UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

TIA
Tecnología, Investigación y Academia

Impacto de los requerimientos en la calidad de *software*

Impact of Requirements on Software Quality

Cindy Tatiana Rodríguez Barajas[1]

Para citar este artículo: Rodríguez, C. T. (2017). Impacto de los requerimientos en la calidad de *software*. *TIA*, 5(2), pp. 161-173.

Resumen

El siguiente artículo pretende dar a conocer el impacto que tiene la definición de los requerimientos en la calidad del desarrollo de *software*; se darán a conocer conceptos básicos sobre requerimientos y la correcta forma de realizar su definición (levantamiento), calidad de *software* enfocada a pruebas estáticas y un breve análisis estadístico, de cómo los requerimientos influyen en la calidad de *software* y, por ende, en los costos del proceso de desarrollo de *software*.

Palabras clave: calidad de *software*, defectos, elicitación, pruebas estáticas, requerimientos.

Abstract

This article intends to make known the impact that has definition of requirements in the quality of software development. Basic concepts of requirements and the right way to define them (survey), quality of software focused on static tests and a brief statistical analysis of how requirements influence the quality of software and, therefore, the costs of software development process will be announced.

Keywords: software quality, defects, elicitation, static test, requirements..

Artículo de investigación

Fecha de recepción:
14-10-2014
Fecha de aceptación:
27-06-2017

ISSN: 2344-8288
Vol. 5 No. 2
Julio - diciembre 2017
Bogotá-Colombia

[1] Ingeniera Industrial, Universidad Distrital Francisco José de Caldas. Correo electrónico: cindy_taty@hotmail.com

INTRODUCCIÓN

Hoy en día el proceso de desarrollo de *software* es de vital importancia para muchas empresas, pues a través de él se da solución a las necesidades de los usuarios. Una de las etapas más importantes dentro de dicho proceso consiste en la definición de requerimientos, ya que es en esta etapa donde se plasman de forma clara y concisa todos los requisitos del usuario.

Sin embargo, muchos proyectos de desarrollo de *software* no tienen el éxito esperado debido a la deficiente información que posee el documento de requerimientos; es por esta razón que a través del presente documento se desea presentar información clave para entender qué es un requerimiento y las diferentes técnicas que existen para realizar una adecuada definición de los mismos.

Asimismo, se desea mostrar una serie de estadísticas que evidencian cómo los requerimientos afectan de forma significativa la calidad del *software* y, por ende, los costos asociados al proceso de desarrollo de *software*.

CONTENIDO

Datos relevantes sobre requerimientos

Definición de requerimiento

Un requerimiento se puede definir como una necesidad documentada que establece la forma en la cual debe funcionar un sistema; debe incluir, de forma clara y concisa, los atributos y características que del sistema para cumplir con las expectativas del usuario.

De acuerdo con el estándar IEEE 830 [1], un requerimiento se define como "Condición o capacidad que debe poseer un sistema o componente de un sistema para satisfacer un contrato. Un estándar, una especificación u otro documento formalmente impuesto". Sin embargo, los requerimientos se ven influenciados por factores como las perspectivas y prejuicios de los usuarios respecto al sistema o las políticas organizacionales, lo cual puede llegar a incidir de forma negativa en las etapas posteriores del desarrollo de *software* si no se realiza una adecuada definición de los mismos [2].

Es por esta razón que un apropiado conocimiento acerca de las características de los requerimientos y de las técnicas que permiten plasmar las verdaderas necesidades de los usuarios, ofrece ventajas tales como la reducción de esfuerzo en la etapa de desarrollo, acertada estimación de costos y fácil identificación de posibles mejoras.

Características de los requerimientos

De acuerdo con el estándar IEEE 830, un requerimiento debe cumplir con ciertas características, para ser considerado de calidad. En la Figura 1 se muestran de manera general las características fundamentales.

A continuación, se desarrolla lo expuesto en la Figura 1.

- Correcto: un requerimiento cumple esta característica únicamente si las definiciones que se encuentran en el documento corresponden a necesidades reales de los usuarios.
- No ambiguo: se refiere a que las necesidades plasmadas en el documento de requerimientos poseen una única interpretación.
- Completo: se considera que un requerimiento se encuentra completo si incluye todas las definiciones de las funcionalidades deseadas, interfaces (externas y graficas), entradas, salidas, flujos alternos y terminología.
- Verificable: cuando a través de un proceso una persona o herramienta puede constatar que el sistema satisface lo establecido en el requerimiento, por ejemplo: "la salida se suministra dentro de los veinte segundo siguientes al evento E el 60% de las veces" [3].

- Consistente: indica que un requerimiento es consistente si y solo si, el conjunto de requisitos allí definidos no se contradice entre ellos o generan conflicto.
- Priorizable: un requerimiento debe ser priorizable, es decir, debe permitir determinar su importancia con relación a las necesidades de los usuarios.
- Modificable: si al realizar un cambio sobre las definiciones ya establecidas de forma fácil, completa y consistente, se considera que el requerimiento cumple con esta característica.
- Explorable: también conocida como "Trazabilidad", se cumple esta característica cuando es posible rastrear de forma sencilla el origen de los requerimientos y los componentes del sistema que ejecutarán las funcionalidades descritas.
- Utilizable: cuando es posible utilizar el documento de requerimientos para las tareas de mantenimiento.

Tipos de requerimientos

Los requerimientos pueden ser clasificados en dos tipos, los requerimientos funcionales y los no funcionales [4].

Los requerimientos funcionales hacen referencia a las actividades y servicios que debe ejecutar el sistema, este tipo de requerimientos tiene en cuenta las entradas, salidas y el procesamiento y almacenamiento de los datos; también son conocidos como requerimientos de comportamiento. Por lo general, están descritos en términos de usuarios y tradicionalmente se detallan a través de la sentencia "debería" [5].

Por ejemplo, para un sistema de biblioteca universitario, usado por estudiantes y personal docente, se podrían definir los siguientes requerimientos funcionales:

- El usuario deberá tener la posibilidad de buscar en el conjunto inicial de la base de datos o seleccionar un subconjunto de ella.
- El sistema deberá proporcionar visores adecuados para que el usuario lea documentos en el almacén de documentos.
- A cada pedido se le deberá asignar un identificador único que el usuario podrá copiar al área de almacenamiento permanente de la cuenta.

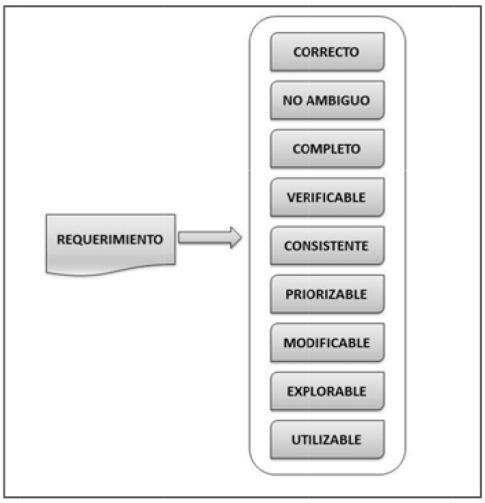

Figura 1. Características de los requerimientos según IEEE

Fuente: elaboración propia.

Los ejemplos son tomados textualmente de [2].

Por otro lado, los requerimientos no funcionales se relacionan con a las propiedades que debe poseer el sistema a fin de soportar las funcionalidades definidas por el usuario; en este tipo de requerimientos se describen las capacidades técnicas que el sistema debe poseer, tales como fiabilidad, tiempos de respuesta, disponibilidad, rendimiento, etc.

De acuerdo con Sommerville, este tipo de requerimientos en ocasiones son más críticos que los requerimientos funcionales, ya que la omisión de alguno de estos requisitos puede hacer que el sistema sea inútil para el usuario.

En la Figura 2 se muestra la tipología de los requerimientos no funcionales.

Estructura básica de un requerimiento

Una posible estructura de documento de requerimientos, de acuerdo con el estándar IEEE 830, debe contener como mínimo los elementos que se describen en la Figura 3.

A continuación se desarrolla lo expuesto en la Figura 3.

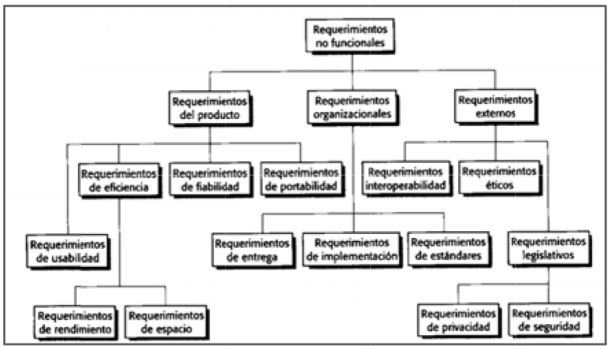

Figura 2. Tipo de requerimientos no funcionales según Summerville

Fuente: [2].

1. INTRODUCCION
1.1. Propósito
1.2. Alcance
1.3. Definiciones, siglas y abreviaciones
1.4. Referencias
1.5 Apreciación Global

2. DESCRIPCION GLOBAL
2.1. Perspectiva del producto
2.2. Funciones del producto
2.3. Características del usuario
2.4. Restricciones
2.5. Atención y dependencias

3. REQUERIMIENTOS ESPECIFICOS
3.1. Interfaces Externa
3.2. Funcionalidades
3.3. Requisitos de Desarrollo
3.4. Base de Datos Lógica
3.5. Restricciones de Diseño
3.6. Atributos de Calidad del Software
3.7. Modelo orientado a objetos

4. APENDICE
5. INDICE

Figura 3. Estructura de Requerimiento según IEEE 830

Fuente: elaboración propia.

- Propósito: describe el objetivo general de los que se desea realizar y los usuarios hacia los cuales va dirigido.
- Alcance: describe lo que hará el sistema y lo que no hará; además, describe los beneficios que traerá la implementación del sistema.
- Definiciones: establece el vocabulario que se utilizará a lo largo del requerimiento.
- Referencias: listado de documentos adicionales que soporten el requerimiento tales como casos de uso, concepto de expertos sobre los problemas actuales, etc.
- Apreciación global: describe de forma muy general el contenido del requerimiento y cómo se encuentra organizado.
- Perspectiva del producto: en esta sección se presenta el caso de negocio y el concepto operacional del sistema; describe cómo el sistema dará solución a una problemática del negocio y lo usuarios a quien servirá.
- Funciones del producto: en esta sección se deben resumir las principales capacidades del sistema, es aquí donde se incluyen los casos de uso.
- Características del usuario: describe y justifica las habilidades técnicas y capacidades de cada clase de usuario.
- Restricciones: se incluyen todas las limitaciones del sistema tales como políticas reguladoras, interfaces con otras aplicaciones, funciones de control, seguridad del sistema, etc.
- Atención y dependencias: en esta sección se debe nombrar cada uno de los factores que tienen incidencia sobre el requerimiento. En caso de que el requerimiento dependa de uno más grande, se deben relacionar los requisitos de mayor jerarquía y las interfaces entre los sistemas.
- Interfaces externas: detallas todas las salidas y entradas del sistema.
- Funcionalidades: incluye una especificación detallada de cada caso de uso donde se definen las acciones fundamentales.
- Requisitos de desarrollo: se describen requisitos estáticos y dinámicos del sistema como el número de terminales que debe soportar el sistema o número de usuarios que tendrá el sistema.
- Base de datos lógica: describe el tipo de información que utilizará el sistema, frecuencia de uso de los datos, entidad de los datos y sus relaciones, restricciones de integridad de la información, etc.
- Restricciones de diseño: especifica las restricciones que se puedan presentar por imposición de estándares, limitaciones del *hardware*, etc.
- Atributos de calidad de *software*: definición clara de cómo se da cumplimiento a las características de un requerimiento.
- Modelo orientado a objetos: describe la arquitectura del sistema, el diagrama de clases y los diagramas de actividad.

Técnicas para especificación de requerimientos

Una adecuada especificación de requerimientos permitirá generar información consistente, clara y compacta. Hoy en día existen múltiples técnicas para el levantamiento de requerimientos, esta actividad se conoce como elicitación.

Algunos de los autores más importantes que proponen varias de las técnicas comúnmente utilizadas actualmente son Loucopoulos [6], y Nuseibeh y Easterbrook [7]. De acuerdo con Loucopulos [6], algunas de las técnicas de elicitación de requerimientos que pueden ser utilizadas para el levantamiento de requerimientos se describen en la siguiente Tabla 1.

De acuerdo con Nuseibeh y Easterbrook [7], las técnicas de elicitación se pueden resumir en lo expuesto en la Tabla 2.

Calidad de *software*

La calidad de *software* se refiere al conjunto de cualidades que caracterizan un producto de *software* y que determinan su utilidad y existencia; la calidad es sinónimo de eficiencia, flexibilidad, mantenibilidad, portabilidad, usabilidad, seguridad e integridad [8] y [9].

Uno de los elementos fundamentales para determinar la calidad de *software* es el *testing* o pruebas. Las pruebas permiten validar y verificar el *software*, entendiendo la validación como el proceso que permite determinar si el *software* cumple los requisitos y la verificación como el proceso para determinar si el producto de una determinada fase satisface las condiciones definidas para continuar con la siguiente [10].

Tabla 1. Técnicas de elicitación de acuerdo con Loucopulos

Originadas por el Usuario	Esta técnica obtiene la información directamente del usuario a través de entrevistas o de la lluvia de ideas.
Escenarios	Se obtiene una descripción parcial del comportamiento del sistema
Análisis de Formularios	Corresponde a una recopilación estructurada de las variables de entrada, a fin de identificar como ingresaran los datos y su recuperación.
Lenguaje Natural	Forma de presentación del conocimiento se presenta con un vocabulario informal y predefinido, lo que resulta bastante familiar para el usuario.
Reuso de Requerimientos	Establece que requerimientos capturados con anterioridad pueden usarse nuevamente para especificar una aplicación similar.
Análisis de Tareas	Se recopilan todas las interacciones que pueden existir entre hombre - máquina para el nuevo sistema.

Fuente: elaboración propia a partir de [6].

Tabla 2. Técnicas de elicitación de acuerdo con Nuseibeh y Easterbrook

Técnicas Tradicionales	Abarcan una amplia clase de métodos de recolección de datos genéricos, incluyendo el uso de cuestionarios, entrevistas y el análisis de documentación existente.
Técnicas de Elicitación Grupales	Se busca una mejor comprensión de las necesidades de los usuarios, utilizando lluvia de ideas o RAD/JAD
Prototipos	Es un modelo final del producto de software, que permite evaluar determinados atributos del mismo sin necesidad que esté disponible.
Técnicas Orientadas por Modelos	Proveen un modelo específico del tipo de información a capturar. Abarcan métodos basados en escenarios y objetivos.
Técnicas Cognitivas	Técnicas desarrolladas para la adquisición del conocimiento destinado a sistemas basados en conocimiento. Abarcan: Análisis de Protocolo, Laddering, Distribución de tarjetas o Grillas de Repertorio

Fuente: elaboración propia a partir de [7].

Uno de los pilares fundamentales en la etapa de pruebas, es hallar todos los posibles defectos del *software* antes de continuar a una etapa posterior dentro del proceso de desarrollo de *software*.

Un estudio realizado por James Martin [11], reveló que la causa del 56% de los defectos encontrados en el proceso de pruebas, se debe a errores introducidos en la fase de requisitos como resultado de requerimientos mal escritos, ambiguos o incompletos (Figura 4).

El esfuerzo que implica corregir este tipo de defectos es mucho mayor que cualquier otro (82%), como se ve en la Figura 5.

Sin embargo, esta situación se presenta en gran medida porque el esfuerzo de las pruebas se encuentra concentrado en una fase tradicional de pruebas, en la cual los defectos se encuentran hasta el final del proyecto (Figura 6).

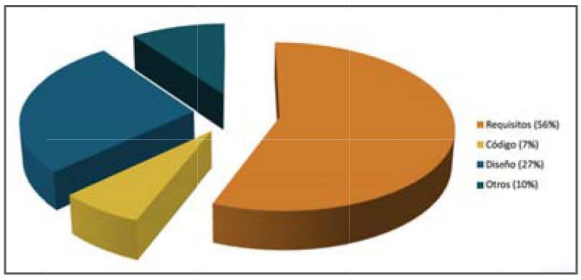

Figura 4. Distribución típica del origen de los defectos
Fuente: elaboración propia [12].

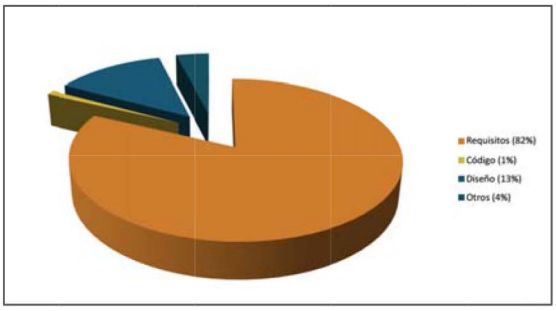

Figura 5. Distribución típica del esfuerzo para resolver errores
Fuente: elaboración propia [12].

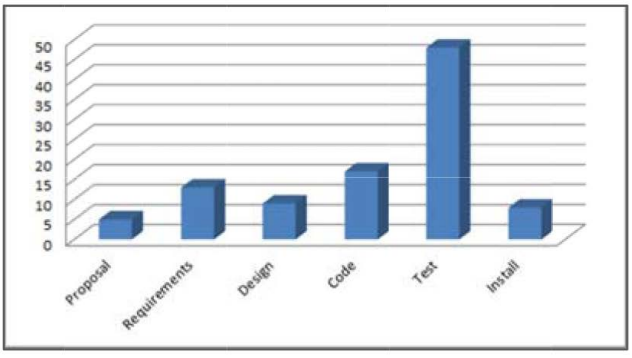

Figura 6. Uso de los recursos de pruebas en las etapas de desarrollo de *software*
Fuente: elaboración propia.

Momento en el cual el costo de reparar los defectos es alto y riesgoso, ya que es posible encontrarse con errores complejos que requieran un tiempo considerable para su solución, lo cual afectaría el cronograma y presupuesto del proyecto o errores que no son factibles de solucionar, incumpliendo así los requerimientos del usuario.

De acuerdo con los datos promedio de la industria, el costo de reparar un error de requerimiento es aproximadamente de 139 US, si se encontrara durante la fase de requerimientos; sin embargo, si ese error se encontrara en fases posteriores el costo de arreglarlo aumentaría significativamente (Figura 7).

Por ejemplo, si se encontrará un error por requerimiento en la fase de pruebas, el costo de reparación de este defecto correspondería a los costos asociados de las fases anteriores a la detección hasta la fase donde se introdujo el error (Figura 8).

Es por esta razón que se debe garantizar la calidad del software desde el inicio del proceso, realizando una adecuada definición de los requerimientos y validando desde el inicio que estos cumplan con las características de un buen requerimiento; para esto, se recomienda incluir de forma temprana al equipo de pruebas a fin de que se realicen pruebas estáticas que permitan identificar los defectos de forma temprana.

Las técnicas de revisión estática se conocen como revisiones porque detectan manualmente defectos en cualquier producto del desarrollo; dentro de las técnicas de revisiones se encuentran las revisiones informales, las inspecciones, walktrough y auditorías [13].

- Revisiones informales: consiste en un intercambio de opiniones entre los participantes.
- Inspecciones: son un método de análisis para verificar y validar un producto de software manualmente. Son un proceso definido donde un equipo de personas analiza el producto través de una técnica de lectura, a fin de detectar defectos antes que inicie la fase de pruebas funcionales.
- Walktrough: consiste en simular la ejecución de casos de prueba, es decir, se recorre le programa imitando lo que haría la computadora.

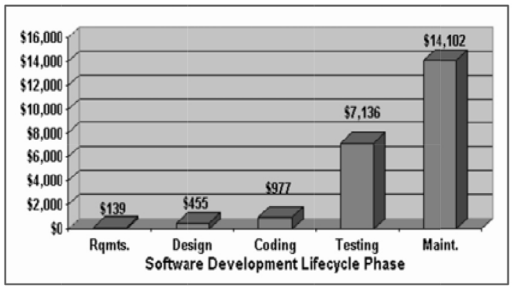

Figura 7. Costo de la corrección de defectos en el ciclo de vida de desarrollo de *software*
Fuente: elaboración propia.

Costo error de RQ encontrado en pruebas	= Costo Pruebas + Costo Desarrollo + Costo Diseño
	= 7136 + 977 + 455
	= 8568

Figura 8. Costo de corrección de un defecto de requerimiento encontrado en pruebas
Fuente: elaboración propia.

- Auditorías: contrastan los artefactos generados durante el desarrollo con estándares generales o de la organización.

Para la evaluación de requerimientos, es posible utilizar la técnica de lectura basada en listas de comprobación, ya que proporcionan un apoyo mediante preguntas que se deben responder a medida que se lee el artefacto.

Una lista de comprobación para requisitos contiene preguntas sobre los defectos típicos que suelen aparecer en los requisitos (Figura 9).

Dentro de este check lists, las primeras preguntas corresponden a los criterios de calidad de los requisitos, mientras que las últimas establecen los problemas típicos cuando se especifican los requisitos.

La fase de definición de requerimientos determina la calidad del software desde la perspectiva del usuario, quien es el que finalmente hará uso del sistema; es por esta razón que los requisitos necesitan ser evaluados de forma crítica para prevenir y disminuir los costos asociados a este tipo de errores.

CONCLUSIONES

A fin de evidenciar en un entorno real el impacto de los requerimientos dentro del proceso de *software*, se aplicaron pruebas estáticas a una muestra de requerimientos diferentes dentro de una entidad financiera; esta evaluación se realizó a través de un formulario en el cual, por medio de preguntas, se identificó si el requerimiento cumplía o no cumplía las características fundamentales de un requerimiento.

En la Tabla 3, la Tabla 4 y la Tabla 5 se encuentra el formulario de evaluación utilizado y los criterios de evaluación definidos.

A continuación, se presentan los resultados más significativos de la aplicación de este formulario.

De la muestra de requerimientos utilizada para la evaluación, se pudo identificar que ningún documento cumple completamente con las características de calidad, solo el 30% de dichos requerimientos cumple con algunas características de un buen requerimiento (Figura 10).

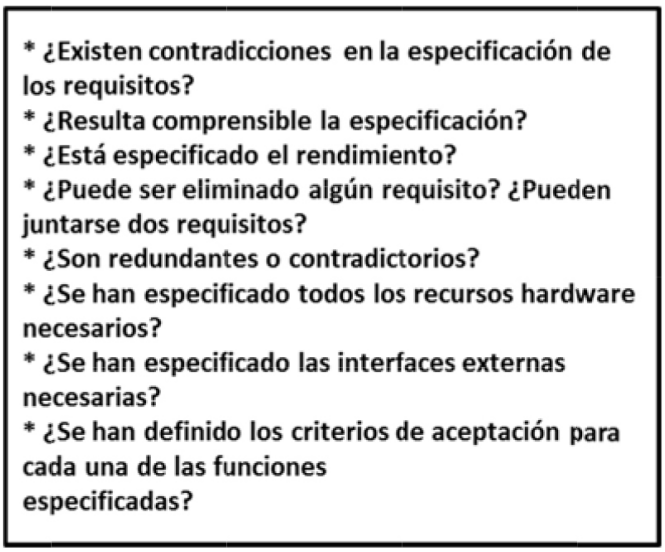

Figura 9. Ejemplo *check list* para verificación de requerimientos
Fuente: elaboración propia.

Tabla 3. Formulario validación requisitos

Aspecto a Evaluar	Pregunta de Validacion
Necesario	1. ¿Es necesario el requerimiento para el objetivo de la aplicación?
	2. ¿Es clara la importancia del requerimiento dentro de la organización?
	3. ¿La omision del requerimiento provocaría una deficiencia en el sistema?
Viable	4. ¿Estan los requerimientos dentro del ambito del proyecto?
	5. ¿Hay identificación de riesgos en los requerimientos?
	6. ¿Hoy consistencia con las politicas y organización del cliente?
Consistente	7. ¿El documento esta bien organizado?
	8. ¿El esquema general es consistente con los requerimientos de alto nivel?
	9. ¿El requerimiento NO entra en conflicto con otro requerimiento ni se encuentra duplicado?
	10. ¿Estan las referencias o relaciones con otros requerimientos bien definidas?
Correcto	11. ¿Estan los acronimos bien definidos?
	12. ¿El documento se encuentra libre de errores ortograficos y gramaticales?
	13. ¿En el requerimiento se identifican casos de prueba o vivencias reales?
Completo	14. ¿Se encuentran identificadas las funcionalidades que involucra el requerimiento?
	15. ¿Estan definidas las interfaces graficas requeridas?
	16. ¿Estan definidas las entradas y salidas?
	17. ¿Estan definidos los limites operativos de las funcionalidades?
	18. ¿Estan definidas todas las funcionalidades que se necesitan?
No Ambiguo	19. ¿El requerimiento se interpreta igual al ser leido por más de una persona?
	20. ¿Esta la funcionalidad claramente definida?
	21. ¿La redaccion del requerimiento es simple y clara?
Verificable	22. ¿Existen soportes que ayuden a entender el requerimiento?
	23. ¿Se encuentra definido un metodo de verificacion para cada funcionalidad?
	24. ¿Se puede hacer una trazabilidad del requerimiento hasta su origen?
Priorizable	25. ¿Se encuentra incluida la prioridad de cada requerimiento?
	26. ¿Se establecio un orden jerarquico en la definicion de los requerimientos?
	27. ¿Los requerimientos que están relacionados unos con otros se encuentran identificados con un nivel de impacto o importancia?

Fuente: elaboración propia.

Tabla 4. Aspectos a evaluar dentro de la prueba de requerimientos

ASPECTOS A EVALUAR	TOTAL ITEMS A EVALUAR
Necesario	3
Viable	3
Consistente	4
Correcto	3
Completo	5
No Ambigüo	3
Verificable	3
Priorizable	3

Fuente: elaboración propia.

Tabla 5. Criterios de Evaluación para la prueba de Requerimientos

CRITERIO DE EVALUACION			
Aprobado	Si los items a evaluar tienen nivel de cumplimiento entre	80%	100%
Ajustar	Si los items a evaluar tienen nivel de cumplimiento entre	40%	79%
No aprobado	Si los items a evaluar tienen nivel de cumplimiento entre	0%	39%

Fuente: elaboración propia.

La Figura 11 evidencia que la mayoría de los requerimientos evaluados no cumplen las características de viabilidad, correcto, completo y no ambiguo; sin embargo, se evidencia que el aspecto en el cual los requerimientos tienen un cumplimiento óptimo es la consistencia.

De acuerdo a la evaluación realizada, se identificó que los ítems donde más inconvenientes se presentan a en el momento de la definición de un requerimiento de calidad son (Figura 12):

- Identificación de riesgos.
- Consistencia con políticas de la organización.
- Definición de acrónimos.
- Ortografía del requerimiento.
- Identificación de casos de prueba o vivencia reales.
- Identificación clara de las funcionalidades.
- Definición de interfaces gráficas.
- Definición entradas y salidas.
- Límites operativos.
- Interpretación del requerimiento por todos los involucrados.
- Redacción del requerimiento.

Figura 10. Estado de requerimientos evaluados

Fuente: elaboración propia.

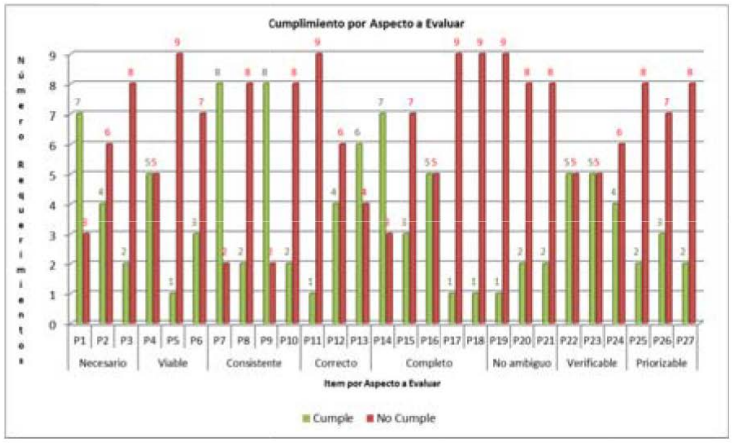

Figura 11. Cumplimiento de los requerimientos por aspecto a evaluar

Fuente: elaboración propia.

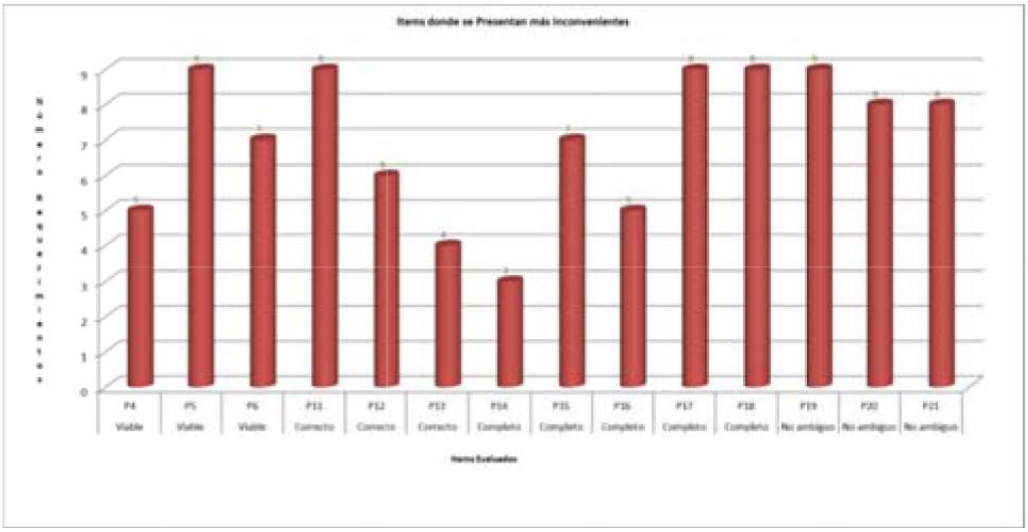

Figura 12. Ítems donde se encuentran los principales problemas
Fuente: elaboración propia.

Además, se verificó para los proyectos de desarrollo de *software* que se encuentran en pruebas, de acuerdo al reporte de errores, el porcentaje que corresponde a defectos por requerimientos (Figura 13).

Para los proyectos en proceso de pruebas, se encontró que del total de defectos reportados el 23% de los errores se dan por inadecuada definición de requerimientos.

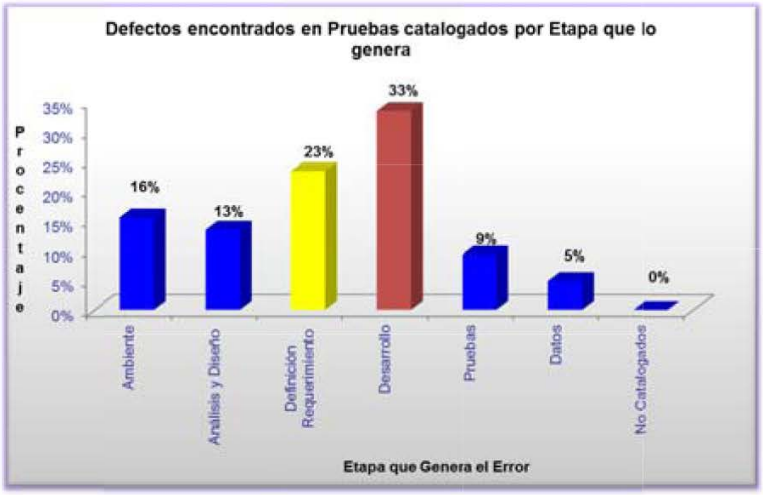

Figura 13. Defectos encontrados en pruebas catalogados por etapa que genera el error
Fuente: elaboración propia.

REFERENCIAS

[1] IEEE. (1998). *Especificación de Requerimientos de Software, IEEE/ANSI 830–1998*. Autor.

[2] Sommerville. (2005). *Ingeniería del Software*. [Séptima edición]. Madrid: Pearsson Education S. A.

[3] Monferrer A. (2000). *E78 Ingeniería del Software, Curso de ingeniería informática 2000–2001*. Castellón de la Plana: Departament d' Informàtica, Universitat Jaume I.

[4] Fernández, S. n. (2006). Desarrollo de sistemas de información: Una metodología basada en el modelado. *E-aula politécnica, 120*, 82-86.

[5] Castillo, J. (2007). *Seminario MIS–CIMAT: Perfil del ingeniero de requerimientos*. México: Centro de Investigación en Matemáticas A.C.

[6] Loucopoulos P. y Karakostas, V. (1995). *System Requirements Engineering*. Michigan: McGraw Hill International Software Quality Assurance Series.

[7] Nuseibeh B. y Easterbrook S. (2000). *Requirements Engineering: A Roadmap, ICSE 2000*. Irlanda: Limerick.

[8] ISO. (2014). *ISO/IEC 9126, Software Engineering–Product*. Autor.

[9] ISO (2011). *ISO/IEC 25010, System and Software Engineering–System and Software Quality Requeriments and Evaluation*. Autor.

[10] Pressman, S. y Roger, S. (2005). *Ingeniería de Software: Un enfoque práctico*. México: McGraw Hill Computer Science Series.

[11] Mogyorodi, G. (2003) What is Requirements Based Testing? *The Journal of Defense Software Engineering,* 12-15.

[12] Lazic L., y Mastorakis N. (2008). Cost Effective Software Test Metrics. *WSEAS Transactions on Computers, 6*(7), 599-616.

[13] Dolado J., Ramos, I. y Tuya, J. (2007). *Técnicas cuantitativas para la gestión en la ingeniería del software*. Netbiblo.