders de dichos entornos, para un dominio de aplicación concreto. Con este amplio abanico de stakeholders

involucrados, se cubren todas las vistas o perspectivas sobre el producto. Utilizando como escenario ejemplo, el dominio de una biblioteca universitaria, los entornos serían las bibliotecas de diferentes universidades (biblioteca universitaria A, biblioteca universitaria B,… biblioteca universitaria N,) y los stakeholders serían los usuarios alumnos, los usuarios profesores, los usuarios investigadores, la propia bibliotecaria, usuarios externos, el desarrollador de la aplicación, etc.

Esta agrupación de stakeholders origina como resultado el conjunto de vistas o perspectivas del producto (véase la *figura 14*). Continuando con el ejemplo de una biblioteca universitaria, el conjunto de vistas o perspectivas de producto estaría compuesto por la perspectiva de los usuarios alumnos, la perspectiva de los usuarios profesores, la perspectiva de los usuarios investigadores, la perspectiva de la propia bibliotecaria, la perspectiva de usuarios externos y la perspectiva del desarrollador de la aplicación, etc.

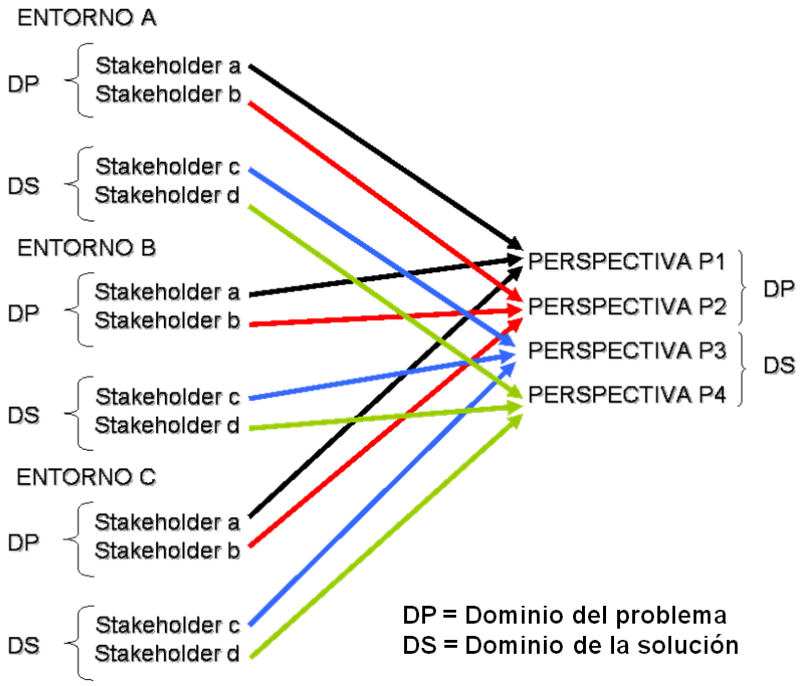


Figura14: Vistas o perspectivas del producto por entorno

Es decir, se analiza el dominio del problema, desde las perspectivas identificadas. Como estamos hablando de un enfoque de linea de productos, las perspectivas se obtienen agrupando stakeholders similares de entornos diferentes (véase la *figura 15*).

Figura15: Vistas del producto

Paso 2: Priorización de las vistas del sistema

La diversidad de stakeholders y de sus expectativas, obliga a una nueva operación, la jerarquización. Se priorizan las vistas o perspectivas identificadas en el paso anterior. Para eso se utiliza un sistema de ponderaciones.

ELVIRA utiliza el siguiente baremo para indicar la relevancia de cada vista

- 9: es una vista fundamental, imprescindible del sistema. La información aportada por esta vista es esencial ya que recoge gran parte de la funcionalidad principal del sistema a crear.
- 6: es una vista bastante influyente del sistema. La información aportada por esta vista proporciona funcionalidades interesantes y relevantes respecto al funcionamiento del sistema a crear.
- 3: es una vista relativamente influyente en el sistema. La información aportada por esta vista no resulta trascendental pero puede proporcionar ideas que optimicen el sistema o lo hagan más sencillo o completo.

ELVIRA realiza esta ponderación, basándose en la influencia que cada vista tiene sobre el producto. En este paso no se consideran los diferentes entornos, se plantea la priorización a nivel de dominio de aplicación, es decir, se priorizan las vistas y no los stakeholders que las conforman. Se considera que los stakeholder, sean del entorno que sean, tendrán todos la misma importancia relativa ante el sistema.

6.2.2 ETAPA 2: OBTENCIÓN DE REQUISITOS POR VISTA

En esta etapa se recogen, analizan y tratan las expectativas de las perspectivas o vistas identificadas en la etapa anterior. Es una etapa de obtención de información.

El objetivo de esta etapa es entender lo QUE cada vista espera del sistema. Se trata de captar la voz de cada vista, interpretarla y traducirla. Se busca información referente a funcionalidades y especificaciones del producto, al mantenimiento, a los

servicios esperados del producto, a futuras ampliaciones, a limitaciones particulares de la perspectiva individual así como sus costumbres, sus métodos de trabajo, las particularidades de su entorno, etc.

Esta etapa está formada por cuatro pasos:

- La obtención individual de la información/conocimiento de los stakeholders de cada vista
- El análisis individual de la información/conocimiento de los stakeholders de cada vista
- La ponderación de los requisitos de los stakeholder de cada vista
- La especificación de los requisitos por vista

Los tres primeros pasos cubren las actividades de ingeniería de requisitos tradicional: se extraen, analizan y priorizan los requisitos para todos y cada uno de los stakeholders identificados en todos los entornos. Estos pasos no tienen por qué realizarse de forma secuencial, normalmente, en la práctica se convierte en un ciclo iterativo. Como resultado de estos tres pasos, se obtienen N documentos de especificación de requisitos (ERS), uno por cada combinación entre stakeholder y entorno (véase la *figura 16*).

En estos ERS la información estará estructurada y priorizada.

ELVIRA, toma como fuente de los requisitos a los stakeholders tanto del dominio del problema como de la solución, ya que entiende que cualquier restricción funcional, normativa, etc., surgirá de alguno de los stakeholders de forma individual durante la obtención de requisitos. Para recoger estas expectativas se emplea el lenguaje natural

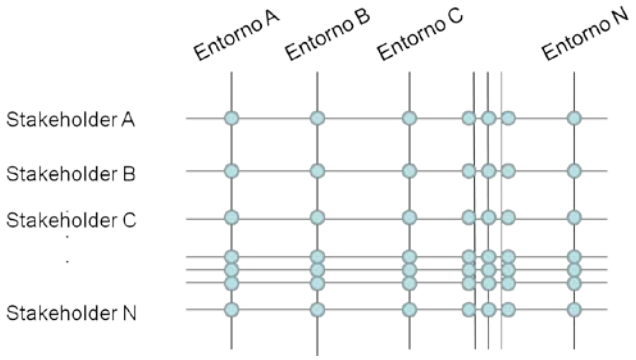


Figura 16: Obtención de requisitos por perspectivas

El último paso de esta etapa especifica los requisitos por vista. Esto lo hace a partir de los ERS obtenidos en los tres pasos anteriores, y utilizando QFD como soporte.

Como resultado se obtiene la lista jerarquizada de expectativas de las diferentes vistas del producto, o lo que tradicionalmente en ingeniería de software se denomina documento de especificación de requisitos (ERS), solo que en este caso se obtiene por cada vista o perspectiva del sistema (véase la *figura 17*).

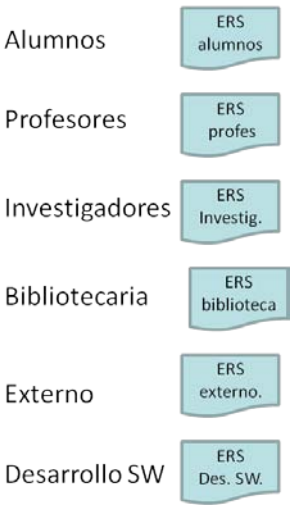


Figura17: ERS por vista.

A continuación se explican de forma detallada los pasos que conforman esta etapa.

Paso 3: Obtención individual de la información/conocimiento de los stakeholders de cada vista

Se obtiene la información de cada uno de los stakeholders de las vistas identificadas en el paso uno del método. ELVIRA no entra en la forma de conseguir esta información, se podrá realizar de cualquiera de las formas tradicionales [Kotonya98], [Sutcliffe02], [Briand00], mediante entrevistas, encuestas, prototipos etc. Lo único exigido es que se recoja la información por stakeholder.

Estamos hablando de una información con un nivel de abstracción alto. Se pretende obtener las necesidades del usuario, las funcionalidades o servicios que espera del producto.

Paso 4 Análisis individual de la información/conocimiento de los stakeholders de cada vista

Como ya se ha comentado, la información es tratada utilizando el lenguaje natural. Esto implica que para identificar los requisitos (necesidades) de los diferentes stakeholders, es necesario analizar y ordenar la información disponible. Normalmente, estos requisitos estarán expresados con diferentes niveles de abstracción y esta etapa puede conducirnos a una estructuración jerárquica, en diferentes niveles, de los requisitos.

El resultado de esta etapa es una lista organizada y estructurada de requisitos de alto nivel, por stakeholder. ELVIRA no obliga a especificar esta información con ningún formato determinado. Sin embargo, si aconseja que sea una lista estructurada, y que los requisitos sean funcionalidades de alto nivel.

Paso 5: Ponderación de los requisitos de los stakeholders de cada vista

Llegados a este paso, tenemos en nuestras manos, el conjunto completo de necesidades de los stakeholders de todos los entornos identificados. Sin embargo, no todas las necesidades tienen la misma importancia de cara a los stakeholders. Por ejemplo para un stakeholder *Usuario Profesor* en el entorno A, puede ser *básico* el requisito R4 y para el mismo tipo de stakeholder (*Usuario Profesor*) en el entorno B, el R4 le puede parecer solo *interesante*.

En este paso se ponderan los requisitos o necesidades, permitiendo así mostrar la importancia de dicha funcionalidad para cada stakeholder de forma independiente.

Como se ha dicho en el capítulo anterior, existen diversos criterios de priorización de requisitos. ELVIRA realiza una priorización basada en el grado de importancia relativa de un requisito para el stakeholder que lo especifica. Para realizar esta priorización, se utiliza el siguiente baremo, que es el proporcionado por QFD.

- Básico (9): es esencial para el sistema. No se puede prescindir de él.
- Interesante (6): es relevante para el sistema pero no imprescindible para su funcionamiento.
- Innecesario pero no negativo (3): no aporta nada a nivel funcional, pero puede aportar mejoras en otros niveles, por ejemplo a nivel de usabilidad, accesibilidad, etc.

- Negativo (0): su existencia es desfavorable y puede perjudicar el funcionamiento del sistema.

Los requisitos básicos son los que resultan imprescindibles en el producto. Deben estar si o si. Los requisitos interesantes son requisitos que se desean pero no son críticos para el funcionamiento eficiente del producto. Los requisitos innecesarios pero no negativos son utilidades que soporta el producto pero que no influyen en su esencia funcional. Los requisitos baremados como negativos, pretenden mostrar aquellas funcionalidades que el stakeholder no desea tener en su producto. En ocasiones los stakeholders tienen muy claro los requisitos que no quieren.

Paso 6: Especificación de los requisitos por vista

Gracias a los pasos anteriores, se ha finalizado la labor de obtención y análisis de la información y conocimiento de todos los stakeholders en todos los entornos. En este paso el objetivo es descubrir y especificar los requisitos por vista o perspectiva. Para ello, se toma como punto de partida la información obtenida en el paso anterior, es decir, la información estructurada y priorizada de las preferencias de todos los stakeholders.

En este paso, se comienza a utilizar la primera matriz de HOQ. En la parte izquierda de la matriz se colocan los stakeholders que conforman una vista. No se consideran diferentes pesos para los stakeholders. En la parte horizontal se van colocando los requisitos explicitados por los stakeholders. Para completar esta parte, es necesario mirar uno por uno los ERS obtenidos entre los pasos 3, 4 y 5, e ir añadiendo como una nueva columna cada requisito nuevo. En el cuerpo de la matriz se irán indicando los impactos stakeholder-requisito con su peso. Es posible que alguna

casilla quede vacía. En ese caso refleja que un determinado stakeholder no ha explicitado ese requisito.

Se aplica la primera matriz HOQ y se obtiene una primera jerarquización de las expectativas de una perspectiva. Esta operación se repite para todas las perspectivas (véase la *figura 18*).

Expectativas de la perspectiva P1	**R1**	**R2**	**R3**	**R4**	**R5**
stakeholder a-entorno A	9	6	6		9
stakeholder a-entorno B	3		3		
stakeholder a-entorno C		6	6	6	6
Peso de las expectativas	**12**	**12**	**15**	**6**	**15**

Expectativas de la perspectiva P2	**R1**	**R2**	**R3**	**R4**	**R5**
stakeholder b-entorno A	3	3	3	3	3
stakeholder b-entorno B	9		6	6	
stakeholder b-entorno C	9	9	9	6	9
Peso de las expectativas	**21**	**12**	**18**	**15**	**12**

Figura18: Jerarquización de las expectativas de una vista o perspectiva

6.2.3 ETAPA 3: DETERMINACIÓN DE REQUISITOS DE LA LPS

En esta etapa el objetivo es responder claramente a la pregunta "*¿Qué vamos a desarrollar?*". Es fundamental satisfacer todas las expectativas del cliente, pero también hay que tener en cuenta las posibilidades y expectativas de la organización.

En esta etapa se hace una distinción entre las expectativas del dominio del problema, y las expectativas del domino de la solución. Esta diferenciación nos viene dada desde la etapa 1 punto 1. En este momento, disponemos de información de las perspectivas de dominio del problema y las perspectivas del dominio de la solución.

La "Determinación de Requisitos de la LPS" es una etapa de toma de decisiones fundamentalmente, está formada por dos pasos (7 y 8), y los datos cuantitativos de la matriz HOQ ayudarán en este proceso.

Como resultado se obtiene la definición de la LPS, pudiendo distinguir entre las partes comunes y variables de la misma.

Paso 7: Determinación de las perspectivas de dominio del problema y del dominio de la solución

El objetivo de este paso es identificar las necesidades del dominio del problema y del dominio de la solución (véase la *figura 19*).

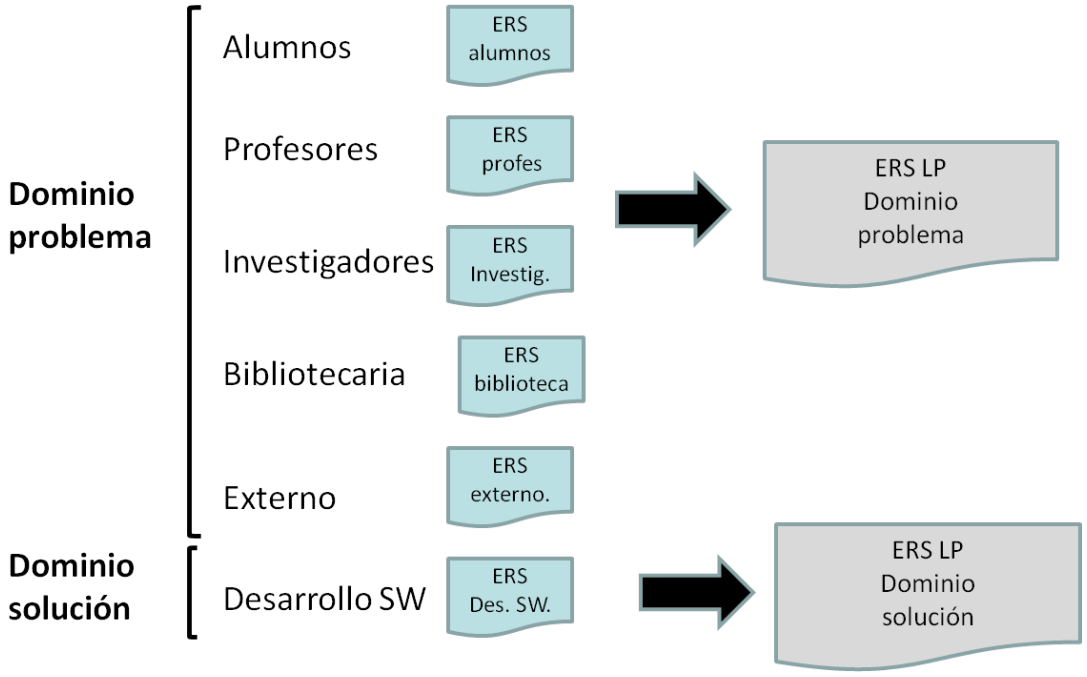


Figura19: ERS del dominio del problema y del dominio de la solución.

Se trata de una segunda vuelta a la primera matriz HOQ, cruzando información de perspectivas en lugar de stakeholders. Es decir, se trabaja con información más general, pero cuantificada.

Se realizarán dos matrices de correlación, una para cada uno de los dominios. En esta ocasión, sí se tiene en cuenta la importancia de la vista o perspectiva respecto del producto, es decir su peso (véase la *figura 20*). Como salida del proceso tendremos dos ERS (tabulados), uno refleja las necesidades del dominio del problema y otro las del dominio de la solución.

Expectativas dominio problema	Peso	R1	R2	R3	R4	R5
Perspectiva P1	9	20	12	18	2	5
Perspectiva P2	6	3	25	3	3	4
Peso de las expectativas		**198**	**258**	**180**	**36**	**69**

Expectativas dominio solucion	Peso	R1	R2	R3	R4	R5
Perspectiva P1	6	9	6	6		9
Perspectiva P2	6	3		3	6	
Peso de las expectativas		**72**	**36**	**54**	**36**	**54**

Figura20: Matrices HOQ por dominio.

Paso 8: Determinación de LPS

Ahora, y siguiendo el método QFD, se nos plantea la pregunta "*¿Cómo realizar esos QUEs? "*

ELVIRA considera que la perspectiva del dominio de la solución es, en definitiva, información del "*cómo*", además considera que se puede obtener esta información de forma independiente a los "*qué*" obtenidos, y luego cruzarla en una matriz *"qué-cómo*" (véase la *figura 21*). Así, en este paso, se realiza una matriz de correlación entre la perspectiva del dominio del problema (eje vertical) y la perspectiva del dominio de la solución (eje horizontal).

Dominio del problema
Cómo

Dominio de la solución
Qué

	RP1	RP2	RP3	RP4	RP…
RS1	1	1		3	
RS2	3	9	9		
RS3	3	1		3	
RS4	9	9	9	9	
RS5	3	3	3		
RS…					

Figura21: Matriz HOQ "qué-cómo".

En la intersección horizontal-vertical de la matriz de correlación se indica para cada uno de los *qué* el *cómo* correspondiente. Esta relación también puede ponderarse. Así se podrá indicar si es una relación

- Fuerte: 9 puntos. Está claramente definido cómo dar solución a la necesidad planteada.
- Media: 3 puntos: Esta parcialmente definido como dar solución a la necesidad planteada.
- Débil: 1 punto. Es una relación con poca fuerza. No está definido como dar solución a la necesidad planteada.

Como resultado obtendremos una jerarquización de los requisitos del dominio del problema, basada en los requisitos del dominio de la solución. Este resultado se puede representar gráficamente.

Con esta información es posible determinar los requisitos comunes y variables de la LPS.

6.3 RESULTADO DE ELVIRA

El producto resultado de la actividad de IR es un documento de especificación de requisitos [Sommerville97]. Así, ELVIRA proporciona como resultado una lista de requisitos priorizada, de la cual sabemos además qué requisito es común y cuál variable.

El ERS proporcionado por ELVIRA surge de la integración de las perspectivas de los diversos stakeholders y de este modo recoge las necesidades de todos ellos.

Hay que señalar además que ELVIRA permite adaptar la información proporcionada a todo tipo de formato, pudiendo así integrarse en estándares como el DoD DI-MCCR-80025ª, el IEEE ANSI 830-1984, o adaptarse a cualquier estilo definido.

CAPÍTULO 7 - VALIDACIÓN

En este capítulo mostramos los medios empleados para validar la hipótesis planteada en esta investigación. En los siguientes apartados analizamos la hipótesis y seleccionamos las estrategias de validación más apropiadas, describiendo las actividades y experimentos realizados, y mostrando los resultados obtenidos.

7.1 ESTRATEGIA DE VALIDACIÓN DE LA HIPÓTESIS

Entendemos, que ELVIRA se debe validar desde una doble perspectiva. Por un lado, debemos comprobar que se puede usar en las fases iniciales de definición de producto, y que permite integrar diferentes vistas o perspectivas, es decir, se valida la filosofía fundamental del método. Por otro lado, debemos validar la utilidad del método ELVIRA en sí mismo, es decir el resultado que proporciona debe ser adecuado a las entradas al mismo.

Este enfoque de validación está ligado directamente con la hipótesis planteada en el apartado 1.4: "*Es posible integrar y utilizar el conocimiento de fuentes diversas en un método sistemático para la definición de una línea de productos software que dé valor al usuario, en las fases iniciales del desarrollo de software".*

Destacar que la técnica de validación de experimentos formales y los resultados obtenidos al proceso aplicado han sido publicados en los congresos JISBD 2007 y VASOP 2010 respectivamente [Ereño07], [Ereño10].

7.2 DISEÑO Y EJECUCIÓN DEL EXPERIMENTO

Para poder obtener resultados útiles y significativos de un experimento, es necesario definirlo y ejecutarlo de forma sistemática. En este trabajo de investigación hemos seguido el proceso experimental detallado en [Wohlin00], el cual detallamos en los siguientes apartados.

7.2.1 CONTEXTO

Los participantes en el experimento debían elaborar un ERS_LP sobre un dominio determinado. Para esta labor, debían simular las necesidades de los stakeholders de ese sistema, es decir, no existían clientes reales en el contexto del experimento. Por ese motivo, se eligió un dominio que resultaba familiar a los participantes. Concretamente se trataba de un portal dirigido a estudiantes donde se ofrecían todo tipo de servicios interesantes en un entorno universitario.

Para el desarrollo de este experimento se contó con la participación de alumnos de 4º de Ingeniería Informática. Todos ellos tenían conocimientos similares para la realización del experimento.

Se crearon dos grupos de forma aleatoria (GA y GB). Cada grupo constaba de 15 participantes. El grupo GA aplicó el método ELVIRA para la elaboración del ERS_LP, mientras que el grupo GB, utilizó lo que hemos denominado método tradicional o ad-hoc. Dentro de cada grupo se formaron equipos de dos-tres personas, ya que entendíamos que la IR no es un trabajo individual, y que equipos formados por más de una persona favorecían la discusión y simulaban mejor un entorno real. Se crearon así 5 equipos dentro de cada grupo (EA1, EA2, EA3, EA4, EA5, EB1, EB2, EB3,

etcEB4 y EB5). En la *figura 22* se puede ver de forma gráfica la organización de grupos y equipos, es decir la estructura de participantes en el experimento.

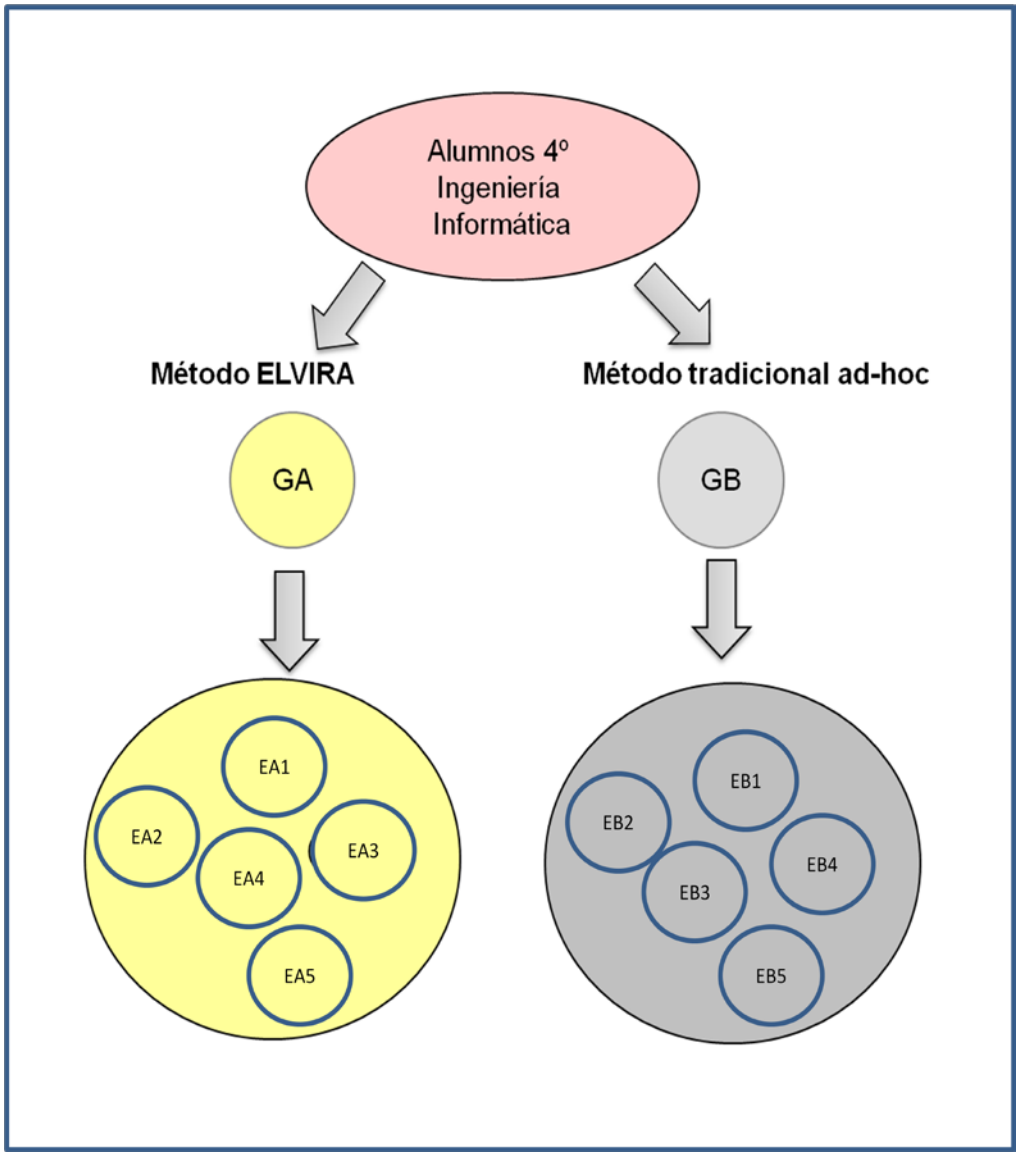


Figura 22: Estructura de participantes en el experimento

Con el objetivo de conseguir una buena motivación de los participantes en el experimento, se planteó como una práctica asociada a la asignatura "Ingeniería del Software", de tal modo que su realización puntuaba en la nota final de la asignatura. Ninguno de los participantes conocía de antemano que se trataba de un experimento.

7.2.2 DOCUMENTACIÓN

Tradicionalmente se ha prestado poca atención a la documentación que se genera a lo largo del ciclo de vida del software, lo que ha dado lugar a diversos problemas como textos mal redactados, difíciles de entender, obsoletos, incompletos, etc.

Todo esto, además de dificultar enormemente el mantenimiento del software, degrada su utilidad al no disponer de una correcta información sobre todas las posibilidades que ofrecen los productos de software construidos.

Para corroborar la hipótesis planteada es necesario comparar los ERS_LP elaborados por los grupos GA y GB. Con el fin de que los resultados fueran comparables se elaboró una plantilla para el ERS_LP [ver anexo A]. Aunque son varios los formatos y las indicaciones sobre el contenido correcto de un documento de especificación de requisitos, se optó por diseñar una plantilla de ERS_LP sencilla, que no generara dudas. De este modo se primó el contenido y la originalidad. La única directriz dada a los equipos fue la utilización de un lenguaje claro y conciso.

7.2.3 VARIABLES

La variable independiente en este experimento ha sido la elección del método para el desarrollo del ERS_LP. Se trata una variable nominal que puede tomar dos valores: método ELVIRA o método ad-hoc.

Nuestro objetivo ha sido medir cuál de los dos métodos ofrece un resultado más completo. El término "completo" en este caso hace referencia al documento de especificación de requisitos en su totalidad.

Según [Costello95] la completitud de la especificación de requisitos comprende los siguientes aspectos:

- Que todo lo necesario esté en la especificación de requisitos.
- Que el nivel de desglose de los requisitos sea apropiado.
- Que no aparezcan elementos del tipo AD (A definir).

Estos autores [Costello95] también afirman que el primer y segundo aspecto se puede comprobar mediante inspecciones de las especificaciones de requisitos. De este modo proponen medidas como por ejemplo, contar el número de requisitos de un nivel que sirven para detallar alguno de un nivel superior.

Teniendo en cuenta lo ya comentado y que estamos hablando de una LPS, donde es fundamental la variabilidad, hemos definido las siguientes variables dependientes:

- Número total de requisitos. Esta variable recoge las necesidades de cada stakeholder.
- Número de requisitos comunes. Son los requisitos a implementar en todos los productos de la línea. Se pueden considerar como la esencia funcional del producto a desarrollar.

- Número de requisitos variables. Se ha denominado así a aquellas funcionalidades a elegir a la hora de adquirir el producto. Normalmente representan necesidades particulares de ciertos productos dentro de la linea de productos. A estos requisitos también se les denomina requisitos opcionales.
- Numero de variantes. Se ha denominado así a aquellos requisitos, tanto comunes como variables que permiten opciones sobre su puesta en práctica.

7.2.4 PREPARACIÓN DE LA EJECUCIÓN

Todos los participantes tenían conocimientos similares en las áreas de IR, orientación a vistas y LPS, sin embargo antes de la realización del experimento y de forma independiente al mismo, se les plantearon dos ejercicios cuyo objetivo era prepararles en cierta tareas y hacerles reflexionar.

El primer ejercicio consistió en el desarrollo de un ERS para un sistema software utilizando la estructura propuesta por el Estándar IEEE-830. Con este ejercicio se buscaba lograr cierta agilidad en la tarea de completar un ERS con una estructura compleja, como la propuesta.

El segundo ejercicio consistió en la identificación de las posibles vistas o perspectivas para un sistema software. Con este ejercicio se buscaba hacer evidente la necesidad de participación de varios stakeholders en el proceso de IR.

Para la realización del experimento se realizó una sesión previa de formación de dos horas al grupo GA en el método ELVIRA. Con esto pretendíamos evitar la comunicación entre los diversos participantes, y que se generará algún tipo de suspicacia.

7.2.5 EJECUCIÓN DEL EXPERIMENTO

El experimento duró 4 horas. Cada grupo ocupó un aula diferente. El grupo A estuvo en todo momento asistido por un experto en IR y en ELVIRA, que atendió única y exclusivamente las dudas que surgieron sobre el método. El grupo B estuvo asistido por un experto en IR y desconocedor del método ELVIRA. Aunque estuvieron acompañados por expertos, la actitud de éstos fue la de no contestar a ninguna pregunta relacionada con la solución al problema planteado. Su presencia estaba justificada para dotar de un carácter formal al experimento, y en el caso del grupo A para atender dudas sobre el método.

Al comienzo de la sesión se les entregó a ambos grupos el mismo enunciado y el mismo formato de ERS_LP. Ambos documentos pueden encontrarse en los anexos A y B respectivamente.

7.2.6 VALIDEZ DEL EXPERIMENTO

Además de diseñar y realizar el experimento bajo un proceso sistemático [Wohlin00], para evitar las posibles amenazas a la validez del mismo, se tomaron las siguientes medidas:

- Todos los participantes en el experimento tenían una experiencia y conocimientos similares.
- El dominio de experimentación era familiar a todos los participantes.
- El documento a generar fue diseñado primando la facilidad de cumplimentación, de tal modo que los participantes dedicaran sus esfuerzos a la elaboración correcta del contenido y no tuvieran problemas con el formato.

- Los participantes no eran conscientes de estar involucrados en un experimento. Esto y el hecho de obtener cierta puntuación en una asignatura relacionada, aseguraron una buena disposición.
- No se permitió que los participantes hablaran entre ellos desde el momento que un grupo recibió la formación necesaria para el experimento, hasta la finalización del mismo.

7.3 VALIDACIÓN DEL MÉTODO

Para validar la primera parte de la hipótesis "*Es posible integrar y utilizar el conocimiento de fuentes diversas en un método sistemático para la definición de una línea de productos software….*" hemos analizado e interpretado los resultados del experimento buscando la eficacia del método ELVIRA, frente al método ad-hoc. Cuando se hace referencia al método ad-hoc, se piensa en la situación, más que típica, en la cual el ingeniero software habla con diversos stakeholders (los que considera necesario) y utiliza su intuición y experiencia en el desarrollo de un ERS_LP.

Para medir la eficacia del método ELVIRA hemos elegido el atributo completitud (explicado en el apartado 7.2.3). De este modo hemos analizado las variables dependientes identificadas respecto a este atributo, y hemos comparado su comportamiento respecto a la variable dependiente. Es decir hemos comparado si el resultado de los grupos con método, medido en unidades de requisitos descubiertos, es más completo o no que el resultado de los grupos sin método. La respuesta a esta cuestión validará o refutará la primera parte de la hipótesis.

Previamente al análisis cuantitativo de los requisitos hemos realizado un análisis cualitativo de los mismos. Este análisis es imprescindible, teniendo en cuenta que

para realizar la especificación de requisitos se ha utilizado el lenguaje natural. El análisis cualitativo ha permitido entender el significado de cada requisito, identificar los requisitos erróneos, y de este modo eliminarlos y no contabilizarlos.

Al realizar este análisis cualitativo nos hemos percatado de lo siguiente: en el diseño del experimento se habían identificado como variables dependientes, entre otras, requisitos variables y variantes. Sin embargo, al analizar los datos nos ha resultado imposible realizar esta diferenciación. La plantilla propuesta como resultado solo hacía referencia a requisitos variables y los participantes en el experimento utilizaron este apartado para ambos tipos de requisitos. Por ese motivo, hemos decidido contabilizar todos los impactos en este apartado como requisitos variables.

También hemos comparado cualitativamente los requisitos obtenidos por los equipos con método y sin método. El resultado muestra que no hay ningún requisito identificado por el grupo sin método que no haya sido también identificado por el grupo con método. Se puede afirmar de este modo que el grupo que utiliza el método ELVIRA ofrece un ERS más completo, desde un punto de vista cualitativo.

7.3.1 ANÁLISIS DE LOS RESULTADOS

En la *tabla 7* se pueden ver los datos obtenidos tras realizar el experimento. En la parte izquierda, y encabezando cada fila podemos identificar a los dos grupos participantes en el experimento (el grupo A que ha aplicado el método ELVIRA y el grupo B que ha aplicado un método ad-hoc) y a los 5 equipos que conforman cada uno de los grupos (EA1, EA2,…, EA5, EB1, EB2,…, EB5). Las columnas representan las variables dependientes identificadas en el experimento: requisitos totales, requisitos comunes y requisitos variables. La intersección fila-columna muestra la

cantidad de requisitos identificados por cada equipo. También se pueden ver en la tabla los totales calculados para cada tipo de requisito. Esta tabla muestra los datos brutos a partir de los cuales se ha realizado el análisis de resultados.

Tanto en la *tabla 7*, como en las figuras utilizadas para mostrar los resultados en este apartado, se han utilizado los colores granate y azul para diferenciar a los grupos participantes. Así, para representar al grupo que aplica el método ELVIRA se ha utilizado el color granate, mientras que para representar al grupo que aplica el método ad-hoc se ha utilizado el color azul. De este modo resulta más sencillo realizar el contraste de resultados.

Si comparamos a un nivel global los requisitos totales especificados por ambos grupos (GA y GB) (véase la *figura 23*), se aprecia claramente que el grupo que utiliza el método ELVIRA ofrece un ERS más completo, con mayor número de requisitos, que el ofrecido por el grupo con el método ad-hoc. De hecho el grupo A produce más del doble de resultados que el grupo B (306/127=2.4).

	Equipos	Requisitos Totales	Requisitos Comunes	Requisitos Variables
Grupo A Método ELVIRA	EA1	92	20	72
	EA2	45	33	12
	EA3	70	44	26
	EA4	42	15	27
	EA5	57	24	33
	Totales	**306**	**136**	**170**
Grupo B Método Ad-hoc	EB1	18	5	13
	EB3	23	14	9
	EB4	21	9	12
	EB5	34	16	18
	EB7	31	7	24
	Totales	**127**	**51**	**76**

Tabla7: Datos obtenidos en el experimento

La *figura 23* está realizada a partir de los datos en bruto. Para obtener un valor más representativo del número de requisitos promedio generado por cada grupo, hemos calculado el promedio aritmético a partir de los datos en bruto (véase las *figuras 23 y 24*). Este dato indica que el grupo que utiliza el método ELVIRA genera un promedio de 61,2 requisitos mientras que el grupo con el método ad-hoc genera un promedio de 25,4 requisitos

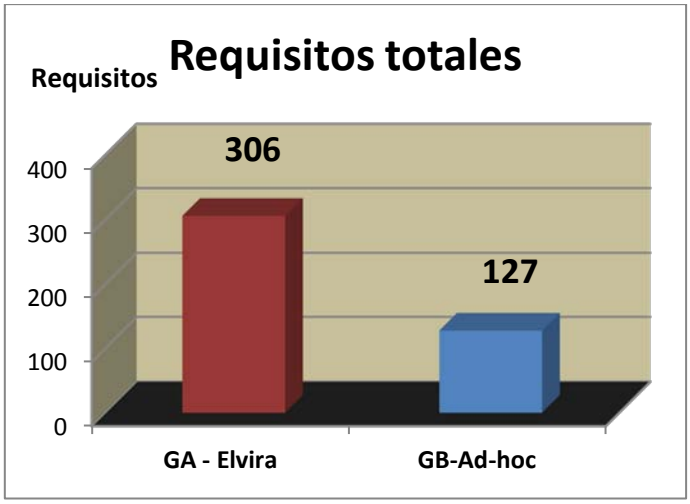


Figura 23: Requisitos totales por grupo

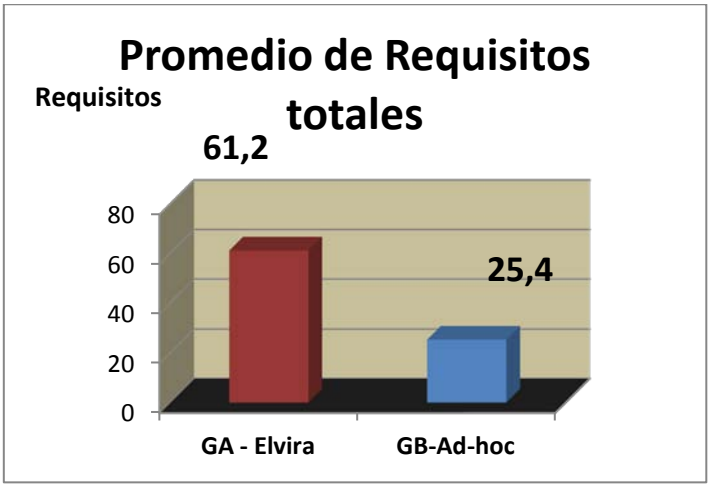


Figura 24: Promedio de requisitos por grupo

Las figuras 23 y 24 muestran los datos globales a nivel de grupo, pero es interesante comprobar lo que ha sucedido a nivel individual, en cada uno de los equipos. La *figura 25* muestra esta información representando en el eje vertical la cantidad de requisitos y en el eje horizontal los equipos. Con color granate se representan los equipos que han aplicado el método ELVIRA y en color azul los equipos que han aplicado el método ad-hoc. Se pueden ver los resultados por equipo agrupados en dos líneas enfrentadas de tal manera que a simple golpe de vista se ve la diferencia entre los grupos. De nuevo se observa que los equipos que han aplicado el método ELVIRA han generado mayor número de requisitos, es decir han producido un resultado más completo, en comparación a los equipos con el método ad-hoc. Hay que señalar que incluso el mínimo resultado de los equipos que han aplicado el método ELVIRA, supera al máximo resultado de los equipos sin método (42 requisitos frente a 34 requisitos), como se aprecia en la *figura 26*.

Figura25: Requisitos totales por equipo

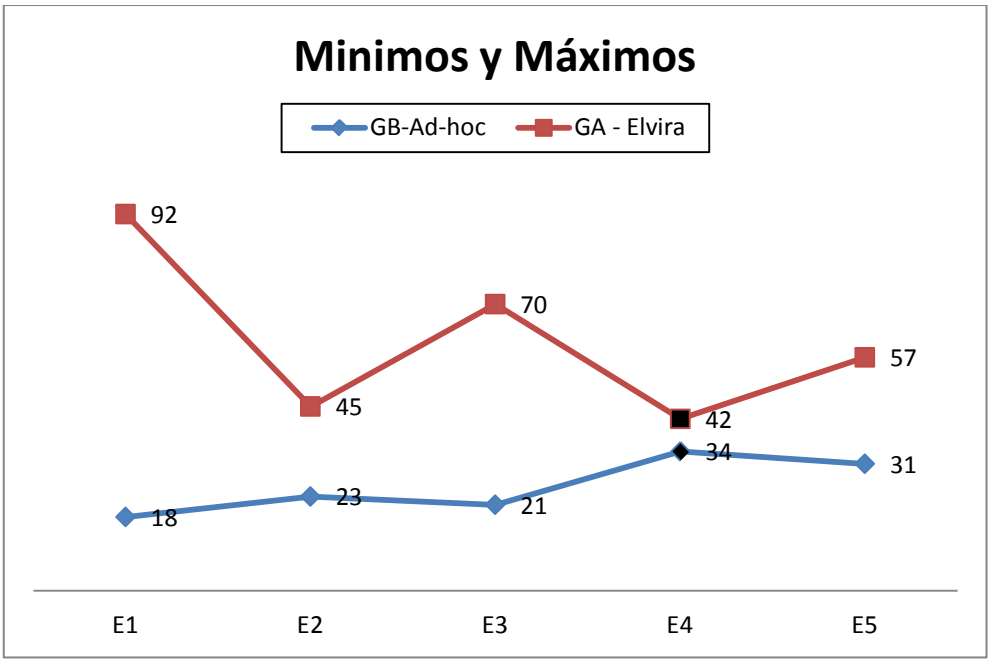


Figura 26: Requisitos mínimos y máximos

Es posible entrar en mayor detalle en el análisis de los datos. Recordemos que el experimento consiste en producir la especificación de requisitos para una línea de productos software. Esto significa que como resultado se genera un conjunto de requisitos clasificados en requisitos comunes y requisitos variables. En las *figuras 27 y 28* se puede ver esta información clasificada por grupos, tanto con datos brutos como con promedios. En ambos casos obtienen resultados más completos los grupos que aplican ELVIRA.

Desplegando la información de requisitos comunes y variables a nivel de equipo se sigue manteniendo la tendencia de resultados. En la *figura 29* se muestran los datos desglosados en equipos para los requisitos comunes y en la *figura 30* los datos para requisitos variables. En estas figuras se ha seguido el mismo formato de datos que en las figuras anteriores.

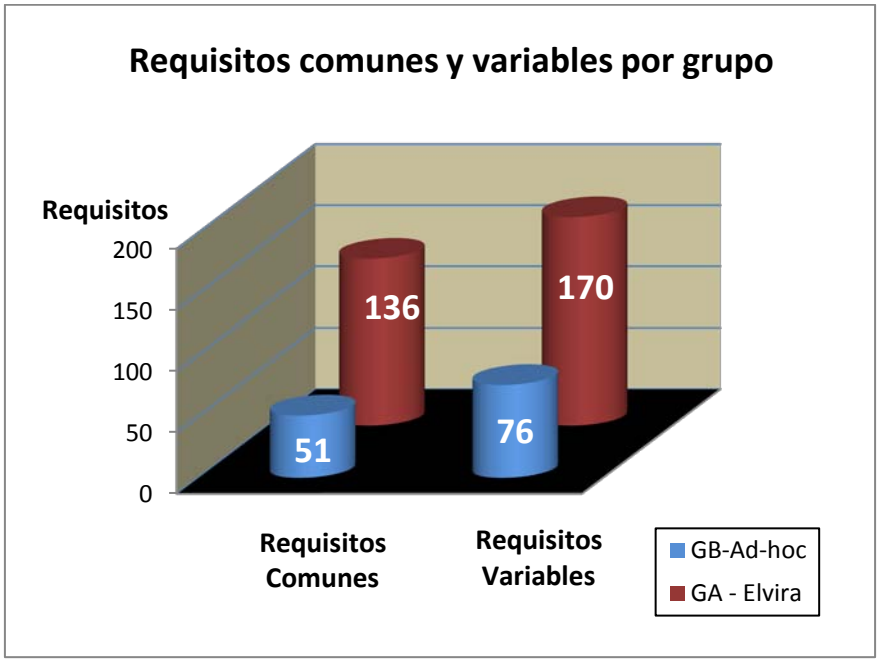


Figura 27: Requisitos comunes y variables por grupo

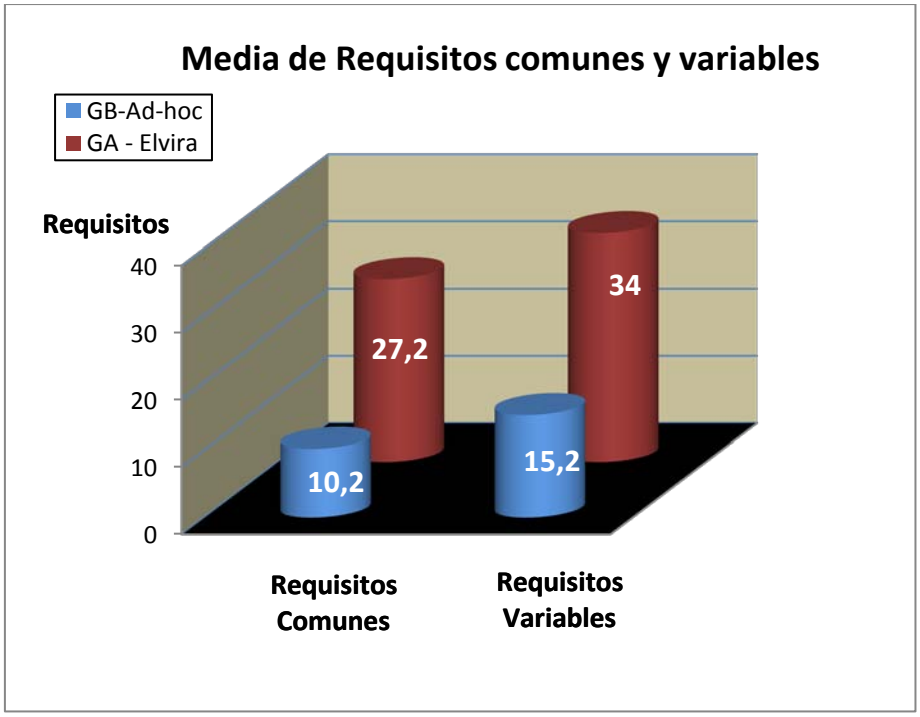


Figura28: Media de requisitos comunes y variables por grupo

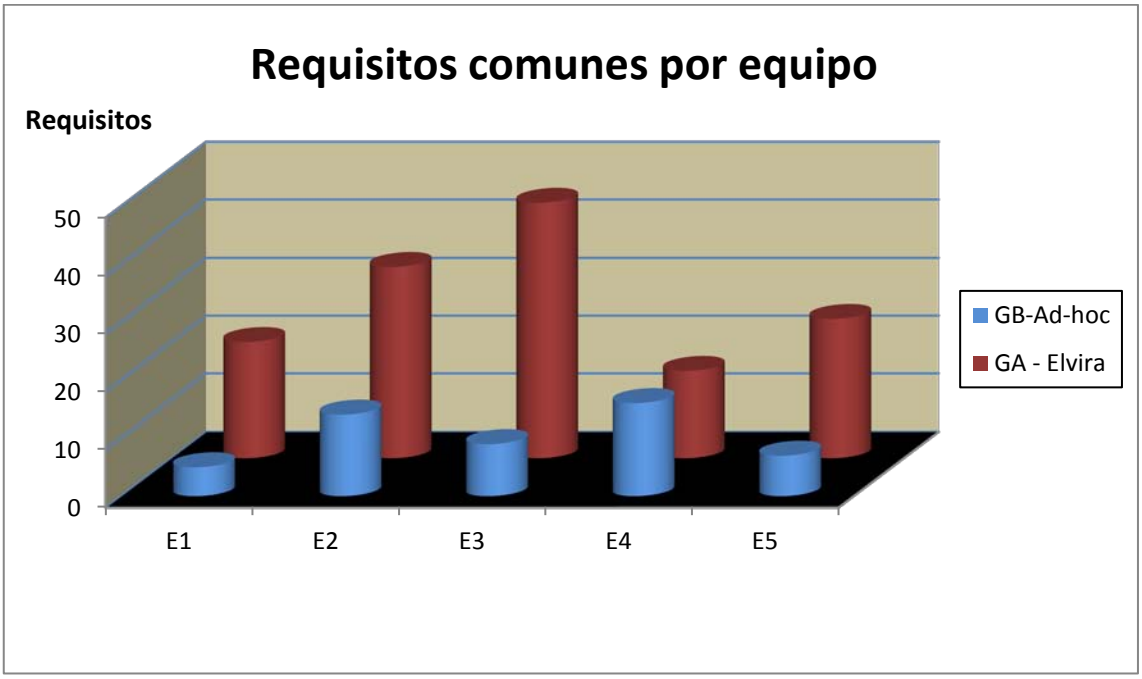


Figura 29: Requisitos comunes por equipo

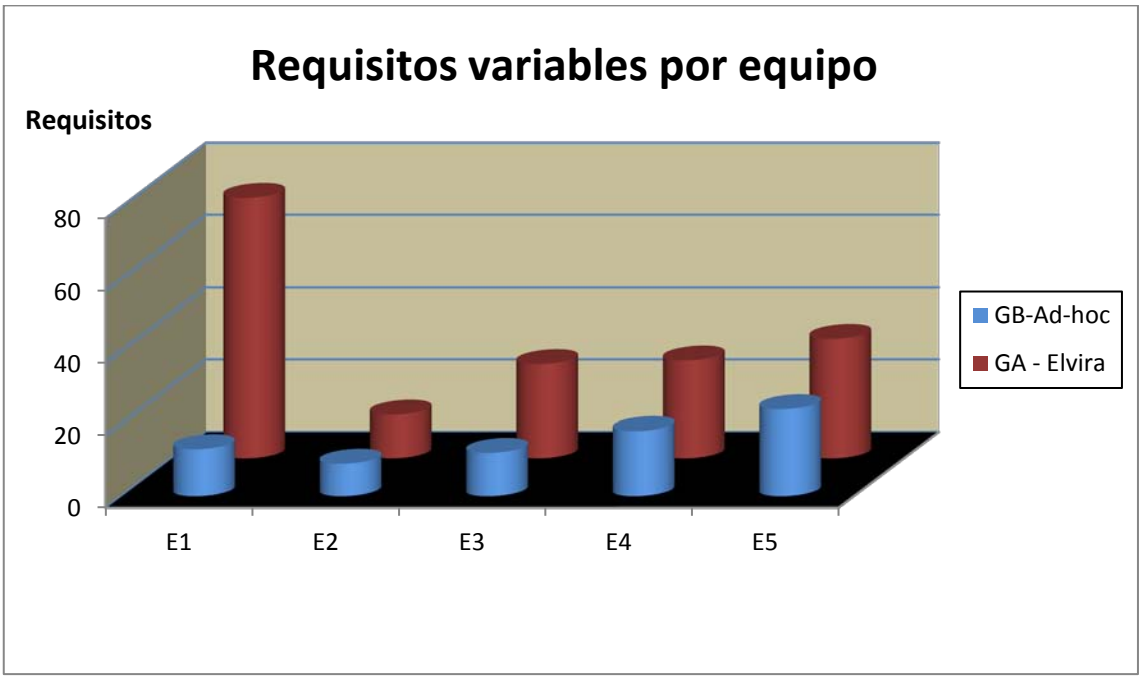


Figura 30: Requisitos variables por equipo

Hemos analizado los datos a diferentes niveles de profundidad, desde los datos globales a nivel de grupo hasta por los datos a nivel de equipo. Así mismo se han desglosado los requisitos totales en requisitos comunes y variables. En todos los casos el resultado es el mismo. Los participantes que han utilizado el método ELVIRA obtienen mayor número de requisitos, es decir, ERS más completos, frente a los participantes que han utilizado el método ad-hoc.

La idea de resultado "más completo" queda muy bien reflejada en la *figura 31*. Mediante un diagrama de área, se muestra la cobertura proporcionada por los grupos participantes en unidades de requisitos. En color azul se muestra el área (el espacio de requisitos) cubierto por el método ad-hoc. En color granate se muestra el área o espacio de requisitos cubierto por el método ELVIRA. Claramente el método ELVIRA proporciona un resultado más completo.

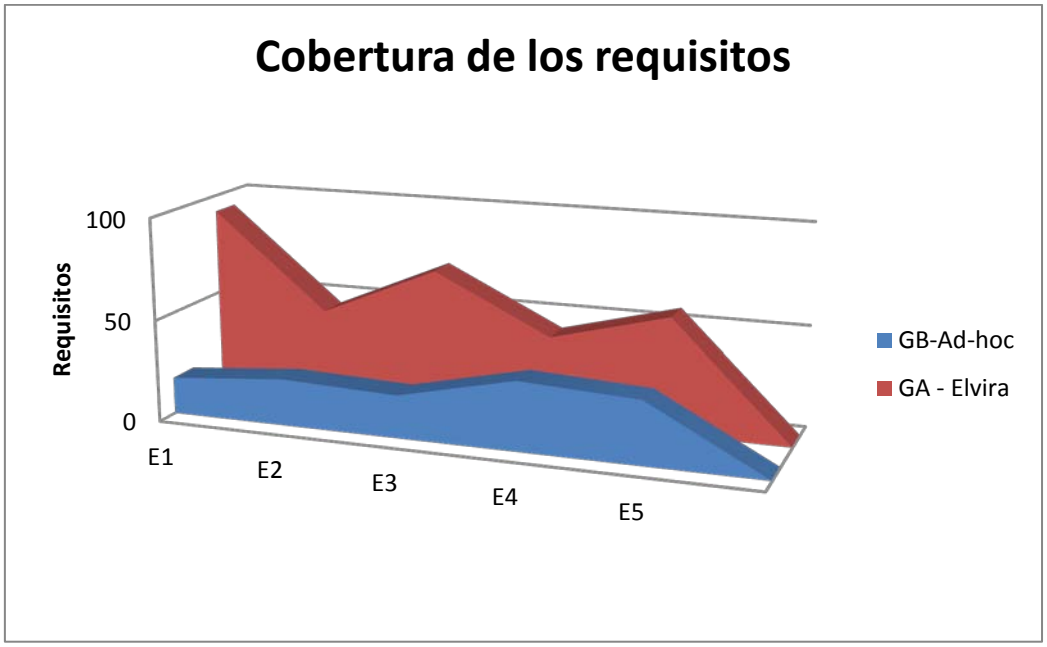


Figura 31: Cobertura de los requisitos

Ante estos resultados se puede ratificar la primera parte de la hipótesis y decir que el método ELVIRA es eficaz, permite obtener ERS_LP más completos.

Como curiosidad, queremos indicar que la proporción entre requisitos comunes y variables fue similar, independientemente del método utilizado (véase la figura32).

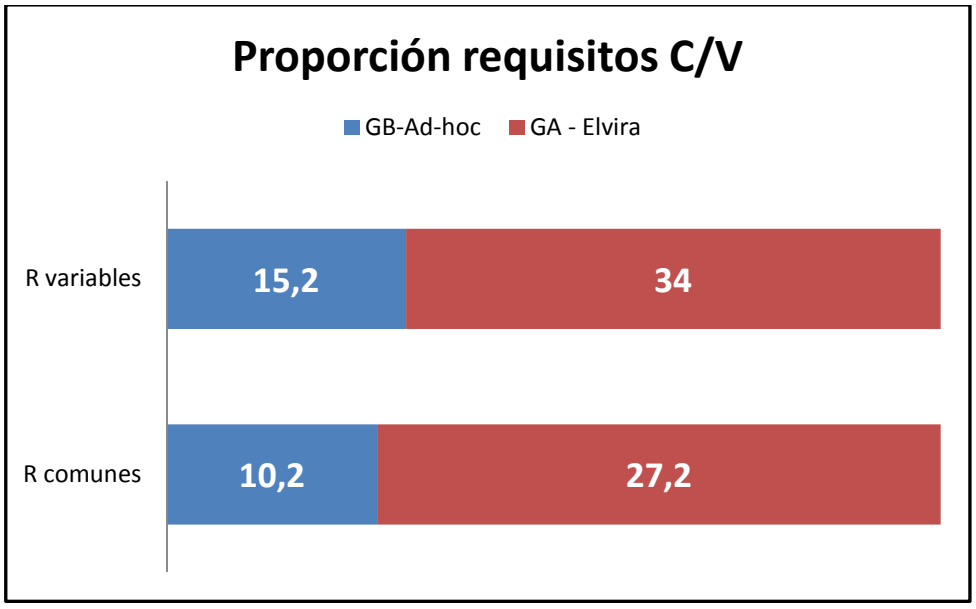


Figura 32: Proporción entre requisitos comunes y variables

Se ha llegado a otras conclusiones cualitativas, mediante la observación de diversas situaciones y actitudes surgidas durante el experimento, que consideramos interesantes:

- El Grupo sin método:
  - Todos los equipos tardaron en empezar. En una encuesta realizada posteriormente todos confirmaron esta actitud. Sus palabras fueron que les daba miedo empezar y echaban de menos un método formal.
  - Todos los equipos terminaron la elaboración de la plantilla.

- No utilizaron sus conocimientos. No aplicaron las vistas.

- El Grupo con método:
  - Comenzaron a trabajar desde el momento inicial. Surgieron algunas dudas sobre el método, pero ninguna sobre el dominio ni el objetivo del ejercicio.
  - No todos los equipos tuvieron tiempo de completar la plantilla, sin embargo todos finalizaron la aplicación del método.

7.4 VALIDACIÓN DE LA UTILIDAD DEL MÉTODO

Otra de las cuestiones que hemos querido evaluar es la utilidad del método per sé. Como dice Montgomery [Montgomery96] "*la calidad es el grado hasta el cual los productos satisfacen las necesidades de la gente que los usa"*. Hemos querido comprobar esto en las fases iniciales de definición de producto, ya que no tendría ningún sentido hacerlo a posteriori. Hablamos así de una validación temprana de los requisitos. Este objetivo es trazable con la segunda parte de la hipótesis: "*Es posible integrar y utilizar el conocimiento de fuentes diversas en un* ***método*** *sistemático para la definición de una línea de productos software que* ***de valor al usuario, en las fases iniciales del desarrollo de software*****".**

En los siguientes apartados explicamos la forma en la que tradicionalmente se ha medido el concepto "*valor*" en la ingeniería del software y se presenta un nuevo enfoque para medir este indicador; concretamente el uso de una métrica del contexto de medios sociales (Social Media), el indicador IOR (Impact of Relationships), para determinar el valor de un producto en el contexto de una LPS.

7.4.1 EL CONCEPTO "VALOR" EN LA INGENIERÍA DE SOFTWARE

La ingeniería de software basada en valor o ISBV se fundamenta en la implicación de los stakeholders en todos los niveles del desarrollo, ya que sus necesidades y preferencias individuales determinan el éxito del proyecto y el valor del producto [Piattini07], [Piattini08]. Centrándonos en la fase de ingeniería de requisitos, la ISBV establece que no todas las funcionalidades (y por extensión, requisitos, componentes o casos prueba, entre otros) tienen igual importancia, ya que algunas de ellas aportan más "valor" que otras. Por lo tanto, es necesario proveer de mecanismos que gestionen esta diferenciación de los distintos artefactos software [Bohem05]. De este modo, se define "valor" como la cuantificación de la importancia que un determinado artefacto o tarea tiene para todos los implicados (o stakeholders) en un sistema [Bohem05].

Está claro que la cuantificación de todos estos tipos de beneficios es algo complejo e incluso subjetivo [Halling04]. Quizás este es el motivo de que la mayoría de los trabajos realizados utilicen el costo como la medida de valor de los productos. Así, el "*retorno de inversión"* (ROI-*Return on Investment*) es el indicador más comúnmente utilizado para medir el valor de los productos. Sin embargo, este indicador está muy orientado al grupo de desarrollo, y se centra sobre todo en los costes de desarrollo y el retorno obtenido, mientras que las relaciones con los stakeholders se basan en conversaciones y no en dinero.

En situaciones como éstas, donde el retorno de la inversión no se traduce de forma directa en ventas, surge la necesidad de otros indicadores que cuantifiquen la influencia de los stakeholders en los productos desarrollados para justificar la inversión. El foco de nuestra validación se basa en el uso de la métrica IOR (Impact of Relationships), para determinar el valor de un producto en el contexto de una LPS.

El indicador IOR tiene su origen en el contexto *Social Media*. La forma más sencilla de definir dicho contexto es hablar de un nuevo entorno social compuesto por varios medios online que facilitan las relaciones, la comunicación y la interacción entre usuarios además de posibilitar la generación y la valoración de contenido y permitir compartirlos de manera sencilla y sin la necesidad de conocimientos técnicos avanzados [Kaplan10]. Los *Medios Sociales* constituyen un nuevo entorno de comunicación social online. Es un fenómeno en alza, que ha cambiado la forma en que los usuarios se comunican, comparten experiencias y generan contenidos [Celaya08].

La traducción de *Social Media* en español es *Medios Sociales*, no Redes Sociales como se dice normalmente. Por lo tanto, no estamos hablando de una nueva realidad social ni posiblemente de una nueva estrategia de relación, sino de nuevos entornos que favorecen las formas de relación tradicionales donde la conversación, la comunicación y el intercambio de conocimiento en todas sus formas, se manifiestan con mayor intensidad.

Las relaciones son la nueva moneda de rentabilidad. Se pasa así, de una economía de mercado a una economía de las relaciones. Las nuevas fórmulas y modelos de comunicación han obligado a un análisis del impacto y el valor generado gracias a las interacciones de la marca con los usuarios. De esta manera, el indicador IOR es un indicador de la eficacia de las acciones de comunicación empresarial [Peña13], [Lamas10] y permite medir la interacción de los stakeholders y el impacto de esa interacción en el producto final [Castello12].

Podemos concluir este aparatado con los dos axiomas siguientes:

- La principal medida del éxito de un sistema software, es el grado con el cual éste cumple los objetivos para los que fue creado.
- El valor de los productos es función de las preferencias de los stakeholders.

7.4.2 IDENTIFICACIÓN DE MÉTRICAS

Como hemos comentado, el IOR permite medir las *interacciones* que los participantes tienen en su *comunidad* y cómo esta relación *impacta* en el producto.

En el contexto de líneas de producto software, hemos entendido la *comunidad* como el grupo de stakeholders, las *interacciones* como el proceso de ingeniería de requisitos, y el *impacto* como la influencia de las preferencias de los stakeholders en el producto final.

Estas relaciones se cuantifican en base a cinco métricas:

- **Valor Esperado de Cliente (VEC):** Conjunto de requisitos que conforman la perspectiva de un stakeholder concreto en un contexto concreto.
- **Valor Línea de Producto (VLP):** Requisitos o características que ofrece la LPS.
- **Valor Real (VR):** Valor que ofrece la LPS a un stakeholder concreto en un contexto concreto
- **Valor Potencial (VP):** Valor que puede ofrecer la LPS a un stakeholder concreto en un contexto concreto
- **Valor No Cubierto (VNC):** Requisitos específicos de un stakeholder, no cubiertos por la LPS

En la *figura 33* se puede ver de forma gráfica la relación entre todas ellas.

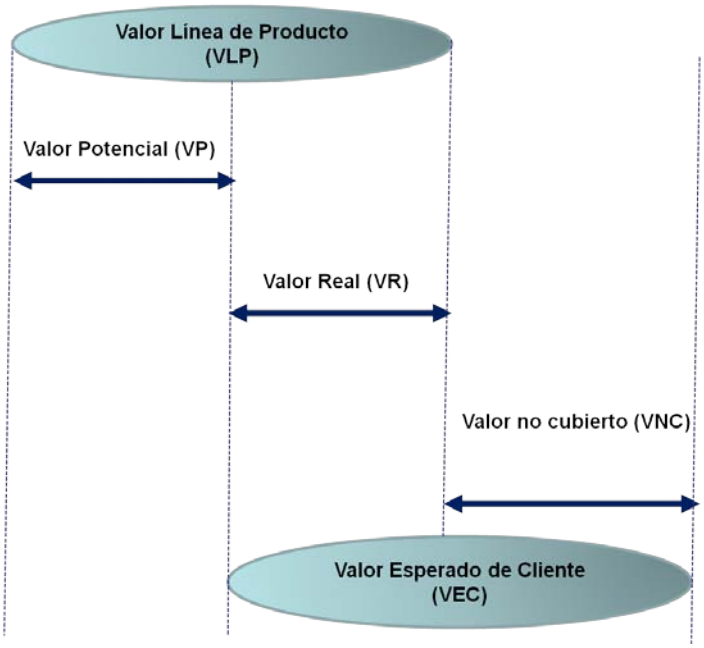


Figura33: Métricas del IOR.

7.4.3 APLICACIÓN DE MÉTRICAS

El proceso de ingeniería de requisitos para una línea de productos software plantea un escenario donde diferentes stakeholders en diferentes contextos comunican sus preferencias sobre el futuro producto (véase la *figura 34*).

Si queremos determinar el valor que tiene para cada stakeholder el producto definido, es necesario confrontar los requisitos de cada vista, con los requisitos de la LPS (salida del método). Y es ahí donde hemos aplicado las métricas definidas; Hemos aplicado el conjunto de métricas definidas sobre los resultados producidos por cuatro equipos del grupo A, concretamente los equipos EA2, EA3, EA4 y EA5, que utilizaron el método ELVIRA en el transcurso del experimento (hemos excluido el equipo EA1 ya que no finalizaron la aplicación del método y por lo tanto no utilizamos sus resultados en esta parte de la validación).

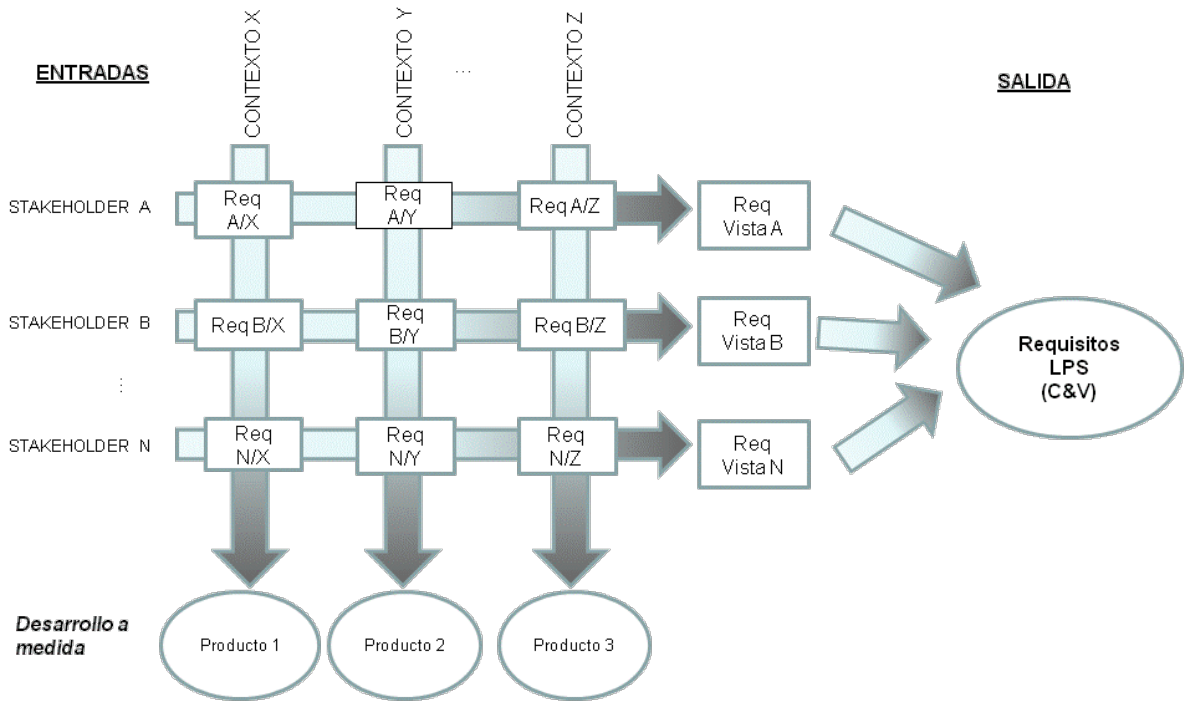


Figura34: Proceso de Ingeniería de Requisitos para LPS

En la *tabla 8* se pueden ver los valores obtenidos. Como todos estos equipos han utilizado el método ELVIRA, asociado a cada equipos mostramos los stakeholders identificados (la cantidad de stakeholders por equipo no tiene por qué ser a misma). Las columnas representan las métricas utilizadas para determinar el valor: Valor Esperado de Cliente (VEC), Valor Línea de Producto (VLP), Valor Real (VR), Valor Potencial (VP) y Valor No Cubierto (VNC). La intersección fila-columna se valora en unidades de requisitos. Esta tabla muestra los datos brutos a partir de los cuales se ha realizado el análisis de resultados.

		VEC	VLP	VR	VP	VNC
EA2	STK_A	33	35	31	4	2
	STK_B	25		22	13	3
EA3	STK_C	41	70	37	33	4
	STK_D	18		17	53	1
	STK_E	32		30	40	2
EA4	STK_F	32	42	32	10	0
	STK_G	24		22	20	2
EA5	STK_H	37	57	36	21	1
	STK_I	22		22	35	0
	STK_J	26		23	34	3

Tabla8: Métricas IOR sobre los resultados del experimento

7.4.4 ANÁLISIS E INTERPRETACIÓN

Para comprobar la utilidad del método ELVIRA hemos determinado el porcentaje de valor que el método ofrece a cada stakeholder. Para esto hemos utilizado la fórmula siguiente:

$$\%\ \text{valor} = \frac{VRstkn * 100}{\text{VEC n}}$$

Donde

- VRstkn = valor real para el stakeholder n
- VEC n = valor esperado del stakeholder n

La fórmula anterior permite determinar el valor aportado por el método ELVIRA a un stakeholder concreto. Para realizar este cálculo se enfrenta el valor esperado para un stakeholder contra el valor real obtenido para ese mismo stakeholder. Este valor se calcula de forma porcentual.

En la *tabla 9* se muestran los resultados de aplicar dicha fórmula, es decir, se muestra el valor aportado por el método ELVIRA de forma individual, para cada stakeholders. La segunda columna de la tabla muestra la parte no cubierta por ELVIRA (a esta parte la hemos denominado "no-valor"). Al ser datos porcentuales, el no-valor se obtiene mediante la diferencia entre el 100% y el valor para cada stakeholder.

		% Valor	% No valor
EA2	STK_A	93,9393939	6,060606061
	STK_B	88	12
EA3	STK_C	90,2439024	9,756097561
	STK_D	94,4444444	5,555555556
	STK_E	93,75	6,25
EA4	STK_F	100	0
	STK_G	91,6666667	8,333333333
EA5	STK_H	97,2972973	2,702702703
	STK_I	100	0
	STK_J	88,4615385	11,53846154

Tabla 9: Valor aportado por el método ELVIRA

En la *figura 35* se muestran estos mismos datos de forma gráfica. En color verde se representa el concepto valor y en color naranja el concepto no-valor. A simple vista se puede ver que el porcentaje de no-valor resulta muy bajo en todos los casos, incluso inexistente en dos de ellos, el stakeholder F (STK_F) del equipo EA4, y el stakeholder I (STK_I) del equipo EA5.

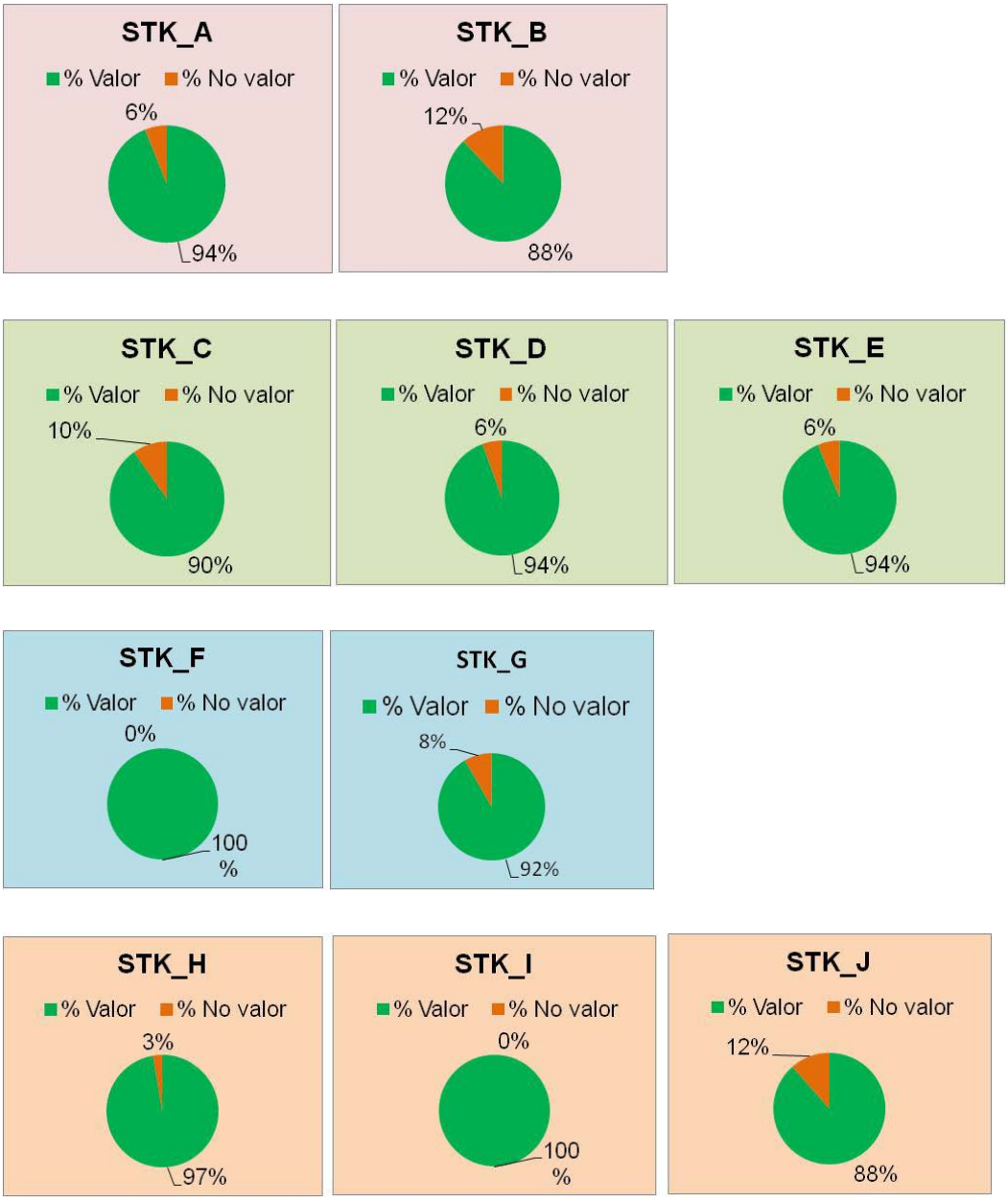


Figura 35: Valor que aporta ELVIRA a los stakeholders

Los datos resultan determinantes. Como se observa en la *figura 36*, el máximo no-valor no supera el 12%, mientras que el mínimo valor ofrecido es de un 88%.

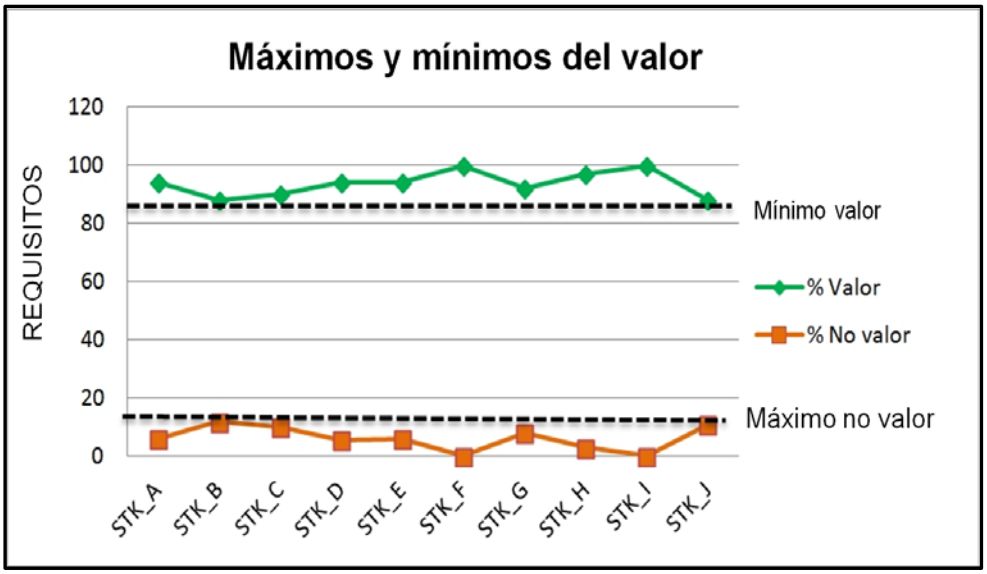


Figura 36: Valor mínimo y máximo aportado por ELVIRA

Calculando el promedio se puede afirmar que ELVIRA proporciona un valor medio del 93.78% a cualquier stakeholder implicado (véase la *figura 37*). Es un valor realmente alto.

Claramente el método ELVIRA *da valor al usuario, en las fases iniciales del desarrollo de software*". En consecuencia, se puede ratificar la segunda parte de la hipótesis.

Además de proporcionar un valor realmente alto a los stakeholders, el método ELVIRA va un paso más allá. Gracias a la integración de stakeholders, el método es capaz de ofrecer un valor potencial, es decir un conjunto de requisitos que un stakeholder concreto no ha tenido en cuenta en este momento o no considera

interesantes, pero que tampoco ha rechazado. De este modo ELVIRA, gracias a las aportaciones del resto de stakeholders, puede ofrecer un conjunto de requisitos potencialmente interesantes en un futuro. En la *figura 38* se muestra el valor potencial que ELVIRA ofrece individualmente a cada stakeholder.

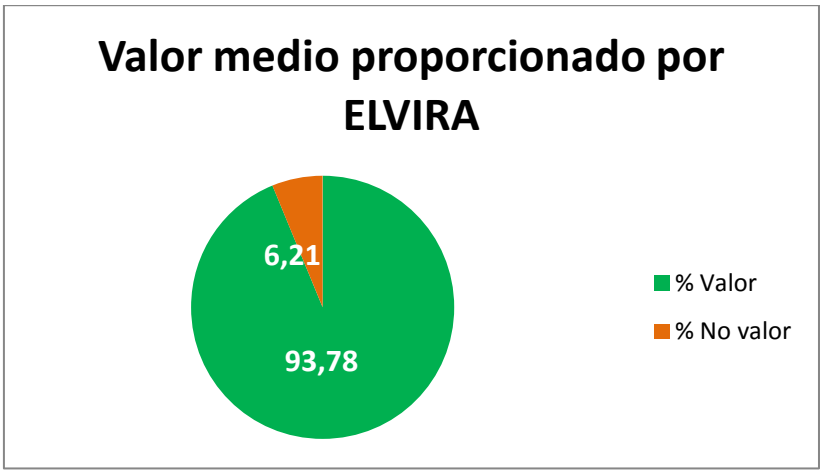


Figura37: Valor medio proporcionado por ELVIRA

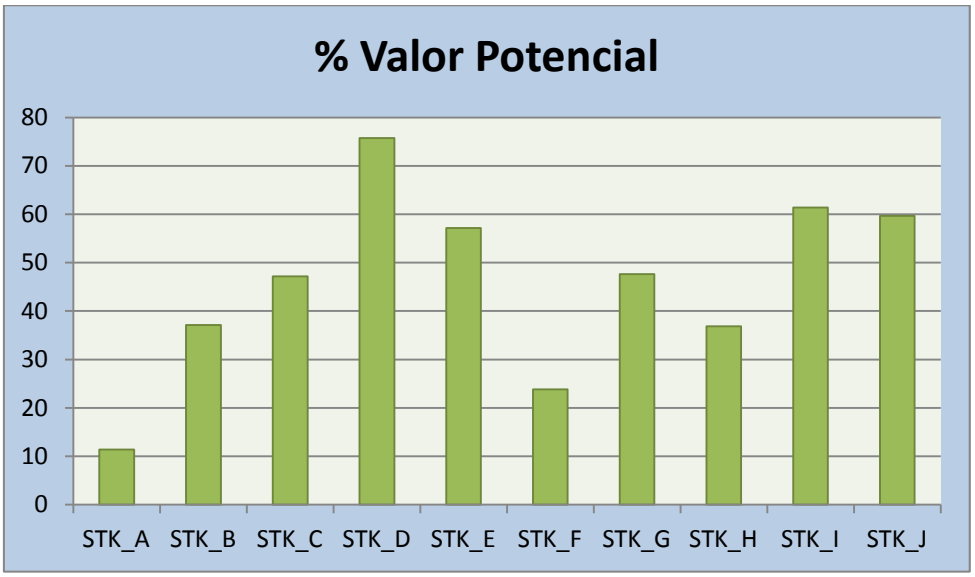


Figura 38: Valor potencial que ofrece ELVIRA

7.5 CONCLUSIONES DE LA VALIDACIÓN

El objetivo del presente capítulo ha sido demostrar, mediante un experimento formal, con ELVIRA la validez de la hipótesis expuesta en el capítulo 1:

"*Es posible integrar y utilizar el conocimiento de fuentes diversas en un método sistemático para la definición de una línea de productos software que dé valor al usuario, en las fases iniciales del desarrollo de software*".

Hemos realizado esta validación desde una doble perspectiva. Por un lado se ha validado la filosofía fundamental del método y por otro se ha validado la utilidad del método en sí mismo.

La validación de la filosofía fundamental del método tiene que ver con la primera parte de la hipótesis "*Es posible integrar y utilizar el conocimiento de fuentes diversas en un método sistemático para la definición de una línea de productos software*….". Por este motivo se han analizado los resultados del experimento buscando la eficacia de ELVIRA. A tenor de los resultados obtenidos se puede ratificar la primera parte de la hipótesis y decir que el método ELVIRA es eficaz. El método ELVIRA permite obtener ERS_LP más completos.

La validación de la utilidad del método tiene que ver con la segunda parte de la hipótesis: "*Es posible integrar y utilizar el conocimiento de fuentes diversas en un método sistemático para la definición de una línea de productos software que dé valor al usuario, en las fases iniciales del desarrollo de software*". Hay que recalcar que esta validación se ha realizado en las fases iniciales de producto y no una vez terminado el desarrollo del mismo, por lo tanto se ha realizado una validación temprana de requisitos. Se han analizado los resultados del experimento utilizando un indicador del contexto de Medios Sociales (Social Media), el indicador IOR (*Impact of Relationships*). Los resultados obtenidos permiten ratificar la segunda

parte de la hipótesis y afirmar que el método ELVIRA da valor al usuario en las fases iniciales del desarrollo de software.

CAPÍTULO 8- CONCLUSIONES Y LÍNEAS ABIERTAS

Aunque a lo largo de los capítulos anteriores hemos ido estableciendo las conclusiones que íbamos alcanzando, en este capítulo recogemos todas ellas frente a los objetivos inicialmente definidos. Así mismo, planteamos posibles líneas de acción futuras relacionadas con esta tesis.

8.1 ANÁLISIS DE CONSECUCIÓN DE OBJETIVOS

Para la consecución del objetivo planteado en el capítulo 1 "desarrollo un método formal para estructurar una familia de productos en función a las demandas y preferencias del mercado objetivo"planteamos las siguientes tareas:

En primer lugar, hemos realizado un estudio en profundidad del estado del arte relativo al enfoque de desarrollo de líneas de producto software, de la ingeniería de requisitos, y del enfoque de "Perspectivas o Vistas".

Como resultado hemos alcanzado las siguientes conclusiones:

- El mundo académico y el mundo empresarial van a velocidades diferentes en estos ámbitos. Mientras en el mundo académico la ingeniería de requisitos cada día ocupa más espacio de investigación y de formación, en el mundo empresarial es considerada una fase más, casi nunca se lleva a cabo formalmente y muy pocas veces deja evidencias tangibles y gestionadas. La distancia entre la investigación y la práctica sigue siendo muy grande.

- Si bien es clara la necesidad de crear una especificación de requisitos completa y consistente, existe una dificultad notoria por parte de los stakeholders, tanto del dominio del problema como del dominio de la solución, para realizar esta tarea. Los stakeholders del dominio del problema tienen grandes dificultades para definir y estructurar sus necesidades, mientras que los stakeholders del dominio de la solución tienen tendencia a plantear la solución antes de comprender el problema en su totalidad. Además, en cada proyecto concreto, la integración dentro de la ingeniería de requisitos de los distintos stakeholders se realiza ad-hoc y de un modo distinto. Ninguno de los métodos existentes se centra en la fase de ingeniería de requisitos, ni la aborda formalmente desde el punto de vista del cliente.
- La dificultad de experimentación en ingeniería de requisitos es clara. Tiene en su origen a las empresas que con su reticencia a mostrar información que consideran confidencial y con sus prisas en sacar productos al mercado, dificultan enormemente la labor de validación empírica.

Sobre esta base, hemos propuesto, desarrollado y validado un nuevo método, principal aportación de esta tesis, que hemos denominado ELVIRA. Con los resultados obtenidos en la experimentación realizada con el método, de cara a su validación, podemos afirmar que:

- Es un método que cubre las fases iniciales del desarrollo software, es decir, se centra en la ingeniería de requisitos y lo hace desde el punto de vista del cliente. Esto permite extraer y estructurar la diversidad de conocimiento requerida.

- Es un método que integra a los diversos stakeholders en la definición del producto, permitiendo trabajar tanto con stakeholders del dominio del problema como del dominio de la solución desde el principio.
- Es un método novedoso en el tratamiento de las inconsistencias. No las elimina, sino que las utiliza en el análisis de partes comunes y variantes de una LPS. De este modo, soporta la identificación de solapamientos y propone su resolución en base a ciertos criterios, guiando a los ingenieros de software en el tratamiento de los solapamientos de una manera estructurada.
- Es un método que permite cuantificar el valor del producto para cada stakeholder sin usar métricas tradicionales basadas en costos. Es capaz de cuantificar la importancia del producto, es decir, el grado en que los productos satisfacen las necesidades de la gente que los usa. Esto lo hace en las fases iniciales de definición de producto, ya que no tendría ningún sentido hacerlo a posteriori. La validación temprana y no basada en costos es una aportación importante de este trabajo.
- Es un método eficaz, sistemático y sencillo de usar. Permite obtener especificaciones de requisitos de líneas de producto software más completas, y facilita la definición de una línea de productos software que da valor al usuario en la ingeniería de requisitos.

Con todo esto, se ha ratificado la hipótesis planteada "*Es posible integrar y utilizar el conocimiento de fuentes diversas en un método sistemático para la definición de una línea de productos software que dé valor al usuario, en las fases iniciales del desarrollo de software*".

En cuanto a las aportaciones en forma de publicaciones realizadas en diversos foros científicos nacionales e internacionales, destacamos las siguientes:

- Ereño, M., Cortázar, D. R. “Utilización de QFD en la toma de decisiones para la estructuración de una familia de productos”.
    - AEMES. Revista Procesos y Métricas. Nº5. agosto 2005.
    - Taller de apoyo a la decisión en Ing. Sw. CEDI, Granada. 2005
- Ereño, M., Landa, U., Cortázar, D. R. 2005. “Software product lines structuring based upon market demands”. In Proceedings of the 2005 Conference on Specification and Verification of Component-Based Systems (Lisbon, Portugal). SAVCBS '05. ACM Press, New York, NY. September 2005. SIGSOFT 10.1145/1108768.1123072.
- Ereño,M., Cortázar, R. : ELVIRA method effectiveness. Story of an experiment. JISBD 2007. Eficacia del método ELVIRA-Relato de un experimento. In XII Jornadas JISBD. ISBN: 978-84-9732-595-0. (2007).
- M. Ereño, Jokin Arizmendi, Eusebi Calonge R. Joxeja Oiarbide “ TRANSER: ingeniería de requisitos en un Desarrollo Orientado al Mercado”. 11th Workshop on Requirements Engineering WER 2008.
- Ereño,M., Cortázar, R.: Getting the product value with IOR. PROFES '10 Proceedings of the 11th International Conference on Product Focused Software. Pages 118-119. ACM New York, ISBN: 978-1-4503-0281-4. (2010)

8.2 LÍNEAS FUTURAS

Esta tesis deja abiertas tres vías de continuación de la investigación realizada que se esbozan a continuación.

En primer lugar hay que señalar que el método ELVIRA, recoge gran cantidad de información valiosa, proporcionada por las distintas perspectivas de los distintos stakeholders. La gestión manual de la misma puede resultar pesada. Sería interesante desarrollar una aplicación software que automatizara y diera soporte al

método. Esta aplicación proporcionaría un medio más adecuado para la aplicación práctica del método, permitiría automatizar ciertas actividades y guiar la secuencia de pasos, actuando como un asistente que indicara qué hay que hacer en cada momento.

El segundo lugar sería muy interesante la adaptación del método ELVIRA al enfoque de desarrollo ágil.

Las metodologías ágiles son una aproximación al desarrollo software que apuestan por una cantidad apropiada de proceso, intentando evitar los burocráticos caminos de las metodologías tradicionales. Buscan el equilibrio en la relación proceso/esfuerzo y para ello se enfocan en los participantes y en los resultados [Fowler01].

Analizando los valores y principios del enfoque ágil, hemos detectado algunas características que apuntan a la posibilidad de aplicar ELVIRA en este contexto. Estas características son:

- La motivación que propició la creación de ELVIRA: la colaboración con el cliente-usuario desde etapas tempranas. EL enfoque ágil propone que exista una interacción constante entre el cliente y el equipo de desarrollo. Esta colaboración entre ambos será la que marque la marcha del proyecto y asegure su éxito. Esta idea está alineada totalmente con la filosofía subyacente a ELVIRA
- Se utiliza un enfoque basado en el "valor" para construir software. Como ya hemos comentado en el capítulo 7, el éxito del producto está ligado a las necesidades y preferencias individuales de los clientes. El enfoque ágil promueve la capacidad de respuesta a los cambios a lo largo del desarrollo, y no los perciben como un lastre sino como una oportunidad para mejorar el sistema e incrementar la satisfacción del cliente.

Desde el punto de vista del tipo de proyectos de desarrollo, el enfoque ágil está mostrando su efectividad en proyectos con requisitos muy cambiantes y cuando se exige reducir drásticamente los tiempos de desarrollo pero manteniendo una alta calidad. Esta casuística de proyecto, también es habitual en el enfoque de desarrollo de líneas de producto software. Este es otro indicador sobre la posibilidad de aplicar ELVIRA en el contexto ágil.

El último aspecto a investigar sería la adaptación del método ELVIRA en proyectos del ámbito Web 2.0.

El término Web 2.0 se le atribuye a Tim O´Reilly y Dale Dougherty, los cuales lo nombraron durante la conferencia O'Reilly Media Web 2.0 en 2004, estableciéndolo como *"una segunda generación en la historia de la web basada en comunidades de usuarios y una gama especial de servicios y aplicaciones de Internet que se modifica gracias a la participación social".*

La Web 2.0 está transformando de forma radical la forma en la que las empresas están utilizando Internet para sus negocios. Se sitúa al usuario en el centro del modelo de interacción, convirtiéndolo en el productor de contenidos, en desarrollador de aplicaciones, en partícipe, a fin de cuentas. El usuario pasa de tener un papel pasivo a ser un participante activo.

Asociado a este cambio de actitud, viene otro cambio muy interesante. Los usuarios hoy en día son hábiles con la tecnología, utilizan habitualmente herramientas y servicios web 2.0. Es más, el uso de la tecnología ha traspasado los límites de lo laboral y se ha instaurado en lo social y doméstico. Así, el modo en que estos usuarios comunican sus necesidades ha cambiado, y esto podría facilitar mucho la comunicación en la fase de ingeniería de requisitos.

Quizás ahora el problema es que la comunidad de usuarios aporta sus opiniones constantemente y utilizando diversos canales. Las empresas necesitan canalizar y

organizar toda esta información. Sería interesante analizar la aplicabilidad de ELVIRA como método para regular de alguna manera esa participación, gestionar las conversaciones y dar soporte a los nuevos modelos de negocio de la era 2.0.

Y para terminar dando apoyo a esta última idea quiero recordar las palabras de portada de la revista Time en el año 2006 , que destaca la importancia de los usuarios (stakeholders) que crean y usan la red, a la hora de crear y estructurar la nueva democracia digital: "*YOU. Yes, you. You control the Information Age. Welcome to your world".*

Bibliografía

[Agile01] Agile Alliance, http://www.agilealliance.org/.

[Akao90] Akao, Y. (1990): Quality Function Deployment. Integrating Customer Requirements into Product Design. Productivity Press, Cambridge MA. ISBN: 0-915299-41-0

[Alex04] I. Alexander and S. Robertson. (2004):Understanding Project Sociology by Modeling Stakeholders. IEEE Software. Vol.. 21, issue 1, pages 23–27

[Alenljung05] Alenljung, B. & Persson, A. (2005): Decision-making from the decision-maker's perspective. A framework for analysing decision situations. Proceedings of the 4th International Conference on Business Informatics Research. Pages 13-22. Skövde, Sweden

[Alford77] Alford M. W. (1977): A Requirements Engineering. Methodology for Real-Time Process Requirements. IEEE Transactions on Software Engineering, Vol. 3, issue 1, pages 60-69

[America00] America, P., Obbink, H., Müller, J., and Rob van Ommering R.V., COPA (2000): A Component-Oriented Platform Architecting Method for Families of Software Intensive Electronic Products. Tutorial for SPLC1, the First Software Product Line Conference. Denver, Colorado.

[Atkinson02] Atkinson, C., Bayer, J., Bunse, C., Kamsties, E., Laitenberger, O., Laqua, R., Muthig, D., Paech, B., Wüst, J. and Zettel, J. (2002): Component-based Product Line Engineering with UML. Addison Wesley

[Atkinson00] Colin Atkinson, Joachim Bayer, Dirk Muthig. (2000): Component-Based Product Line Development. The KobrA Approach, 1st International Software Product Line Conference.

[Aurum03] Aurum, A.; Wohlin, C. (2003): The fundamental nature of requirements engineering activities as a decision-making process. Journal, Information and Software Technology. Vol 45, issue 14, pages 945-954. Elsevier.

[Avison99] Avison, D., Lan, F., Myers, M. & Nielsen, A. (1999): Action Research. Vol 42, pages 94-97. Communications of the ACM

[Avison99] Avison, D., Lan, F., Myers, M. & Nielsen, A. (1999): Action Research. Vol 42, pages 94-97. Communications of the ACM

[Babar10] M. Ali Babar, L. Chen, F. Shull. (2010): Managing variability in software product lines.IEEE Software. Vol 27, pages 89–91

[Ballejos11] Ballejos, L.C. and Montagna, J.M. (2011): Modeling stakeholders for information systems design process. Requirements Engineering. Springer.

[Balzer91] R. Balzer. (1991): Tolerating inconsistency. En Proc. 13th Int. Conf. on Software Engineering. Pages 158-165. IEEE Computer Society Press.

[Barlas96] Barlas, S. (1996): Anatomy of a Runaway. What Grounded the AAS. IEEE Software. Vol 13, issue 1, pages 104-6

[Baskerville99] Baskerville, R. (1999): Investigating Information Systems with Action Research. Communications of the Association for Information Systems.Vol 2, issue 4

[Bass98] L. Bass, P. Clements, R. Kazman (1998): Software Architecture in Practice. Addison Wesley

[Beck01] Beck, K., et al. (2001): The Agile Manifesto. Manifesto for Agile Software Development. www.agilemanifesto.org

[Beck99] Beck, K. (1999).Extreme programming explained. Addison-Wesley, Upper Saddle River

[Bell07] P. Bell. (2007): Proc. Conf. Object-Oriented Programming Systems Languages and Applications (OOPSLA 07). A Practical High Volume

Software Product Line. Pages 994–1003. ACM Press

[Berander05] Berander, P., A. Andrews (2005),: Requirements Prioritization. A. Aurum, C. Wohlin (eds.), Engineering and Managing Requirements, Springer.

[Bohem05] Boehm, B. (2005): Overview and Agenda. Value-Based Software Engineering. Pages 3-14. Springer

[Boehm94] B.W.Bohem, P.Bose, E.Horowitz, y M.J. Lee (1994): Software Requirements as Negotiated Win Conditions. Proceedings of the First International Conference on Requirements Engineering.

[Booch94] Booch, G.(1994): Object-oriented Analysis and Design with Applications. Benjamin Cummings, Redwood City, Ca.

[Bosch10] Bosch, J.(2010): Toward Compositional Software Product Lines. IEEE Software. Vol 27, issue 3, pages 29-34

[Bosch02] Bosch et al. (2002): Variability Issues in Software Product Lines. Proc. 4th Int'l Workshop on Software Product-Family Eng. Pages 13–21. Springer

[Bosch00] Bosch, J. (2000): Design And Use of Software Architectures. Adopting and evolving a product-line approach. Addison-Wesley ACM Press

[Bosch98] Bosch, J. (1998): Product-Line Architectures in Industry: A Case Study. Proceedings of the 21st International Conference on Software Engineering (ICSE'99)

[Booch94] Booch, G. (1994): *Object-oriented Analysis and Design with Applications*, Benjamin Cummings, Redwood City, Ca

[Boudreau01] Boudreau, M.C., Gefen, D. and Straub, D. (2001): Validation in IS Research: A State-of-the-Art Assessment. Vol. 25, issue 1, pages 1-16. MIS Quarterly

[Bowman96] Bowman, H., Derrick, J., Linington, P. and Steen, M. (1996): Cross-viewpoint consistency in Open Distributed Processing. BCS/IEE Software Eng. Vol 11, issue 1, pages 44-57

[Bourque99] P.Bourque, R.Dupuis, y A.Abran (1999): The guide to the software engineering body of knowledge. IEEE Softtware. Vol 16, issue

[Bray02] Bray, I.K. (2002): An introduction to requirements engineering. Harlow, England:Addison Wesley

[Briand00] Briand, L., Arisholm, S., Counsell, F., Houdek, F. and Thévenod-Fosse, P. (2000): Empirical Studies of Object-Oriented Artefacts, Methods, and Processes: State of the Art and Future Directions. Empirical Software Engineering. Vol. 4, issue 4, pages 387-404

[Brooks87] Brooks, F. (1987): No silver Bullet. Essence and Accidents of Software Engineering.. Vol 20, pages 10-19. Computer

[Burchil97] Burchill, Gary and Christina Hepner Brodie. (1997): Voices into Choices: Acting on the Voice of the Customer. Center for Quality Management. Joiner Publication Madison, WI.

[Carlshamre02] Carlshamre, P. (2002): A Usability Perspective on Requirements Engineering. Methodology to Product Development. Issue 726. Linköping University, Linköping Studies in Science and Technology.

[Castello12] Castelló Martínez, A. (2012): Del ROI al IOR: el retorno de la inversión de la comunicación empresarial y publicitaria en medios sociales. ISBN 978-84-615-5678-6

[Celaya08] Celaya, J. (2008): La empresa en la Web 2.0. Madrid: Gestión 2000.

[Chastek01] Chastek, G. Donohoe, P. Kang, K. , Thiel, S. (2001): Product Line Analysis: A Practical Introduction (CMU/SEI-2001-TR-001, ADA396137). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University

[Chechik01] Chechik, M.(2001): A framework for multi-valued reasoning over inconsistent viewpoints. ICSE '01 Proceedings of the 23rd International Conference on Software Engineering. Pages 411-420. IEEE Computer Society Washington, ISBN:0-7695-1050-7

[Chen09] Chen, M.A. Babar, N. Ali (2009): Variability management in software product lines: a systematic review. Pages 81-90. SPLC'09

[Clarke01] S. Clarke and R. J. Walker, Composition Patterns (2001): An Approach to Designing Reusable Aspects, International Conference on Software Engineering (ICSE).

[Clements02] Clements, P. and Northrop, L. (2002): Software Product Lines. Practices and Patterns. Addison Wesley

[Cohen95] Cohen, L. (1995): Quality Function Deployment: How to Make QFD Work for You. ISBN: 0-201-63330-2

[Conger94] Conger, S. (1994): The New Software Engineering. International Thomson Publishing.

[Cooper88] Cooper, R. G. and Kleinschmidt, E. J. (1988): Resource Allocation in the New Product Process. Vol 17, pages 249-262. Industrial Marketing Management

[Coriat00] Coriat, M., Jourdan, J. and Boisbourdin, F. (2000): The SPLIT Method, Building Product Lines for Software-Intensive Systems. Pages 147 – 166. SPLC1 Massachusetts, Kluwer Academic Publishers

[Costello95] Rita J. Costello, Dar-Biai Liu (1995): Metrics for Requirements Engineering. Journal of Systems and Software archive. Vol. 29, issue 1, pages 39-63

[Cotterell95] Cotterell, M. and Hughes, B., (1995): Software Project Management. International Thomson Publishing.

[Czarnecki00] Czarnecki, K. and Eisenecker, U.W. (2000): Generative Programming: Methods, Tools, and Applications. Addison Wesley, Boston

[Davis03] Davis, A. (2003): The art of requirements triage. Vol. 36, issue 3, pages 42-49. IEEE Computer

[Davis95] Davis, A. (1995): 201 Principles of Software Development. McGraw-Hill, ISBN: 0-07-015840-1

[DeBaud98] DeBaud, J., Bayer, J., Flege, O., Knauber, P., Lagua, R., Muthing, D., Schmid, K., Widen, T. (1999): PuLSE: a methodology to develop software product lines. SSR '99 Proceedings of the 1999 symposium on Software reusability. Pages 122-131. ACM New York, NY, ISBN: 1-58113-101-1

[Diaz11] Díaz, J., Pérez, J., Alarcón, P., Garbajosa. J. (2011): Agile Product Line Engineering - A Systematic Literature Review. Software—Practice & Experience archive. Volume 41, Issue 8, Pages 921-941. John Wiley & Sons, Inc. New York, NY

[Dieste12] Dieste, O., Fernández, E., García, R., Juristo, N. (2012): Comparison of metaanalysis methods: understanding the influence of experiments' statistical parameters. EMSE

[Dieste09] Dieste, O., Fernández, E., Pesado, P., García-Martínez, R. (2009): Analysis of Inspection Technique Performance. Proceedings XV Congreso Argentino de Ciencias de la Computación. Pages 961-970. Workshop de Ingeniería de Software, ISBN 978-897-24068-4-1

[Dieste07] Dieste, O., Griman, A. (2007): Developing Search Strategies for Detecting Relevant Experiments for Systematic Reviews. ESEM '07 Proceedings of the First International Symposium on Empirical Software Engineering and Measurement. Pages 215-224. IEEE Computer Society Washington, ISBN:0-7695-2886-4

[dinwal02] A. Dingwall-Smith and A. Finkelstein. (2002): From Requirements to Monitors by way of Aspects, AOSD 2002 Workshop on Early Aspects, 2002

[Dix93] Dix, A., Finlay, J. Abowd, G. , Beale, R. (1993): Human-Computer Interaction. Prentice-Hall.

[Duran00] Durán, A. (2000): Un Entorno Metodológico de Ingeniería de requisitos para Sistemas de Información. Tesis doctoral, Universidad de Sevilla

[Easterbrook05] Easterbrook, S., Aranda, J., Fan, Y., Horkoff, J., Leica, M., Abdul, R. (2005): Do Viewpoints Lead to Better Conceptual Models? An Exploratory Case Study. RE '05 Proceedings of the 13th IEEE

International Conference on Requirements Engineering. Pages 199-208. IEEE Computer Society Washington, ISBN: 0-7695-2425-7

[Easterbrook01] Easterbrook, S., Nuseibeh, B. (1996): Using viewpoints for inconsistency management. Software Engineering Journal. Volume 11, issue 1, pages 31-43. BCS/IEE Press

[Easterbrook96] Easterbrook, S. (1996): Learning from Inconsistency. IWSSD '96 Proceedings of the 8th International Workshop on Software Specification and Design. Page 136. IEEE Computer Society Washington, ISBN:0-8186-7361-3

[Easterbrook96a] Easterbrook S. (1991): Handling conflict between domain descriptions with computer-supported negotiation. Knowledge Acquisition.Volume 3, issue 3, pages 255-289. Academic Press Ltd, London

[Easterbrook95] Easterbrook, S. M., & Nuseibeh, B. A. (1995): Managing Inconsistencies in an Evolving Specification. In Second IEEE Symposium on Requirements Engineering, (pp. 48-55). York, UK: IEEE Computer Society Press.

[Easterbrook91] S. Easterbrook. (1991): Handling conflict between domain descriptions with computer-supported negotiation. Knowledge Acquisition, 3:255-289

[Ebert05] Ebert, C. & Wieringa, R.J. (2005) :Requirements engineering: Solutions and trends. In: A. Aurum & C. Wohlin (Eds.) Engineering and managing software requirements (pp 453-476). Berlin, Germany: Springer

[Elrad01] T. Elrad, R. Filman, and A. Bader (eds.) (2001): Theme Section on Aspect-Oriented Programming, Communications of ACM, Vol. 44, No. 10

[Ereño10] Ereño,M., Cortazar, R. (2010): Getting the product value with IOR. PROFES '10 Proceedings of the 11th International Conference on Product Focused Software. Pages 118-119 . ACM New York, ISBN: 978-1-4503-0281-4

[Ereño08] Ereño, M., Arizmendi, J., Calonge, E., Oiarbide, J. (2008): TRANSER:

Ingeniería de requisitos en un Desarrollo Orientado al Mercado. 11th Workshop on Requirements Engineering WER

[Ereño07] Ereño, M., Cortazar, R. (2007): ELVIRA method effectiveness. Story of an experiment. JISBD 2007. Eficacia del método ELVIRA-Relato de un experimento. XII Jornadas JISBD. ISBN: 978-84-9732-595-0

[Ereño05] Ereño, M., Cortazar, D. R. (2005): Utilización de QFD en la toma de decisiones para la estructuración de una familia de productos. Taller de apoyo a la decisión en Ing. Sw.. CEDI, Granada.

[Ereño05a] Ereño, M., Cortazar, D. R. (2005): Utilización de QFD en la toma de decisiones para la estructuración de una familia de productos. AEMES. Revista Procesos y Metricas. Nº5. agosto 2005

[Eriksson07] Eriksson, U. (2007): Kravhantering för IT-system: Requirements engineering for IT systems. Lund, Sweden: Studentlitteratur

[Erikkson93] Erikkson, I. (1993): Quality Function Deployment: A Tool to Improve Software Quality. Information and Software Technology. Volume 35, issue 9, pages 491-498

[Esterman01] Esterman, M., and Ishii, K. (2001): Concurrent Product Development Across the Suppy Chain: Development of Integrator/Supplier Risk and Coupling Indices. ASME 2001 Design Engineering Technical Conferences and Computers and Information in Engineering Conference

[Faegri08] Hanssen, G., Fígri, T. (2008): Process fusion: An industrial case study on agile software product line engineering. Journal of Systems and Software archive. Volume 81, issue 6, pages 843-854. Elsevier Science Inc. New York, NY

[Faulk97] Faulk, S.R. (1997): Software Requirement: A tutorial. Software Requirements Engineering. Pages 128-149. Los Alamitos, CA: IEEE Computer Society Press

[Ferdinardi02] Ferdinandi, P. L. (2002): A Requirements Pattern. Succeeding in the Internet Economy. Addison-Wesley

[Fernandez10] Fernández, E., Pollo, M., Amatriain, H., Dieste, O., Pesado, P.,

García- Martínez, R. (2010): Ingeniería de Software Empírica. Aplicabilidad de Métodos de Síntesis Cuantitativa. En Ingeniería de Software e Ingeniería del Conocimiento: Tendencias de Investigación e Innovación Tecnológica en Iberoamérica. Páginas 287-297. Alfaomega Grupo Editor, ISBN: 978-607-707-096-2

[Fernandez09] Fernández, E., Dieste, O., Pesado, P., García-Martínez, R. (2009): Pautas para Agregar Estudios Experimentales en Ingeniería del Software.Proceedings XIV Jornadas de Ingeniería del Software y Bases de Datos. Páginas 91-102. ISBN 978-84-692-4211-7

[Freitas11] Freitas da Silva, I., et al. (2011): Agile Software Product Lines: a Systematic Mapping Study. Sotware Practice and Experience. Volume 41, issue 8, pages 899-920. John Wiley & Sons, Inc. New York, NY

[Finkelstein96] Finkelstein A., Spanoudakis, G., Till D. (1996): Managing Interference. ISAW '96 Joint proceedings of the second international software architecture workshop (ISAW-2) and international workshop on multiple perspectives in software development (Viewpoints '96) on SIGSOFT '96 workshops. Pages 172-174. ACM New York, NY, ISBN:0-89791-867-3

[Finkelstein94] Finkelstein, A., Gabbay, D., Hunter, A., Kramer, J., Nuseibeh, B. (1994): Inconsistency handling in multiperspective specifications. IEEE Transactions on Software Engineering. Volume 20 Issue 8, pages 569-578. IEEE Press Piscataway, NJ

[Fowler01] M. Fowler and J. Highsmith. (2001): The Agile Manifesto. Software Development Magazine. August. http://www.sdmagazine.com/documents/s=844/sdm0108a/0108a.htm

[Freeman84] Freeman, R.E. (1984): Strategic Management: A stakeholder approach. Pitman, Boston

[Gabbay91] Gabbay, D., Hunter, A. (1991): Making Inconsistency Respectable: A Logical Framework for Inconsistency in Reasoning. FAIR '91 Proceedings of the International Workshop on Fundamentals of Artificial Intelligence Research. Pages 19-32. Springer-Verlag London, UK, ISBN:3-540-54507-7

[Gallivan03] Gallivan, M.J., Keil, M., (2003): The User-Developer Communication Process: A Critical Case Study. Information Systems Journal. Volume 13, issue 1, pages 37-68

[GAO79] GAO (1979): Contracting for Computer Software Development-Serious Problems Require Management Attention to Avoid Wasting Millions. US General Accounting Office

[GarciaDuque09] Garcia-Duque, J., Pazos-Arias, J.J., Lopez-Nores, M., Blanco-Fernandez, Y., Fernandez-Vilas, A., Diaz-Redondo R.P., Ramos-Cabrer, M., Gil-Solla, A. (2009): Methodologies to evolve formal specifications through refinement and retrenchment in an analysis and revision cycle. Requirements Engineering Journal. Volume 14, issue 3, pages 129-153. Springer-Verlag New York

[Gibbs94] Gibbs, W.W. (1994): Software's Chronic crisis. Scientific American. Volume 271, issue 3, pages 72-81

[Goguen93] Goguen, J.A., Linde, C. (1993): Techniques for requirements elicitation.Proceedings of IEEE international symposium on requirements engineering (RE 1993). Pages 152–164. ISBN: 0-8186-3120-1

[Glass02] Glass, R.L., Vessey, I. ,Ramesh, V. (2002): Research in software engineering: an analysis of the literature. Information and Software Technology. Volume 51, issue 1, pages 68-70. Butterworth-Heinemann Newton, MA

[Glinz07] Glinz, Martin and Wieringa, Roel J. (2007): Stakeholders in requirements engineering. IEEE Software, 28 (1). pp. 18-20. ISSN 0740-7459

[Gregg01] Gregg, D. G., Kulkarni, U. R. , Vinzé, A. S. (2001): Understanding the Philosophical Underpinnings of Software Engineering Research in Information Systems. Information Systems Frontiers. Volume 3 Issue 2, pages 169-183. Kluwer Academic Publishers Hingham, MA

[Griss98] Griss, M.L., Faravo, J., d'Alessandro, M. (1998): Integrating Feature Modeling with the RSEB. Proceedings of the Fifth International Conference on Software Reuse. Pages 76-85. IEEE Computer

Society Washington, ISBN:0-8186-8377-5

[Grunbacher05] Seyff, N., Hoyer, C., Kroiher, E., Grünbacher, P.(2005): Enhancing GSS-based Requirements Negotiation with Distributed and Mobile Tools.WETICE '05 Proceedings of the 14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise. Pages 87 - 92. IEEE Computer Society Washington, ISBN:0-7695-2362-5

[Grundy99] J. Grundy, (1999): Aspect-Oriented Requirements Engineering for Component-based Software Systems, 4th IEEE International Symposium on RE. IEEE Computer Society Press, pp. 84-91

[Grundy98] Grundy, J., Hosking, J., Mugridge, W. B. (1998): Inconsistency management for multiple-view software development environments. IEEE Trans. on Software Engineering. Volume 24, issue 11, pages 960–981. IEEE Press Piscataway, NJ

[Haag96] Haag, S., Raja, M.K., Schkade, L.L. (1996): Quality Function Deployment - Usage in Software Development. Communications of the ACM. Volume 39, issue 1, pages 41-49. ACM New York, NY

[Halling04] Halling, M., Biffl, S., Grunbacher, P. (2004): The role of valuation in value-based software engineering. Sixth International Workshop on Economics-Driven Software Engineering Research (EDSER-6).

[Halmans03] Halmans, G.; Pohl, K. (2003): Communicating the Variability of a Software-Product Family to Customers. Software and Systems Modeling, 2 1, März, S. 15-36.

[Hanssen08] Hanssen, G.K., Faegri, T.E. (2008): Process Fusion: An Industrial Case Study on Agile Software Product Line Engineering. Journal of Systems and Software. Volume 81, issue 6, pages 843–854. Elsevier Science Inc. New York, NY

[Hofmann04] Hoffmann, M., Kühn, N., Weber, N., Bittner, M. (2004): Requirements for requirements management tools. PRE '04 Proceedings of the Requirements Engineering Conference, 12th IEEE International. Pages 301-308. IEEE Computer Society Washington, ISBN:0-7695-2174-6

[Host02] Höst, M. (2002): Introducing Empirical Software Engineering Methods in Education. CSEET '02 Proceedings of the 15th Conference on Software Engineering Education and Training. Page 170. IEEE Computer Society Washington

[Hunter98] Hunter, A., Nuseibeh, B. (1998): Managing Inconsistent Specifications: Reasoning, Analysis and Action. ACM Transactions on Software Engineering and Methodology (TOSEM). Volume 7, issue 4, pages 335 - 367. ACM New York, NY

[IEEE04] Guide to the Software Engineering-Body of Knowledge. IEEE Comp. Society Press. (2004)

[IEEE98] IEEE/ANSI 830-1998: Standard for Software Requirements Specification. (1998)

[IEEE96] IEEE/EIA 12207.0-1996 - Standard for Information Technology - Software Life Cycle Processes. (1996)

[IEEE90] IEEE Standard Glossary of Software Engineering Terminology (1990): IEEE Standard 610.12-1990. Institute of Electrical and Electronics Engineers. (1990)

[Jaaksi02] Jaaksi, A. (2002): Developing Mobile Browsers in a Product Line. IEEE Software. Volume 10, issue 4, pages 73–80. IEEE Computer Society Press Los Alamitos

[Jacobson97] Jacobson, I., Griss, M., Jonsson, P. (1997): Software Reuse: Architecture, Process and Organization for Business Success. ACM Press/Addison-Wesley, ISBN:0-201-92476-5

[Jacobson93] Jacobson, I., Christensen, M., Jonsson, P., Overgaard, G. (1993): Object-Oriented Software Engineering. Addison-Wesley, Wokingham

[Jackson83] Jackson, M. A. (1983): System Development. Prentice-Hall, London

[Jayazeri99] Jayazeri, M. & Hopper, T. (1999): Management accounting with world class manufacturing:a case study. Management Accounting Research, 10(3):263-303.

[Jenesen09] Jensen, P. (2009): Experiences With Software Product Line Development. Volume 22, issue 1, pages 11–14

[Jiang07] Jiang, L., Eberlein, A. (2007): Selecting Requirements Engineering Techniques based on Project Attributes - A Case Study. ECBS '07 Proceedings of the 14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems. Pages 269-278. IEEE Computer Society Washington, ISBN:0-7695-2772-8

[Jones10] Jones, L.G., Northrop, L.M. (2010): Clearing the Way for Software Product Line Success. IEEE Software. Volume 27 , issue 3, pages 22-28. IEEE Computer Society Press Los Alamitos

[Juristo01] Juristo, N., Moreno, A. (2001): Basics of Software Engineering Experimentation. KSpringer Publishing Company, ISBN:1441950117 9781441950116

[Kaler03] Kaler, J. (2003): Differentiating Stakeholder Theories. Journal of Business Ethics Dordrecht. Volume 46, issue 1, pages 71–83

[Kamsties98] Kamsties, E., Hörmann, K., Schlich, M. (1998): Requirements Engineering in Small and Medium Enterprises. Requirements Engineering. Volume 3, pages 84–90

[Kakarontzas10] Kakarontzas, G., Stamelos, I. (2010): Component Recycling for Agile Methods. QUATIC '10 Proceedings of the 2010 Seventh International Conference on the Quality of Information and Communications Technology. Pages 397-402. IEEE Computer Society Washington, ISBN: 978-0-7695-4241-6

[Kang02] Kang, K.C., Lee, J., Donohue, P. (2002): Feature-Oriented Product Line Engineering. IEEE Software. Volume 10, issue 4, pages 58–65

[Kang98a] Kang K.C. et al. (1998): FORM: A Feature-Oriented Reuse Method with Domain-Specific Reference Architectures. Annals of Software Engineering. Volume 5, issue 1, pages 143–168. J. C. Baltzer AG, Science Publishers Red Bank

[Kang98] Kang, K.C. (1998): Feature-Oriented Development of Applications for a Domain. ICSR '98 Proceedings of the 5th International Conference on Software Reuse. Pages 354-355. IEEE Computer

Society Washington, ISBN:0-8186-8377-5

[Kang90] Kang, K.C. et al. (1990): Feature-Oriented Domain Analysis (FODA) Feasibility Stud. Tech. report CMU/SEI-90-TR-21. Carnegie Mellon Software Eng. Inst.

[Kaplan10] Kaplan Andreas, M., Haenlein, M. (2010): Users of the world, unite! The challenges and opportunities of social media. In Business Horizons. Volume 53, issue 1, pages 59-68

[Karlsson97] Karlsson, J., Ryan, K. (1997): Prioritizing requirements using a cost-value approach. IEEE Software. Volume 14, issue 5, pages 67–74

[Keil95] Keil, M., Carmel, E. (1995): Customer-Developer Links in Software Development. Communications of the ACM. Volume 38, issue 5, pages 33-44. ACM New York

[Khan89] Khan, M. B., Martin, M. P. (1989): Managing The Systems Project. Journal of Systems Management. Volume 40, issue 1, pages 31-36

[Khazanchi98] Khazanchi, D. y Munkvold (1998): B.E. Is Information Systems a Science? Information Systems- The Next Generation. Pages 1-12

[Khurana97] Khurana, A. and Rosenthal, S. R. (1997): Integrating the Fuzzy Front End of New Product Development. Sloan Management Review. Volume 38, issue 2, pages 103-12

[Kitchenham02] Kitchenham, B., Pfleeger, S., Pickard, L., Jones, P., Hoaglin, D., El Emam, K., Rosenberg, J. (2002): Preliminary Guidelines for Empirical Research in Software Engineering. IEEE Transactions on Software Engineering, Volume 28, issue 8, pages 721-734. IEEE Press Piscataway

[Kock01] KOCK, N., LAU, F. (2001): Information Systems Action Research: Serving Two Demanding Masters. Information Technology & People (special issue on Action Research in Information Systems). Volume 14, pages 6-11

[Kotonya98] Kotonya, G., Sommerville, I. (1998): Requirements Engineering: Processes and Techniques. John Wiley & Sons, Inc. New York, ISBN:0471972088

[Kotonya96] Kotonya, G., Sommerville, I. (1996): Requirements Engineering with viewpoints. Software Engineering Journal. Volume 11, issue 1, pages 5-11

[Kotonya92] Kotonya, G. and Sommerville, I. (1992): Viewpoints For Requirements Definition. Software Engineering Journal. Volume 7, issue 6, pages 375-387. Michael Faraday House Herts, UK

[Kujala05] Kujala, S., Kauppinen, M., Lehtola, L., Kojo, T. (2005): The Role of User Involvement in Requirements Quality and Project Success.RE '05 Proceedings of the 13th IEEE International Conference on Requirements Engineering. Pages 75-84. IEEE Computer Society Washington, ISBN:0-7695-2425-7

[Kujala03] Kujala, S. (2003): User Involvement: A Review of the Benefits and Challenges. Behaviour & Information Technology. Volume 22, issue 1, pages 1-16

[Kuloor02] Kuloor, C., Eberlein, A. (2002): Requirements Engineering for Software Product Lines. Proceedings of the 15th International Conference on Software & Systems Engineering and their Applications (ICSSEA'02). Paris, France

[Kuusela05] J. Bosch,G. Florijn,D. Greefhorst,J. Kuusela,H. Obbink,K. Pohl. (2005): Variability Issues in Software Product Lines. PFE '01 Revised Papers from the 4th International Workshop on Software Product-Family Engineering. Pages 13-21. Springer-Verlag London, ISBN:3-540-43659-6

[Kuusela00] Juha Kuusela , Juha Savolainen, (2000): Requirements engineering for product families, Proceedings of the 22nd international conference on Software engineering, p.61-69, June 04-11, 2000, Limerick, Ireland

[Laguna05] Laguna, M. A., González-Baixauli, B. (2005): Goals and MDA in Product Line Requirements Engineering. Técnico 2005-01, Grupo de Investigación GIRO. Departamento de Informática. Universidad de Valladolid

[Lamas10] Lamas, C. (2010): Los medios interactivos y su publicidad. La

medición de audiencias. Telos. No 82

[Lamia95] Lamia, W. M., (1995): Integrating QFD with Object Oriented Software Design Methodologies. Transactions from the Seventh Symposium on Quality Function Deployment. Pages 417-434

[Lamsweerde09] Lamsweerde, A. (2009): Reasoning about alternative requirements options. Conceptual Modeling: Foundations and Applications. Pages 380–397. Springer-Verlag Berlin, Heidelberg, ISBN: 978-3-642-02462-7

[Lamsweerde04] Lamsweerde, A. (2004): Goal-oriented requirements engineering: a roundtrip from research to practice. Proceedings of 12th IEEE joint international requirements engineering conference. Pages 4–8. IEEE Computer Science Press

[Lamsweerde01] Lamsweerde, A. (2001): Goal-Oriented Requirements Engineering: A Guided Tour. RE '01 Proceedings of the Fifth IEEE International Symposium on Requirements Engineering. Page 249. IEEE Computer Society Washington

[Lamsweerde98] Lamsweerde, A., Darimont, R., Letier, E. (1998): Managing conflicts in goal-driven requirements engineering. IEEE Transactions on Software Engineering. Volume 24, issue, pages 908-926. IEEE Press Piscataway

[Lausen02] Lauesen, S. (2002): Software requirements – styles and techniques. Pearson Education, Essex, ISBN:0201745704

[Leffingell03] Leffingwell, D., Widrig, D. (2003): Managing Software Requirements: A Use Case Approach, Second Edition. Pearson Education. ISBN:032112247X

[Lehtola04] Lehtola, L., Kauppinen, M., Kujala, S. (2004): Requirements prioritization challenges in practice. Proceedings of 5th International Conference on Product Focused Softwar e Process Improvement, Lecture Notes in Computer Science. Volume 3009, pages 497-508. Springer-Verlag, Heidelberg

[Leite91] Leite, J.C.P., Freeman, P.A (1991): Requirements validation through viewpoint resolution. IEEE Transactions on Software Engineering. Volume 17, issue 12, pages 1253-1269. IEEE Press

Piscataway, NJ

[Leite89] Leite, J. C. P. (1989): Viewpoints Analysis: A Case study. ACM J. Software Engineering Notes. Volume 14, issue 3, pages 111-119. ACM New York, NY

[Letier02] Letier, E., Lamsweerde, A. (2002): Deriving operational software specifications from system goals. SIGSOFT '02/FSE-10 Proceedings of the 10th ACM SIGSOFT symposium on Foundations of software engineering. Pages 119 - 128. ACM New York, NY , ISBN:1-58113-514-9

[Lewin47] Lewin, K. (1947): Frontiers in Group Dynamics. Human Relations. Volume 1. pages 5-41

[Linden02] Van der Linden, F. (2002): Software Product Families in Europe: The Esaps and Café Projects. IEEE Software. Volume 10, issue 4, pages 41–49

[Liu02a] Lin, S. (2002): Capturing complete and accurate requirements by refinement. IICECCS '02 Proceedings of the Eighth International Conference on Engineering of Complex Computer Systems . Pages 57–67. IEEE Computer Society Washington, ISBN:0-7695-1757-9

[Liu02] Liu, W., Easterbrook, S. M., Mylopoulos, J. (2002): Rule-based detection of inconsistency in uml models. Wkshp on Consistency Problems in UML-Based Software Development, 5th Int. Conf. on the Unified Modeling Language

[Lubars93] Lubars, M., Potts, C., Richter, C. (1993): A review of the state of the practice in requirements modelling. Proceedings of IEEE Symposium on Requirements Engineering (RE´93). IEEE Computer Society Press (1993

[Macauly93] Macaulay, L. (1993): Requirements Capture as a Cooperative Activity. Proc. 1st IEEE Int'l Symp. Requirements Eng. Pages 174–181. IEEE CS Press, ISBN: 0-8186-3120-1

[Machado05] Machado, R.J., Ramos, I., Fernandes, J.M. (2005): Specification of requirements models. A. Aurum & C. Wohlin (Eds), Engineering

and managing software requirements. Pages 47-68. Berlin, Germany: Springer

[Maiden07] Maiden, N., Ncube, C., Robertson, S. (2007): Can Requirements Be Creative? Experiences with an Enhanced Air Space Management System. ICSE '07 Proceedings of the 29th international conference on Software Engineering. Pages 632-641. IEEE Computer Society Washington, ISBN:0-7695-2828-7

[Maiden04] Maiden, N., Robertson, S., Gizikis, A. (2004): Provoking Creativity: Imagine What Your Requirements Could be Like. IEEE Software. Volume 21, issue 5, pages 68-75. IEEE Computer Society Press Los Alamitos

[Marcos08] Marcos, Esperanza,(2008): "Investigación en Ingeniería del Software vs. Desarrollo Software";
http://www.ciencias.holguin.cu/2008/Abril/articulos/ARTI1.htm

[Matulevicius04a] Matulevičius R. (2004): Survey of Requirements Engineering Practice in Lithuanian Software Development Companies. Proceedings of the 13th International Conference on Information Systems Development (ISD 2004). Advances in Theory, Practice and Education. Pages 327-339.Kluwer Academic/Plenum Publishers, Vilnius, Lithuania

[McIlroy68] M.D. McIlroy, (1968): Mass-Produced Software Components,Software Engineering: Report on a Conference by the NATO Science Committee, P. Naur and B. Randell, eds., NATO Scientific Affairs Division, Brussels, 1968, pp. 138-150.

[McPhee02] McPhee, C., Eberlein, A. (2002): Requirements engineering for time-to-market projects. Engineering of Computer-Based Systems, Proceedings. Ninth Annual IEEE International Conference and Workshop. Pages 17 – 24

[McGregor10] McGregor, J. D., Muthig, D., Yoshimura, K., Jensen, P. (2010): Guest editors' introduction: Successful software product line practices. IEEE Software. Volume 27, issue 3, pages 16–21. IEEE Computer Society Press Los Alamitos

[Mctaggart91] Mctaggart, R. (1991): Principles of Participatory Action Research. Adult Education Quarterly. Page 41

[Mendoza06] Mendoza Palacios, R. (2006): Investigación cualitativa y cuantitativa. Diferencias y limitaciones. http://www.gycperu.com/descargas/005investigacion cuali cuanti diferencias y limitac.pdf

[Mitroff83] Mitroff, I.I. (1983): Stakeholders of the Organizational Mind. Jossey-Bass

[Mizuno94] Mizuno, S., Akao, Y., (1994): QFD, the customer-driven approach to quality planning and deployment. Tokio, ISBN: 92-833-1122-1

[Mohan10] Mohan, K., Ramesh, B., Sugumaran, V. (2010): Integrating software product line engineering and agile development. IEEE Software. Volume 27, issue 3, pages 48 –55. IEEE Computer Society Press Los Alamitos

[Montgomery96] Montgomery, D. C. and Runger, G. C.; Alt, F. B.; (1996): Controlling Multiple Stream Processes with Principal Components. International Journal of Production Research 34, pp. 2991–2999.

[Mullery79] Mullery, G.P. (1979): CORE-a method for controlled requirement specification. ICSE '79 Proceedings of the 4th international conference on Software engineering. Pages 126 - 135. IEEE Press Piscataway, NJ

[Myers02] Myers, M. D. (2002): Qualitative Research in Information Systems. MIS Quarterly, 21:2, pp 241-242, junio 1997. Recuperado de MISQ Discovery, http://www.auckland.ae.nz/msis/isworld/, última actualización junio de 2002

[Myers97] Myers, M. D. (1997): Qualitative Research in Information Systems. MIS Quarterly. Volume 21, issue 2, pages 241-242. MISQ Discovery http://www.auckland.ae.nz/msis/isworld/, última actualización junio de 2002

[Narayanaswamy92] Narayanaswamy, K., Goldman, N. (1992): Lazy Consistency: A Basis for Cooperative Software Development. ICSCW '92 Proceedings of the 1992 ACM conference on Computer-supported cooperative work. Pages 257-264. ACM New York, NY, ISBN:0-89791-542-9

[Natt05] Natt och Dag, J., Gervasi, V. (2005): Managing large repositories of natural language requirements. Engineering and managing software requirements. Pages 219-244. Berlin, Germany, Springer

[Nentwich03] Nentwich, C., Emmerich, W., Finkelstein, A., Ellmer, E. (2003): Flexible consistency checking. ACM Transactions on Software Engineering and Methodology (TOSEM). Volume 12, issue 1, pages 28 - 63. ACM New York, NY

[Newman95] Newman, W.M., Lamming, M.G. (1995): Interactive System Design. Addison-Wesley.

[NgoThe03] Ngo-The, A., Ruhe, G. (2003): Requirements Negotiation under Incompleteness and Uncertainty. Software Engineering Knowledge Engineering (SEKE 2003). San Francisco, CA,

[Nikula00] Nikula, U., Sajaniemi, J., Kälviäinen, H. (2000): A State-of-the-Practice Survey on Requirements Engineering in Small- and Medium-Sized Enterprises. TBRC Research Report 1, Telecom Business Research Center Lappeenranta, Lappeenranta University of Technology

[Noor08] Noor, M. A., Rabiser, R., Grunbacher, P. (2008): Agile product line planning: A collaborative approach and a case study. Journal of Systems and Software. Volume 81, issue 6, pages 868 – 882. Elsevier Science Inc. New York, NY

[Northrop04] Northrop, L. (2004): Software Product Line Adoption Roadmap, tech. report CMU/SEI-2004-TR-022. Software Eng. Inst., Carnegie Mellon Univ.
http://www.sei.cmu.edu/library/abstracts/reports/04tr022.cfm

[Northrop02] Northrop. L. (2002): SEI's Software Product Line Tenets. IEEE Software. Volume 19, issue 4, pages 32–40

[Novorita96] Novorita, R.J., Grube, G. (1996): Benefits of structured requirements methods for market-based enterprises. Proceedings of Sixth Annual International INCOSE Symposium. Seattle, WA: INCOSE

[Nuseibeh03] Nuseibeh, B. K., Finkelstein, A (2003): ViewPoints: meaningful relationships are difficult!ICSE 2003:676-683

[Nuseibeh00a] B. Nuseibeh, S. Easterbrook, and A. Russo. (2000): Leveraging inconsistency in software development. IEEE Computer, 33(4):24–29.

[Nuseibeh00] Nuseibeh, B., Easterbrook, S., Russo, A. (2000): Leveraging inconsistency in software development. IEEE Computer. Volume 33, issue 4 pages 24–29

[Nuseibeh00b] B. Nuseibeh and S. Easterbrook. (2000): Requirements engineering: a roadmap. In A. Finkelstein, editor,The Future of Software Engineering, Special Volume published in conjunction with ICSE 2000,.

[Nuseibeh97] Nuseibeh, B. (1997): To be and not to be: On managing inconsistency in software development. In Proceedings of 8th International Workshop on Software Specification and Design (IWSSD-8). Pages 164–169. IEEE CS Press

[Nuseibeh96] Nuseibeh, B., Finkelstein, B., Kramer, J A. (1996): Method Engineering for multi-perspective software development. Information and software technology. Volume 38, pages 267-274

[Nuseibeh94] Nuseibeh, B., Kramer, J., Finkelstein, A. (1994): A Framework for Expressing the Relationships Between Multiple Views in Requirements Specification. Transactions on Software Engineering. Volume 20, issue 10, pages 760-773. IEEE Press Piscataway

[Orasanu93] Orasanu, J. M. (1993): Lessons from research on expert decision making on the flight deck. ICAO Journal (Special issue on Human Factors in Aviation), volume 48, issue 7, pages 20-22. (Reprinted in the FAA Aviation News, pages 5-7). (Reprinted in C. Hutchings, Making the Hard Decisions, in the United Airlines Safetyliner, Volume 1

[Page98] Page, A.L., (1998): Assesing New Product Development Practices and Performance: Establishing Crucial Norms. Journal of Product Innovation Management. Volume 10, issue 4, pages 273-90

[Peña13] Peña Siles, J., González Benítez, A., Ruiz Enrique, D. (2013) :Un Marco Conceptual para la Elaboración de Cuadros de Mando 2.0 (Universidad de Sevilla). E20biz - 2º Congreso Nacional de Empresa 2.0 y Social Business

[Perry00] Perry, D., Porter, A. Votta, L. (2000): Empirical Studies of Software Engineering: A Roadmap. ICSE '00 Proceedings of the Conference on The Future of Software Engineering. Pages 345 - 355. ACM New York, NY, ISBN:1-58113-253-0

[pi12] Alateyah, S., Crowder, R. M., & Wills, G. B. (2012): Towards an integrated model for citizen adoption of E-government services.

[Piattini08] Piattini, M., Garzas, J., Cabrero, D. (2008): Priorización del Valor de Artefactos Software Basada en la Frecuencia de Uso. Jornadas de Ingeniería de software y Bases de Datos (JISBD'08). Gijón, Spain.

[Piattini07] Piattini, M., Garzas, J., Cabrero, D. (2007): La Ingeniería de software Basada en Valor. CUORE-Circulo de Usuarios de Oracle de España. Volume 34, pages 3 -12

[PM96] The Fourth Framework Programm (1996). http://ec.europa.eu/research/fp4.html

[Pohl97] Pohl, K. (1997): Requirements Engineering: An Overview. Encyclopedia of Computer Science and Technology. Volume 36 http://sunsite.informatik.rwth-aachen.de/CREWS/reports96.htm

[Pohl94] Pohl, K. (1994): The three dimensions of Requirements Engineering. A Framework and its Applications. Information Systems. Volume 19, issue 3, pages 243 - 258. Elsevier Science Ltd. Oxford, UK

[Potts95] Potts, C. (1995): Invented Requirements and Imagined Customers: Requirements Engineering for Off-the-Shelf Software. RE '95 Proceedings of the Second IEEE International Symposium on Requirements Engineering. Pages 128–130. IEEE Computer Society Washington, DC, ISBN:0-8186-7017-7

[Potts94] Potts, C., Takahashi, K., Anton, A. (1994): Inquiry-Based requirements Analysis. IEEE Software,. Volume 11, issue 2, pages

21-32. IEEE Computer Society Press Los Alamitos, CA

[Potts94a] Potts, C., Takahashi, K., Anton, A. (1994): Inquiry-Based scenario Analysis of system requirements. Informe tecnico CIT-CC-94/14, Georgia Institute of Technology. www.cc.gatech.edu/computing/SW_Eng

[Pouloudi97] Pouloudi, A., Whitley, E.A. (1997): Stakeholder Identification in Inter-Organisational Systems: Gaining Insights for Drug Use Management Systems. European J. Information Systems. Volume 6, issue 1, pages 1–14

[Pouloudi97a] Pouloudi, A. (1997): Stakeholder Analysis as a Front- End to Knowledge Elicitation. AI & Society - Special double issue on knowledge, elicitation, representation and application. Volume 11, pages 122-137. Springer-Verlag London, UK

[Pressman01b] Pressman, R. S. (2001): Software Engineering: A Practitioner's Approach. McGraw-Hill, Inc. New York, NY, ISBN:007301933X 9780073019338

[Regnell01] Regnell, B., Höst, M., Natt och Dag, J., Beremark, P., Hjelm ,T. (2001): An industrial case study on distributed prioritization in market-driven requirements engineering for packaged software. Requirements Engineering. Volume 6, issue 1, pages 51-62

[Robertson99] Robertson, S., Robertson, J. (1999): Mastering the Requirements Process. Harlow, England: Addison-Wesley.

[Robinson99] Robinson, W., Pawlowski, S. (1999): Managing requirements inconsistency with development goal monitors. EEE Transactions on Software Engineering. Volume 25, issue 6, pages 816–835. IEEE Press Piscataway, NJ

[Ross77] Ross, D., Schoman, K.E. (1977): Structured Analysis for Requirements Definition. IEEE Trans. Software Engineering. Volume 3, issue 1, pages 6-15

[Rumbaugh91] Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., Lorensen, W. (1991): Object-oriented Modeling and Design. Prentice-Hall, Englewood Cliffs, N.J

[Sabetzade03] Sabetzadeh, M., Easterbrook, S. M. (2003): Analysis of inconsistency in graph-based viewpoints: A category-theoretic approach. In 18th IEEE Int. Conf. on Automated Software Engineering

[Sawyer05] Sawyer, P. (2005): Maturing requirements engineering process maturity models. Requirements engineering form sociotechnical systems. Pages 84-99. Hershey, PA, USA: Information Science Publishing

[Sawyer01] Sawyer, P., Kotonya, G. (2001): Software Requirements. Guide to the Software. Engineering Body of Knowledge, Trial Version, IEEE

[Sawyer00] Sawyer, P. (2000): Packaged Software: Challenges for RE. In Proceedings of Sixth International Workshop on Requirements Engineering: Foundation for Software Quality. Pages 137–142. Essen, Germany: Essener Informatik Beiträge

[Sawyer99a] Sawyer, P., Kotonya, G. (1999): SWEBOK: Software Requirements Engineering Knowledge Area Description. Informe técnico Versión 0.5, SWEBOK Project. http://www.swebok.org

[Schmid04] Schmid, K., John, I. (2004): A Customizable Approach to Full Lifecycle Variability Management. Science of Computer Programming. Volume 53, issue 3, pages 259–284. Elsevier North-Holland, Inc. Amsterdam

[Schmid02] Schmid, K., Verlage M. (2002): The Economic Impact of Product Line Adoption and Evolution. IEEE Software. Volume 10, issue 4, pages 50–57

[Schwanke88] Schwanke, R. W., Kaiser, G. E. (1988): Living With Inconsistency in Large Systems. Proceedings of the International Workshop on Software Version and Configuration Control. Pages 98-118. Grassau, Germany

[Seaman99] Seaman, C. B. (1999): Qualitative Methods in Empirical Studies of Software Engineering. IEEE Transactions on Software Engineering. Volume 25, issue 4, pages 557-572. IEEE Press Piscataway, NJ

[SEI01] Software Engineering Institute. (2001). A Framework for Software Product Line – Version 3.0. Product Line Systems Program, Software engineering Institute. Carnegie Mellon University, Pittsburgh, Pennsylvania 15213 (USA). http://www.sei.cmu.edu/plp/framework.html

[Sharp99] Sharp, H., Finkelstein, A., Galal, G. (1999): Stakeholder Identification in the Requirements Engineering Process. DEXA '99 Proceedings of the 10th International Workshop on Database & Expert Systems Applications . Pages 387–391. IEEE Computer Society Washington, ISBN:0-7695-0281-4

[Sinnema07] Sinnema, M., Deelstra, S. (2007): Classifying Variability Modeling Techniques. Information and Software Technology. Volume 49, issue 7, pages 717–739

[Solheim05] Solheim, H., Lillehagen, F., Petersen, S., Jorgensen, H., Anastasiou, M. (2005): Model-driven visual requirements engineering.RE '05 Proceedings of the 13th IEEE International Conference on Requirements Engineering. Pages 421-428. IEEE Computer Society Washington,ISBN:0-7695-2425-7

[Sommerville97] Sommerville, I., Sawyer, P. (1997): Requirements.Engineering: A Good Practice Guide.John Wiley & Sons, Inc. New York

[Sorli08] QFD monograph on a series of Quality tools by AEC (2008)

[Sorli09] Innovating in Product/Process Development (Springer 2009)

[Spanoudakis99] Spanoudakis, G., Finkelstein, A.,Till. D. (1999): Overlaps in Requirements Engineering. Autom. Softw. Eng. Volume 6, issue 2, pages 171-198. Kluwer Academic Publishers Hingham

[Spanoudakis97] Spanoudakis, G.,Finkelstein, A. (1997): Reconciling Requirements: A Method for Managing Interference, Inconsistency and Conflict. Annals of Software Engineering. Volume 3, pages 433-457

[Standishgroup12] The Standish Group (2012): The CHAOS Report, www.standishgroup.com

[Standishgroup09] The Standish Group (2009): The CHAOS

Report, www.standishgroup.com

[Standishgroup95] The Standish Group (1995): The CHAOS Report, www.standishgroup.com

[Steffen00] Steffen, T.,Peruzzi, F. (2000): Starting a product line approach for an envisioned market. In Patric Donohoe, editor, Software Product Lines, Experience and Research Directions. Pages 495–512. Kluwer Academic Publishers

[Straub04] Straub, D., Gefen, D., Boudreau, M.C. (2004): The ISWorld Quantitative, Positivist Research Methods Website Available at http://dstraub.cis.gsu.edu:88/quant/

[Sutcliffe02] Sutcliffe, A. (2002): User-centred requirements engineering: Theory and practice. Springer-Verlag New York, ISBN:1852335173

[Svahnberg00] Svahnberg, M., Bengtsson, P. (2000): Software Product Lines from Customer to Code. Department of Software Engineering and Computer Science. University of Karlskrona/Ronneby, Sweden.

[SWEBOK04] SWEBOK (2004): http://www.swebok.org

[Szyperski98] Szyperski, C. (1998): Component Software – Beyond Object-Oriented Programming. ACM Press/Addison-Wesley Publishing Co. New York, ISBN:0-201-17888-5

[Takebe09] Y. Takebe et al. (2009): Experiences with Software Product Line Engineering in Product-Development-Oriented Organizations, SPLC '09 Proceedings of the 13th International Software Product Line Conference. Pages 275-283. Carnegie Mellon University Pittsburgh, PA

[Thackery90] Thackery, R., Van Treeck, G. (1990): Applying Quality Function Deployment for Software Product Development. Journal of Engineering Design. Volume 1, issue 4, pages 389-410

[Thiel02] Theil, S., Hein, A. (2002): Modeling and Using Product Line Variability in Automotive Systems. IEEE Software. Volume 19, issue 4, pages 66–72. IEEE Computer Society Press Los Alamitos

[Tian06] Tian, K., Cooper, K. (2006): Agile and Software Product Line Methods: Are They So Different? 1st International Workshop on Agile Product Line Engineering (APLE'06), IEEE

[Toft04] Toft, P. (2004): Hewlett-Packard: The HP Owen Firmware Cooperative—A Software Product Line Success Story. Software Product Lines. www.softwareproductlines.com/successes/hp.html.

[Toft00] Toft, P., Coleman, D., Ohta, J. (2000): HP Product Generation Consulting, A Cooperative Model for Cross-Divisional Product Development for a Software Product Line. Proc. 1st Software Product Lines Conference (SPLC 1). Pages 111–132

[Troya99] J. M. Troya and A. Vallecillo. (1999): Specifying Reusable Controllers for Software Components. In Proc. Of FMOODS'99, Kluwer Academic Press, Florence, February 1999.

[Urban93] Urban, G., Hauser, J. (1993): Design and Marketing of New Products. Prentice-Hall, 2ª ed., New Jersey.

[Wadsworth09] Wadsworth, Y. (2009): What is participatory Action Research? Action Research Internation, paper 2

[Weiss99] Weiss, D.M., Lai, C.T.R. (1999): Software Product-Line Engineering: A family-Based Software Development Process. Addison-Wesley Longman Publishing Co., Inc. Boston, ISBN:0-201-69438-7

[Wieger03] Wiegers, K.E. (2003): Software Requirements, 2nd Edition. USA, Microsoft Press.

[Wieger00] Wiegers, K. E. (2000): When Telepathy Won't Do: Requirements Engineering Key Practices. Originally published in Cutter IT Journal available
at http://www.processimpact.com/articles/telepathy.htm

[Wiegers99] Wiegers, K .(1999): Software requirements. Microsoft Press, Redmond

[Wiegers99a] Wieger, K. E. (1999): First things first: Prioritizing Requirements. Software Development.

[WirfsBrock90] Wirfs-Brock, R.J., Johnson, R.E. (1990): Surveying Current Research in Object-Oriented Design. Communications of the ACM. Volume 33, issue 9, pages 105- 124

[Withall07] Withall, S. (2007): Software requirement patterns. Microsoft Press Redmond, WA, USA

[Wood85] Wood-Happer, T. (1985): Research Methods in Information Systems: Using Action Research. Research Methods in Information Systems. Pages 169-191. Amsterdam: North-Holland

[Wohlin00] Wohlin, C., Runeson, P., Höst, M., Ohlson, M., Regnell, B., Wesslén A. (2000): Experimentation in Software Engineering: An Introduction. Kluwer Academic Publishers Norwell, ISBN:0-7923-8682-5

[Woolridge07] Woolridge, R.W., McManus, D.J., Hale, J.E. (2007): Stakeholder Risk Assessment: An outcome-based approach. IEEE software. Volume 24, issue 2, pages 36-45. IEEE Computer Society Press Los Alamitos

[Xie09] Xie, H., Liu, L., Yang, J. (2009): i*-prefer: optimizing requirements elicitation process based on actor preferences. SAC ’09: Proceedings of the 2009 ACM symposium on Applied Computing. Pages 347–354. ACM New York, NY, ISBN: 978-1-60558-166-8

[Yeh92] Yeh, A. (1992): Requirements Engineering Support Technique (REQUEST) – A Market Driven Requirements Management Process. Proceedings of the Second Symposium on Assessment of Quality Software Development Tools. Pages 211–223. IEEE Computer Society Press, Los Alamitos, CA

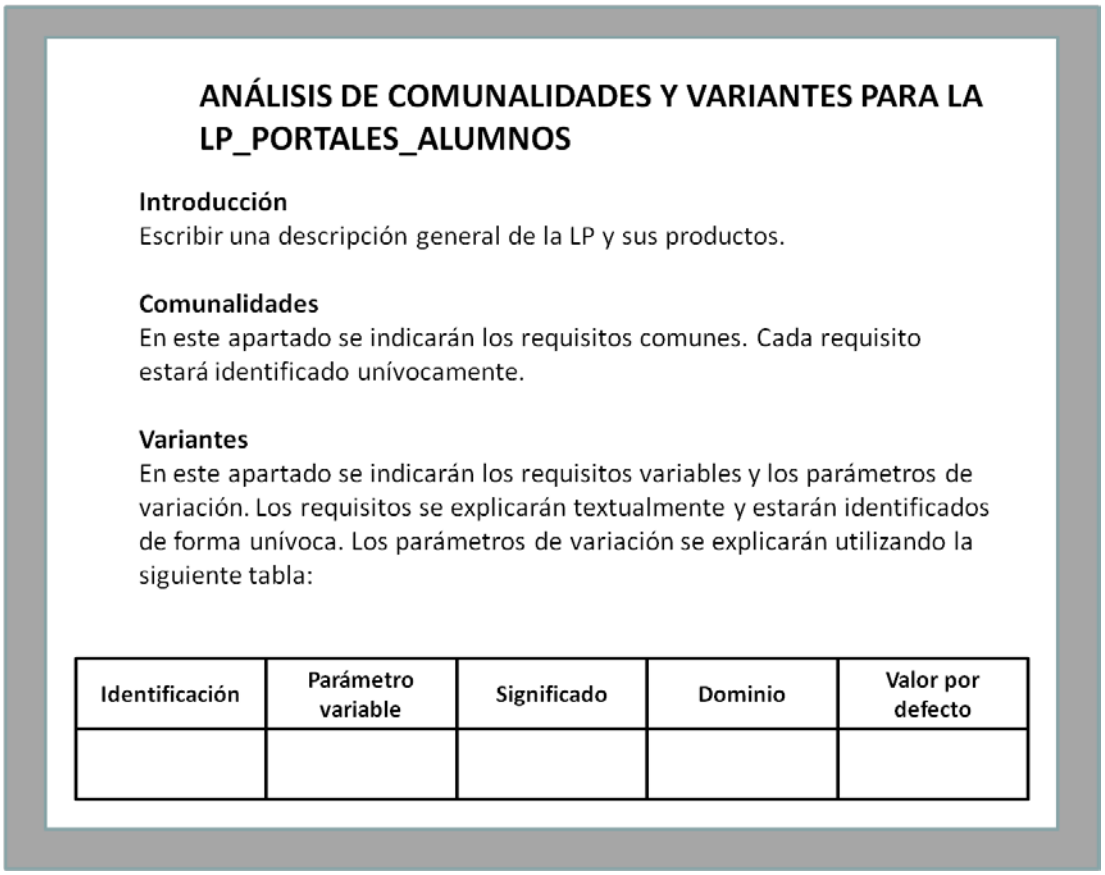
[Yoshimura08] Yoshimura, K. et al. (2008): Factor Analysis Based Approach for Detecting Product Line Variability from Change History.MSR '08 Proceedings of the 2008 international working conference on Mining software repositories. Pages 11-18. ACM New York, NY, ISBN: 978-1-60558-024-1

[Zave97] Zave, P. (1997): Classification of research efforts in requirements engineering. ACM Computing Surveys. Volume 29, issue 4, pages 315-321.ACM New York, NY

[Zowghi05] Zowghi, D., Coulin, C. (2005): Requirements elicitation: A survey of techniques, approaches, and tools. Engineering and managing software requirements. Pages 21-46. Berlin, Germany: Springer

ANEXO A

Plantilla ERS_LP



ANÁLISIS DE COMUNALIDADES Y VARIANTES PARA LA LP_PORTALES_ALUMNOS

Introducción
Escribir una descripción general de la LP y sus productos.

Comunalidades
En este apartado se indicarán los requisitos comunes. Cada requisito estará identificado unívocamente.

Variantes
En este apartado se indicarán los requisitos variables y los parámetros de variación. Los requisitos se explicarán textualmente y estarán identificados de forma unívoca. Los parámetros de variación se explicarán utilizando la siguiente tabla:

Identificación	Parámetro variable	Significado	Dominio	Valor por defecto

ANEXO B

Enunciados utilizados en el experimento formal

Enunciado planteado al grupo que aplicó el método ELVIRA

Experimento – Método ELVIRA

Enunciado Ejercicio

Tras terminar vuestros estudios decidís crear una empresa de desarrollo software. Esta empresa estará especializada en el desarrollo en entornos Web.

Durante los primeros meses recibís varias ofertas para desarrollar portales para alumnos universitarios. En ese momento os planteáis desarrollar una línea de productos de portales Web para estudiantes universitarios.

Como primera tarea decidís analizar el dominio, y realizar las actividades de Ingeniería de Requisitos pertinentes. Como resultado se obtiene un documento ERS que incluye el análisis de comunalidades y variantes.

Se pide:

- Escribir ese documento (ERS_LP) con el siguiente formato:
- Para llegar a escribir este documento, se debe utilizar el método ELVIRA.

Normas para la realización del ejercicio

- Grupos de 3-4 personas
- Lenguaje para la realización del ejercicio: Castellano
- Tiempo máximo para la realización del ejercicio: 4 horas
- El documento resultante se dejará en red:
  Ikasleak\Ikasmailak\IngInformatika\4_A\IngenieriaSoftwareI\ERS_LP
  Dentro de esta carpeta cada grupo dejará su documento. Cuidado con el nombre del documento. No "machaquéis" el documento de otro grupo.

Enunciado planteado al grupo que aplicó el método ad-hoc

Experimento – Método Ad-hoc

Enunciado Ejercicio

Tras terminar vuestros estudios decidís crear una empresa de desarrollo software. Esta empresa estará especializada en el desarrollo en entornos Web.

Durante los primeros meses recibís varias ofertas para desarrollar portales para alumnos universitarios. En ese momento os planteáis desarrollar una línea de productos de portales Web para estudiantes universitarios.

Como primera tarea decidís analizar el dominio, y realizar las actividades de Ingeniería de Requisitos pertinentes. Como resultado se obtiene un documento ERS que incluye el análisis de comunalidades y variantes.

Se pide: Escribir ese documento (ERS_LP) con el siguiente formato:

Normas para la realización del ejercicio

- Grupos de 3-4 personas
- Lenguaje para la realización del ejercicio: Castellano
- Tiempo máximo para la realización del ejercicio: 4 horas
- El documento resultante se dejará en red:
  Ikasleak\Ikasmailak\IngInformatika\4_A\IngenieriaSoftwareI\ERS_LP
  Dentro de esta carpeta cada grupo dejará su documento. Cuidado con el nombre del documento. No "machaquéis" el documento de otro grupo.