Especificación de Requisitos Software según el estándar de IEEE 830

Índice

1 Introducción

El análisis de requisitos es una de las tareas más importantes en el ciclo de vida del desarrollo de software, puesto que en ella se determinan los "planos" de la nueva aplicación.

En cualquier proyecto software los requisitos son las necesidades del producto que se debe desarrollar. Por ello, en la fase de análisis de requisitos se deben identificar claramente estas necesidades y documentarlas. Como resultado de esta fase se debe producir un documento de especificación de requisitos en el que se describa lo que el futuro sistema debe hacer. Por tanto, no se trata simplemente de una actividad de análisis, sino también de síntesis.

El análisis de requisitos se puede definir como el proceso del estudio de las necesidades de los usuarios para llegar a una definición de los requisitos del sistema, hardware o software, así como el proceso de estudio y refinamiento de dichos requisitos, definición proporcionada por el IEEE [Piattini, 1996]. Asimismo, se define requisito como una condición o capacidad que necesita el usuario para resolver un problema o conseguir un objetivo determinado [Piattini, 1996]. Esta definición se extiende y se aplica a las condiciones que debe cumplir o poseer un sistema o uno de sus componentes para satisfacer un contrato, una norma o una especificación.

En la determinación de los requisitos no sólo deben actuar los analistas, es muy importante la participación de los propios usuarios, porque son éstos los que mejor conocen el sistema que se va a automatizar. Analista y cliente se deben poner de acuerdo en las necesidades del nuevo sistema, ya que el cliente no suele entender el proceso de diseño y desarrollo del software como para redactar una especificación de requisitos software (ERS) y los analistas no suelen entender completamente el problema del cliente, debido a que no dominan su área de trabajo.

Así pues, el documento de especificación de requisitos debe ser legible por el cliente, con lo que se evita el malentendido de determinadas situaciones, ya que el cliente participa activamente en la extracción de dichos requisitos.

Basándose en estos requisitos, el ingeniero de software procederá al modelado de la futura aplicación. Para ello, se pueden utilizar diferentes tipos de metodologías entre las que destacan la metodología estructurada y la metodología orientada a objetos (por ejemplo DFDs y UML respectivamente).

La metodología estructurada está basada en la representación de las funciones que debe realizar el sistema y los datos que fluyen entre ellas.

En la metodología orientada a objetos se utiliza el UML [Pierre-Alain, 1997], mediante el cual podemos representar diagramas (casos de uso) que permiten definir el sistema desde el punto de vista del usuario estableciendo las relaciones entre el futuro sistema y su entorno. Estas relaciones se establecen en forma de acciones del usuario y reacciones del sistema.

2 Objetivos de la ERS.

Los principales objetivos que se identifican en la especificación de requisitos software son [Chalmeta, 2000]:

1. Ayudar a los clientes a describir claramente lo que se desea obtener mediante un determinado software: El cliente debe participar activamente en la especificación de requisitos, ya que éste tiene una visión mucho más detallada de los procesos que se llevan a cabo. Asimismo, el cliente se siente partícipe del propio desarrollo.

2. Ayudar a los desarrolladores a entender qué quiere exactamente el cliente: En muchas ocasiones el cliente no sabe exactamente qué es lo que quiere. La ERS permite al cliente definir todos los requisitos que desea y al mismo tiempo los desarrolladores tienen una base fija en la que trabajar. Si no se realiza una buena especificación de requisitos, los costes de desarrollo pueden incrementarse considerablemente, ya que se deben hacer cambios durante la creación de la aplicación.

3. Servir de base para desarrollos de estándares de ERS particulares para cada organización: Cada entidad puede desarrollar sus propios estándares para definir sus necesidades.

Una buena especificación de requisitos software ofrece una serie de ventajas entre las que destacan el contrato entre cliente y desarrolladores (como ya se ha indicado con anterioridad), la reducción del esfuerzo en el desarrollo, una buena base para la estimación de costes y planificación, un punto de referencia para procesos de verificación y validación, y una base para la identificación de posibles mejoras en los procesos analizados.

La ERS es una descripción que debe decir ciertas cosas y al mismo tiempo debe decirlas de una determinada manera. En este documento se presentará una de las formas que viene especificada por el estándar IEEE 830.

Una ERS forma parte de la documentación asociada al software que se está desarrollando, por tanto debe definir correctamente todos los requerimientos, pero no más de los necesarios. Esta documentación no debería describir ningún detalle de diseño, modo de implementación o gestión del proyecto, ya que los requisitos se deben describir de forma que el usuario pueda entenderlos. Al mismo tiempo, se da una mayor flexibilidad a los desarrolladores para la implementación.

Así pues, el grado y el lenguaje utilizado para la documentación de los requisitos estarán en función del nivel que el usuario tenga para entender dichas especificaciones.

3 Características de una buena ERS

Las características deseables para una buena especificación de requisitos software que se indican en el IEEE son las siguientes [Chalmeta, 2000][Piattini, 1996]:

- Correcta
- No ambigua
- Completa
- Verificable
- Consistente
- Clasificada
- Modificable
- Explorable
- Utilizable durante las tareas de mantenimiento y uso

3.1 Corrección

La ERS es correcta si y sólo si todo requisito que figura en ella refleja alguna necesidad real. La corrección de la ERS implica que el sistema implementado será el sistema deseado.

3.2 Ambigüedad

Un documento es no ambiguo si y solo si cada requisito descrito tiene una única interpretación. Cada característica del producto final debe ser descrita utilizando un término único y, en caso de que se utilicen términos similares en distintos contextos, se deben indicar claramente las diferencias entre ellos. Incluso se puede incluir un glosario en el que indicar cada significado específicamente.

Los analistas deben poner un cuidado especial a la hora de especificar los requisitos. El hecho de utilizar el lenguaje natural para hacer la ERS comprensible a los usuarios supone un riesgo muy elevado, porque el lenguaje natural puede llegar a ser muy ambiguo.

Ejemplo:

Todos los clientes tienen el mismo campo de control

1.- ¿Todos tienen el mismo valor en el campo de control?

2.- ¿Todos los campos de control tienen el mismo formato?

3.- ¿Un campo de control se usa para todos los clientes?

En términos generales, el lenguaje natural es de los más ambiguos. Por el contrario existen los lenguajes formales que no son ambiguos, pero son más difíciles de aprender y menos comprensibles para el que no los conoce.

3.3 Completitud

Una ERS es completa si:

- Incluye todos los requisitos significativos del software (relacionados con la funcionalidad, ejecución, diseño, atributos de calidad o interfaces externas).
- Existe una definición de respuestas a todas las posibles entradas, tanto válidas como inválidas, en todas las posibles situaciones.
- Cumple con el estándar utilizado. Si hay alguna parte del estándar que no se utiliza, se debe razonar suficientemente el porqué no se ha utilizado dicho apartado.
- Aparecen etiquetadas todas las figuras, tablas, diagramas, etc, así como definidos todos los términos y unidades de medida empleados.

La ERS debe ser siempre completa, aunque en ocasiones esto no será posible. Por ejemplo si todavía no se han determinado los formatos de los informes finales o por cualquier razón se esta esperando la publicación de un Real Decreto o un reglamento sobre impuestos.

3.4 Verificabilidad

Un requisito se dice que es verificable si existe algún proceso no excesivamente costoso por el cual una persona o una máquina pueda chequear que el software satisface dicho requerimiento.

Ejemplo

No verificables:

El producto debería funcionar bien

El producto debería tener una buena interfaz de usuario

Verificable:

La salida se suministra dentro de los 20 segundos siguientes al evento E el 60% de las veces, y en los 30 segundos siguientes en el 100%

3.5 Consistencia

Una ERS es consistente si y sólo si ningún conjunto de requisitos descritos en ella son contradictorios o entran en conflicto. Se pueden dar tres casos:

- Requisitos que describen el mismo objeto real utilizando distintos términos.
- Las características especificadas de objetos reales. Un requisito establece que todas las luces son verdes y otro que son azules.
- Conflicto lógico o temporal entre dos acciones determinadas. Se llega a un punto en el que dos acciones serían perfectamente válidas (¿sumar o multiplicar?)

3.6 Clasificación

No todos los requisitos son igual de importantes. Los requisitos pueden clasificarse por diversos criterios:

- Importancia: Pueden ser esenciales, condicionales u opcionales.
- Estabilidad: Cambios que pueden afectar al requisito.

Lo ideal es el establecimiento de prioridades, de modo que la implementación de un requisito de menor prioridad no emplee excesivos recursos.

3.7 Modificabilidad

Una ERS es modificable si cualquier cambio puede realizarse de manera fácil, completa y consistente. Para ello, es deseable tener una organización coherente y fácil de usar en la que aparezca el índice o una tabla de contenidos fácilmente accesible.

También es deseable evitar la redundancia, es decir que no aparezca un mismo requisito en más de un lugar de la ERS. No es un error, pero si se tiene que modificar alguna cosa será mucho más cómodo si no tenemos que buscar el mismo requisito en varios lugares.

3.8 Explorabilidad (traceability)

Una ERS es explorable si el origen de cada requerimiento es claro tanto hacia atrás (origen que puede ser un documento, una persona etc.) como hacia delante (componentes del sistema que realizan dicho requisito).

Cuando un requisito de la ERS representa un desglose o una derivación de otro requisito, se debe facilitar tanto las referencias hacia atrás como hacia adelante en el ciclo de vida.

Las referencias hacia delante de la ERS son especialmente importantes para el mantenimiento del software. Cuando el código y los documentos son modificados, es esencial poder comparar el conjunto total de requisitos que puedan verse afectados por estas modificaciones.

3.9 Utilizable durante las tareas de mantenimiento y uso

En la ERS también se deben tener en cuenta las necesidades de mantenimiento. El personal que no ha intervenido directamente en el desarrollo debe ser capaz de encargarse de su mantenimiento. Así, dicha ERS actúa a modo de plano de la aplicación, permitiendo incluso modificaciones que no requieran un cambio en el diseño.

En ocasiones, el equipo de desarrollo supone unos conocimientos que el personal que se encargue del mantenimiento no tiene por qué tener. Por esta razón es necesaria una correcta documentación de las funciones, ya que si no se conoce en detalle su origen, difícilmente podrán ser modificadas.

4 Esquema de la ERS definida en el IEEE 830-1998

La siguiente figura muestra la estructura de la ERS propuesta por el IEEE en su estándar 830 [IEEE, 1998] [upm, 2000].

1 Introducción
- 1.1 Propósito
- 1.2 Ámbito del Sistema
- 1.3 Definiciones, Acrónimos y Abreviaturas
- 1.4 Referencias
- 1.5 Visión general del documento

2 Descripción General
- 2.1 Perspectiva del Producto
- 2.2 Funciones del Producto
- 2.3 Características de los usuarios
- 2.4 Restricciones
- 2.5 Suposiciones y Dependencias
- 2.6 Requisitos Futuros

3 Requisitos Específicos
- 3.1 Interfaces Externas
- 3.2 Funciones
- 3.3 Requisitos de Rendimiento
- 3.4 Restricciones de Diseño
- 3.5 Atributos del Sistema
- 3.6 Otros Requisitos

4 Apéndices

5 Índice

Figura 1. Estructura de una ERS

A continuación se describirá brevemente cada uno de los apartados que se definen en el estándar estudiado.

1 Introducción

En esta sección se proporcionará una introducción a todo el documento de Especificación de Requisitos Software. Consta de varias subsecciones, las cuales son propósito, ámbito del sistema, definiciones, referencias y visión general del documento.

1.1 Propósito

Se definirá el propósito del documento ERS y se especificará a quién va dirigido el documento.

1.2 Ámbito del Sistema

En esta subsección se pondrá nombre al futuro sistema, se explicará lo que el sistema hará y lo que no hará, se describirán los beneficios, objetivos y metas que se espera alcanzar con el futuro sistema y se mantendrán referencias a los documentos de nivel superior que puedan existir.

1.3 Definiciones, Acrónimos y Abreviaturas.

Se definirán aquí todos los términos, acrónimos y abreviaturas utilizadas en el desarrollo de la ERS.

1.4 Referencias

Se presentará una lista completa de todos los documentos referenciados en la ERS.

1.5 Visión General del Producto

Esta subsección describirá brevemente los contenidos y la organización del resto de la ERS.

2 Descripción General

En esta sección se describen todos aquellos factores que afectan al producto y a sus requisitos. En esta sección no se describen los requisitos, sino su contexto. Los detalles de los requisitos de describen en la sección 3, detallándolos y haciendo más fácil su comprensión.

Normalmente podemos encontrar las siguientes subsecciones: Perspectiva del producto, funciones del producto, características de los usuarios, restricciones, suposiciones y futuros requisitos.

2.1 Perspectiva del Producto

Esta subsección debe relacionar el futuro sistema con otros productos. Así pues, podríamos dividir ésta en pequeñas subsecciones indicando cada uno de los puntos a tener en cuenta:

2.1.1. Indicar si es un producto independiente o parte de un sistema mayor
2.1.2. Interfaces de sistema
2.1.3. Interfaces de usuario
  2.1.3.1. Características lógicas del interfaz
  2.1.3.2. Cuestiones de optimización del interfaz de usuario
2.1.4. Interfaces hardware
2.1.5. Interfaces software
  2.1.5.1. Descripción del producto software utilizado
  2.1.5.2. Propósito del interfaz
  2.1.5.3. Definición del interfaz: contenido y formato
2.1.6. Interfaces de comunicaciones
2.1.7. Limitaciones de memoria
2.1.8. Operaciones
  2.1.8.1. Modos de operación de los distintos grupos de usuarios
  2.1.8.2. Periodos de operaciones interactivas y automáticas
  2.1.8.3. Funciones respaldo del procesamiento de datos
  2.1.8.4. Operaciones de backup y recuperación
2.1.9. Requerimientos para adaptarse a la ubicación
2.1.9.1. Indicar cualquier dato o secuencia de inicialización específico de cualquier lugar, modo de operación.
2.1.9.2. Características que deben ser modificadas para una instalación en particular.

2.2 Funciones del Producto

Esta subsección debe proporcionar un resumen de las funciones principales que el software debe llevar a cabo. Las funciones deben estar organizadas de manera que el cliente o cualquier otra persona lo entienda perfectamente. Para ello se pueden utilizar métodos textuales o gráficos, siempre que dichos gráficos reflejen las relaciones entre funciones y no el diseño del sistema.

En la metodología estructurada se podrían utilizar los DFDs y en una metodología orientada a objetos, el funcionamiento y las relaciones del futuro sistema se modelarían a través de los Casos de Uso. En ellos se representa lo que el usuario ve del sistema, así pues facilitará en gran medida su comprensión, siempre y cuando en los diagramas se eviten las ambigüedades.

2.3 Características de los usuarios

Se indica aquí el tipo de usuario al que se dirige la aplicación, así como su experiencia técnica, nivel de conocimientos, etc.

2.4 Restricciones

Se debe indicar aquí cualquier tipo de limitación como pueden ser políticas de la empresa, limitaciones hardware, seguridad, protocolos de comunicación, interfaces con otras aplicaciones, estándares de la empresa en cuanto a interfaces, etc. Serán las limitaciones que se imponen sobre los desarrolladores del producto.

2.5 Suposiciones y Dependencias

En este apartado aparecerá cualquier factor, que si cambia puede afectar a los requerimientos. No son restricciones de diseño, por ejemplo, asumir que un determinado sistema operativo estará disponible, presuponer una cierta organización de las unidades de la empresa. Si cambian ciertos detalles puede ser necesario revisar los requisitos.

2.6 Requisitos Futuros

Se indican aquí posibles mejoras del sistema en el futuro. Estas mejoras deben estudiarse y analizarse una vez concluido y puesto en marcha el sistema. Son modificaciones a realizar en un futuro incierto.

3 Requisitos Específicos

Esta sección de la especificación de requisitos software contiene todos los requerimientos hasta un nivel de detalle suficiente para permitir a los diseñadores diseñar un sistema que satisfaga dichos requisitos, y que permita diseñar las pruebas que ratifiquen que el sistema cumple con las necesidades requeridas.

Los requisitos que se aquí se indiquen deben describir comportamientos externos del sistema, observables por el usuario así como por parte de los operadores y otros sistemas.

Puesto que deben indicarse todos los requisitos, esta sección es la más larga de la ERS y debe cumplir los principios descritos en los primeros apartados de este informe. Estos principios son la corrección, no ambigüedad, completitud, consistencia, clasificación, verificabilidad, modificabilidad, explorabilidad y facilidad de mantenimiento.

Asimismo, éste documento debe ser perfectamente legible por el cliente y por personas de muy distinta formación. Otra de las cuestiones a tener en cuenta en esta sección es la identificación de cada uno de los requisitos mediante algún código o sistema de numeración.

3.1 Interfaces Externas

En esta subsección se definirán los requisitos que afecten a la interfaz de usuario e interfaz con otros sistemas (hardware y software), así como a interfaces de comunicaciones.

3.2 Funciones

En este subsección de deben especificar todas aquellas acciones o funciones que deberá llevar a cabo el sistema a desarrollar. Las acciones que se indican como "el sistema deberá ..." son las que deben incluirse en este apartado.

La estructuración de las funciones a desarrollar por el nuevo sistema no está del todo claro. Se debe tener en cuenta que en el estándar de IEEE 830 de 1983 se establecía que las funciones de deberían expresar como una jerarquía funcional (véase anexo II), puesto que es la que mejor se adaptaba a los DFDs que proponía el análisis estructurado. Con la evolución de la programación y los nuevos métodos de análisis se puede observar como esta estructura no se adapta, por tanto es necesaria la modificación de los estándares.

El estándar IEEE 830, en sus últimas versiones, permite la organización de esta subsección de múltiples formas y simplemente sugiere alguna manera para hacerlo, dejando la oportunidad de utilizar cualquier otra justificando suficientemente la utilización de ésta.

Alguna de las formas sugeridas por el estándar son:

- Por tipo de usuario: Distintos usuarios poseen distintos requisitos. Para cada clase de usuario que exista en la organización, se especifican los requisitos funcionales que le afecten o tengan mayor relación con sus tareas.
- Por objetos: Los objetos son entidades del mundo real que son reflejadas en el sistema. Por tanto, para cada objeto se detallan sus atributos y sus funciones. Los objetos se pueden agrupar en clases. A pesar de realizar el análisis con objetos no obliga a que el diseño del sistema siga el paradigma Orientado a Objetos, aunque lo facilita en gran medida.
- Por objetivos: un objetivo es un servicio que se desea que ofrezca el sistema y que requiere una determinada entrada para obtener su resultado. Para cada objetivo o subobjetivo requerido al sistema, se detallarán las funciones que permitan llevarlo a cabo.
- Por jerarquía funcional: La funcionalidad del sistema se especifica como una jerarquía de funciones que comparten entradas, salidas o datos del propio sistema. Para cada función y subfunción del sistema se detallará la entrada, el proceso en el que interviene y la salida. Normalmente este tipo de análisis implica que el diseño siga el paradigma de diseño estructurado. Por lo general éste sistema se utiliza cuando ninguno de los anteriores se puede aplicar.

Como se puede apreciar, el estándar propone una serie de plantillas según el tipo de sistema con el que nos enfrentemos. Pero en muchas ocasiones la elección se realiza por eliminación, o lo que es lo mismo, se escoge aquel que mejor se adapta a lo que se busca.

3.3 Requisitos de Rendimiento

En esta subsección se incluyen los requisitos relacionados con la carga que se espera que tenga que soportar el sistema (número de usuarios simultáneos, número de terminales ...). Asimismo, se pueden incluir los requisitos que afecten a la información que se vaya a guardar en la base de datos (cantidad de registros en una base de datos, frecuencia de uso...)

3.4 Restricciones de Diseño

Se incluyen aquí todas las restricciones que afecten al diseño de la aplicación, como pueden ser estándares internos de la organización, limitaciones hardware, etc.

3.5 Atributos del Sistema

Se detallarán atributos como la fiabilidad, mantenibilidad, seguridad, mecanismos de acceso restringido (password), usuarios autorizados a realizar ciertas tareas críticas ...

3.6 Otros requisitos

Aquellos requerimientos que no se hayan podido incluir en ninguna de las secciones anteriormente especificadas.

4 Apéndices

Se incluirá aquí cualquier tipo de información relacionada con la ERS, pero que no forme parte de la misma. Por ejemplo, se incluirían los resultados del análisis de costes, restricciones especiales acerca del lenguaje de programación...

5 Índice

Se proporciona un índice para poder tener un acceso rápido a la documentación presentada en la ERS.

5 Conclusiones

Para conseguir el éxito en cualquier desarrollo de software es esencial la comprensión total de los requisitos del usuario. No importa lo bien diseñado o codificado que pueda estar, si no se ha analizado correctamente puede defraudar al usuario y frustrar al desarrollador.

El análisis y la especificación de los requisitos puede parecer una tarea relativamente sencilla, pero, en realidad, el contenido del análisis es muy denso y abundan las malas interpretaciones o la falta de información. Es muy difícil evitar la ambigüedad.

El análisis de requisitos es la fase más importante en el desarrollo de un proyecto software, ya que es en esta fase en la que el usuario indica las especificaciones del futuro sistema, porque de un correcto análisis dependerá la correcta implementación de la aplicación.

El documento de especificación de requisitos software supone una especie de contrato entre cliente y desarrolladores en el que unos indican sus necesidades, mientras que los otros se limitan a implementar lo que se indica en el documento. Principalmente por esta razón tiene tanta importancia la fase de análisis de requisitos.

La tarea del análisis de requisitos es un proceso de descubrimiento, refinamiento, modelado y especificación y, por tanto, el desarrollador y el cliente tienen un papel activo en la obtención de estas necesidades.

La mejor manera de acercar ambas partes es hacer que el cliente forme parte activa del análisis de requisitos permitiendo que pueda interpretarlo y revisarlo. Las últimas tecnologías utilizadas para la obtención de requisitos permiten una mejor comprensión de los documentos de especificaciones, que hasta ahora eran demasiado técnicos para la correcta comprensión por parte del usuario.

Estas técnicas modernas son los casos de uso, que forman parte del UML. Ésta es la principal herramienta utilizada para el diseño completo de proyectos software orientado a objetos. Los casos de uso modelan el sistema desde el punto de vista del usuario, permitiéndole así la comprensión completa del futuro sistema. El nuevo producto se muestra en forma de "historieta".

El hecho de enfocar el análisis de requisitos hacia el usuario tiene una doble ventaja: por un lado evita las tendencias del informático hacia un diseño técnico que permita optimizaciones innecesarias o complicaciones añadidas; por otro lado, la participación del usuario en el proceso y la utilización de su lenguaje cotidiano en la redacción de los casos de uso facilita la identificación de las necesidades del sistema.

Finalmente, se debe indicar que esta fase es posiblemente la más costosa (temporalmente) en el desarrollo de un producto software. Esto se debe a que, en general, el cliente no sabe realmente lo que quiere y requiere la ayuda de los analistas para concretar las funciones que realmente se han de implementar. Por tanto, de la calidad del documento de ERS dependerá el desarrollo y calidad del producto final. La existencia de un estándar, como es el presentado en este trabajo, para la ERS (IEEE 830) permite la coherencia en la especificación de requisitos y ayuda a no dejar cabos sueltos.

6 Bibliografía

[Chalmeta, 1999]: Chalmeta R. "ADSI II. 2º Boletín de transparencias". UJI, 1999.

[Davis A., 1995]: Davis A. 201 Principles of Software Development. 1ª ed. McGraw-Hill, 1995.

[Gause, 1989]: Gause D.C. y G.M. Weinberg. Exploring Requirements: Quality before design. Dorset House, 1989.

[Piattini, 1996]: Piattini Mario G. Análisis y diseño detallado de aplicaciones informáticas de gestión. 1ª ed. RA-MA Editorial, Madrid, 1996.

[Pierre-Alain, 1997]: Pierre-Alain M. Modelado de objetos con UML. 1ª ed. Ediciones Gestión 2000 S.A, Barcelona, 1997.

[cern, 2000]: Web con un amplio glosario de términos de Ingeniería del software. http://dxsting.cern.ch/sting/glossary.html

[IEEE, 1998]: IEEE Recommended practice for software requirements specification. Artículo obtenido de la web del instituto de ingenieros eléctricos y electrónicos (Institute of Electrical and Electronics Engineers). http://www.computer.org/

[thehath, 2000]: Sitio web con abundante información sobre análisis. http://www.thehathway.com/

[upm, 2000]: Web de la universidad politécnica de Madrid, aparecen algunos artículos sobre la IEEE 830 y sobre análisis de requisitos en general. http://www.upm.com.

Anexo A : Glosario de Términos Técnicos

Análisis de requisitos: fase de un proyecto software donde se efectúa un conjunto de actividades con el propósito de comprender el problema planteado con todo detalle y se enuncia el resultado de dicho proceso de comprensión en forma de un planteamiento técnico del problema que se denomina especificación técnica.

Caso de uso: herramienta que modela los servicios que ofrece el sistema a través de un diálogo entre un actor y el sistema. Acciones del usuario y reacciones del sistema. *"Un caso de uso es una secuencia de transacciones proporcionadas por el sistema que proporcionan un resultado mensurable de valores a un actor particular"*

DFD: Diagrama de Flujo de Datos. Es un diagrama en forma de red que representa el flujo de datos y las transformaciones que se aplican sobre ellos al moverse desde la entrada hasta la salida del sistema. Se utiliza para modelizar las funciones del sistema y los datos que fluyen entre ellas a distintos niveles de abstracción. El sistema, por tanto, se modela mediante un conjunto de DFS nivelados en el que los niveles superiores definen las funciones del sistema de forma general u los niveles inferiores definen estas funciones en niveles más detallados.

Especificación de requisitos: proceso de redacción o registro de los requisitos. Se pueden utilizar tanto el lenguaje natural, como modelos gráficos.

Especificación: es un documento que define, de forma completa, precisa y verificable, los requisitos, el diseño, el comportamiento u otras características de un sistema o componente de un sistema [Piattini, 96].

Ingeniería del Software : es el establecimiento y uso de principios de ingeniería robustos, orientados a obtener software económico que sea fiable y funcione de manera eficiente sobre máquinas reales, mediante la aplicación de los siguientes elementos y actividades: los métodos, la planificación y estimación de proyectos, el análisis de los requisitos del sistema y del software, el diseño de estructuras de datos, la arquitectura de programas y procedimientos algorítmicos, la codificación, las pruebas, la instalación y el mantenimiento, las herramientas y los procedimientos.

Proyecto software : conjunto de actividades coordinadas cronológicamente para alcanzar un subconjunto de objetivos a partir de la definición de un subconjunto de necesidades.

Requisitos: Descripción de las necesidades o deseos de un producto.

Software : es el conjunto de programas, procedimientos y documentación asociada a la operación de un sistema informático [Piattini, 96].

UML: Unified Modeling Language. Lenguaje de programación gráfico para el modelado de proyectos software orientados a objetos.

Validación de los requisitos: Proceso de confirmación, por parte de los usuarios o del cliente, de que los requisitos especificados son válidos, consistentes, completos, etc.

Verificación de los requisitos: Proceso de comprobación de que los requisitos realmente cubren las necesidades del cliente.

Anexo B: Otras versiones del IEEE 830