**Universidad Técnica Estatal de Quevedo**

****

**Facultad de Ciencias de la Computación y Diseño Digital**

**Asignatura:**

Ingeniería de Requerimientos

**Curso:**

4to Software “B”

**Docente:**

Ing. Guerrero Ulloa Gleiston Ciceron

**Alumno:**

Manotoa Alava Carlos Alexander

**Link de GITHUB:**

#### <https://github.com/CarlosMC2004/Ingenieria-De-Requerimientos.git>

#### *Descripción*

***De manera autónoma identifica problemas cotidianos que tienen las personas que realizan actividades diversas con grupos de clientes, estudiantes, etc.… prepara una descripción de estas actividades y elabora una lista de requerimientos.***

Realizar actividades grupales comúnmente suele presentar conflictos, entre los más comunes destacan:

* **Conflictos interpersonales:** Se presentan por falta de comunicación efectiva, diferencias culturales, diferencias generacionales o competitividad excesiva.
* **Mala coordinación en los horarios:** Uno de los errores más comunes es realizar reuniones o actividades en horarios donde no todos tienen disponibilidad, este conflicto se presenta generalmente en grupos de trabajo grandes.
* **Conflictos con temas logísticos:** Ocurre cuando la coordinación y responsabilidad no funciona de manera correcta, los grupos al tener limitación de recursos se ven obligados a detener su producción.
* **Conflictos de liderazgo:** Tener personas con falta de liderazgo puede causar una mala comunicación, organización, cuestionamiento de decisiones, imparcialidad o imposiciones arbitrarias.
* **Sobrecarga o plazos te entrega reducidos:** La sobrecarga de trabajo o plazos de entrega muy cortos, pueden generar tensiones, resultados de baja calidad, descoordinación, etc.

Para solucionar estos conflictos se puede aplicar un conjunto de normas y metodologías que combinadas con herramientas tecnológicas ayudan a prever y corregir esto conflicto.

* **Herramientas de organización:** Para manejar la organización se pueden emplear herramientas como calendarios compartidos para la fechas y horas, o un sistema de registros para evitar sobrecargas o confusiones[1] .
* **Comunicación eficiente:** Usar plataformas de comunicación como WhatsApp, correos electrónicos o discord son de gran ayuda siempre y cuando se utilicen de manera correcta evitando la saturación de mensajes, ruido o spam [2].
* **Control de avances:** El emplear herramientas como Google Forms, Excel o encuestas ayudan a resolver dudas o conflictos de manera rápida, ya sea de forma grupal o individual [1].
* **Correcta distribución de recursos:** El asegurar que todos los grupos y subgrupos tengan los recursos necesarios a tiempo asegura que no se detenga el flujo de trabajo y evita que se genera conflictos [3].
* **Manejo de conflictos:** Es clave saber cómo actuar ante comportamientos negativos, el manejar canales privados de comunicación y el respeto ayudan a evitar que los conflictos escalen [4].

***Investiga al menos un marco de trabajo para la ingeniería de requerimientos (debe utilizar fuentes válidas). ¿Este marco es aplicable a proyectos de SW que usted realiza?***

En el desarrollo de software la ingeniería es una fase primordial que influye directamente en el resultado final de un proyecto. El objetivo primordial es capturar, analizar, documentar y validar las necesidades de todos los stakeholders[5].

En la actualidad la complejidad de los proyectos, los requerimientos dinámicos, plazos ajustados y equipos multidisciplinarios. Los marcos de trabajo son metodologías estructuradas que garantizan precisión, consistencia y trazabilidad [5].

Entre los marcos de trabajo más conocidos se encuentran:

* **IEEE 830-1998:**
* **Volere**
* **BABOX:**
* **MBSE:**

En este caso he decidido analizar el estándar “IEEE 830-1998”**,** este estándar indica las directrices para la elaboración de una Especificación de Requerimientos (SRS). Este marco de trabajo ayuda a la comunicación entre desarrolladores, clientes y usuarios finales. También reduce riesgos como ambigüedades, omisiones o contradicciones en los requisitos [6].

**Estándar IEEE 830-1998**

Este marco de trabajo fue publicado por el “Institute of Electrical and Electronics Engineers”, define buenas prácticas para redactar una especificación de requisitos de software (SRS) [6].

La estructura de este marco de trabajo incluye:

1. **Introducción**

En esta sección se da una introducción a todo el documento de

Especificación de Requisitos Software (ERS). Se compone de varias subsecciones [7]:

* Propósito
* Ámbito del sistema
* Definiciones
* Referencias y visión general del documento.
  1. **Propósito**

En esta subsección se define el propósito del documento ERS y se especifica hacia quién va dirigido el documento [7].

* 1. **Ámbito del Sistema**

En este apartado se puede dar un nombre al futuro sistema. También se explica lo que el sistema hará y lo que no hará. Adicionalmente se explica los beneficios, objetivos y metas esperadas [7].

Se referencia toda aquella documentación que incluye al Hardware y software, es importante mantener la consistencia con el documento de especificación de requisitos globales del sistema.

* 1. **Definiciones, Acrónimos y Abreviaturas**

Aquí se definen términos, acrónimos y abreviaturas que se ocuparan en la ERS [7].

* 1. **Referencias**

Aquí se coloca una lista con todos los documentos referenciados en la ERS [7].

* 1. **Visión General del Documento**

Esta subsección describe de manera breve los contenidos y la organización del

resto de la ERS [7].

1. **Descripción General**

En este punto se describen los factores que influyen y afectan al producto y sus requisitos. Es importante comprender que aquí no se describen los requisitos, sino que se da un contexto [7].

Normalmente esta sección se compone de las siguientes subsecciones:

* Perspectiva del producto.
* Funciones del producto
* Características de los usuarios
* Restricciones
* Suposiciones y dependencias
  1. **Perspectiva del Producto**

Aquí se relaciona el sistema con otros productos. También aquí se debe especificar si el sistema es totalmente independiente otro sistema o si es parte de un sistema mayor [7].

* 1. **Funciones del Producto**

En este apartado se muestra un resumen con las funcionalidades que contendrá el sistema. Estas funciones se deberán mostrar de manera organizada, se pueden utilizar gráficos siempre y cuando estos muestren las relaciones de las funciones y no el diseño del sistema [7].

* 1. **Características de los Usuarios**

Aquí se describe a cada uno de los usuarios del producto, incluyendo experiencia educacional y/o técnica [7].

* 1. **Restricciones**

En esta sección se describen las limitaciones que se imponen sobre los desarrolladores del sistema.

* Políticas institucionales.
* Limitaciones del hardware.
* Interfaces con otras aplicaciones.
* Operaciones paralelas.
* Funciones de auditorías.
* Funciones de control.
* Lenguaje(s) de programación.
* Protocolos de comunicación.
* Requisitos de habilidad.
* Criticalidad de la aplicación
* Consideraciones acerca de la seguridad
  1. **Suposiciones y Dependencias**

En esta sección se describe aspectos que si cambian afectan directamente a los requerimientos [7].

* 1. **Requisitos Futuros**

Aquí se abordará futuras mejoras del sistema que se podrían analizar e implementar a futuro [7].

1. **Requisitos Específicos**

Esta es la sección más larga e importante del documento, aquí se colocan los requerimientos a un nivel de detalle suficiente para permitir que los diseñadores puedan implementar una solución la cual se someterá posteriormente a pruebas que demuestren si el sistema satisface estos requerimientos [7].

Aquí todo requisito especifico describirá comportamientos externos del sistema, perceptible por parte de los usuarios, operadores y otros sistemas. Se debe considerar que el documento debe ser legible por personas de distintas profesiones [7].

Aunque en la practica no siempre se realiza, lo ideal es que estos requerimientos posean las siguientes características:

* **Corrección:** La ERS es correcta si y sólo si todo requisito que será implementado refleja alguna necesidad real. Esto demuestra que esta implementación cumple con el resultado deseado [7].
* **No ambiguos:** Cada requisito tendrá un única interpretación, se utilizará gráficos o notaciones formales. En caso de utilizar términos que contengan más de una interpretación, este será definido con precisión en el glosario [7].
* **Completos:** Conviene incluir todas las posibles respuestas a los datos de entrada del sistema, tanto validos como no válidos [7].
* **Consistencia:** Los requisitos no pueden ser contradictorios, en caso de serlo, no es implementable [7].
* **Clasificados:** Normalmente no todos los requisitos son igual de importantes. Los requisitos se clasifican por importancia o por estabilidad, esto nos ayuda a no emplear un exceso de recursos en la implementación [7].
* Verificable: Un requisito verificado es un requisito testeado para verificar si el sistema satisface este requerimiento. En general un sistema ambiguo no es verificable [7].
* **Modificables:** La ERS es modificable si y sólo si se encuentra estructurada de forma que los cambios a los requisitos puedan llevarse acabo de manera fácil, completa y consistente [7].
* **Trazables:** La ERS es trazable si se conoce el origen de cada requisito y se facilita la referencia de cada requisito a los componentes [7].
  1. **Interfaces Externas**

En esta sección se describen los requisitos que afectan directamente a la interfaz de usuario, interfaces con otros sistemas e interfaces de comunicación [7].

* 1. **Funciones**

En esta sección se especificaran todas las funciones que el software deberá llevar acabo. El estándar IEEE 830 en sus ultimas versiones, permite organizar las subsecciones de diversas maneras. Por ejemplo:

**Por tipos de usuario:** Para cada uno de los usuarios que existan se especificarán sus requerimientos funcionales de acuerdo a su relevancia [7].

**Por objetos:** Organiza los requerimientos funcionales detallando sus atributos y funcionalidades. Para cada objeto se describe su propósito, se listan sus prioridades y se especifican las operaciones que el sistema puede realizar [7].

**Por estímulo:** Se especifican las funcionalidades relacionadas con estímulos que el sistema puede recibir [7].

**Por jerarquía funcional:** Si ninguna de las alternativas anteriores resultan, las funcionalidades se especificarán como una jerarquía de funciones que comparten entrada, salida o datos [7].

* 1. **Requisitos de Rendimiento**

Se detallan requisitos relacionados con la carga que se espera que el sistema soporte. Por ejemplo, el número de transacciones o de terminales. También se especificarán los requisitos de datos, es decir, aquellos que afectan a la información que se guardará en la base de datos [7].

* 1. **Restricciones de Diseño**

Se describen todas aquellas decisiones que restringen el diseño libre de la aplicación [7]. Restricciones como estándares o normativas, hardware limitado, etc.

* 1. **Atributos del Sistema**

Aquí se detallarán los atributos de calidad como:

* Fiabilidad
* Mantenibilidad
* Portabilidad
* Seguridad (Tipo e implementación)
* Niveles de acceso
  1. **Otros Requisitos**

Aquí se agregan los requisitos que no encajan con otros apartados. Puede contener todo tipo de información, siempre y cuando, no forme parte de la ERS [7]. Por ejemplo:

1. Formatos de entrada/salida de datos, por pantalla o en listados.

2. Resultados de análisis de costes.

3. Restricciones acerca del lenguaje de programación.

**Referencias**

[1] T. Escovedo and R. N. Melo, “Applying a methodology for collaborative assessment in learning groups,” in *IEEE EDUCON 2010 Conference*, 2010, pp. 1205–1210. doi: 10.1109/EDUCON.2010.5492388.

[2] D.-G. Lee, P.-S. Kim, and S.-W. Lee, “Local group relationship analysis for group activity recognition,” in *2017 17th International Conference on Control, Automation and Systems (ICCAS)*, 2017, pp. 236–238. doi: 10.23919/ICCAS.2017.8204447.

[3] J. Burke, B. Canavan, C. Collins, and M. Pollock, “Work in progress - developing an online resource on effective small group work,” in *2009 39th IEEE Frontiers in Education Conference*, 2009, pp. 1–2. doi: 10.1109/FIE.2009.5350765.

[4] T. R. M. B. Silva, A. Loureiro, and L. B. Ruiz, “A conflict resolution methodology for collective ubiquitous context-aware applications,” in *2009 13th International Conference on Computer Supported Cooperative Work in Design*, 2009, pp. 426–431. doi: 10.1109/CSCWD.2009.4968096.

[5] N. A. Prasetyo and Y. Bandung, “A design of software requirement engineering framework based on knowledge management and Service-Oriented Architecture Decision (SOAD) modeling framework,” in *2015 International Conference on Information Technology Systems and Innovation (ICITSI)*, 2015, pp. 1–6. doi: 10.1109/ICITSI.2015.7437708.

[6] “IEEE Recommended Practice for Software Requirements Specifications,” *IEEE Std 830-1998*, pp. 1–40, 1998, doi: 10.1109/IEEESTD.1998.88286.

[7] IEEE, “Especificación de Requisitos según el estándar de IEEE 830,” *IEEE 830-1998*, 2008.