

Resumen de la materia Ingeniería del Software I


Agustín Curto, agucurto95@gmail.com
Francisco Nievas, frannievas@gmail.com

2017

Contents

| | | |
|----------|---|----------|
| 1 | Introducción | 2 |
| 2 | Requerimientos de Software | 4 |
| 2.1 | Análisis del problema o requerimientos. | 4 |
| 2.1.1 | Modelado de flujo de datos | 4 |
| 2.1.2 | Modelado orientado a objetos | 4 |
| 2.1.3 | Prototipado | 5 |
| 2.2 | Especificación de requerimientos: | 5 |
| 2.2.1 | Características de una SRS | 5 |
| 2.2.2 | Una SRS debe especificar requerimientos sobre | 5 |
| 2.2.3 | Casos de uso | 6 |
| 2.3 | Validación | 6 |
| 2.3.1 | Errores más comunes | 6 |
| 2.4 | Métricas | 6 |
| 2.4.1 | Punto función | 6 |
| 2.4.2 | Métrica de calidad | 7 |

Nota: Este resumen se corresponde con la materia dictada en el año 2017. Los autores no se responsabilizan de posibles cambios que pudiesen realizarse en los temas dictados en la misma, así como tampoco de errores involuntarios que pudiesen existir en dicho resumen.

Por favor, mejorá este documento en github 
<https://github.com/ResumenesFaMAF/resumenIngenieria1>

1 Introducción

Software: Colección de programas, procedimientos, y la documentación y datos asociados que determinan la operación de un sistema de computación.

Dominio del problema:

| | | |
|--------------------------|---------------|----------------------------|
| | Alumno | Industria |
| Error (bug) | Tolerable | No Tolerable |
| Interfaz | No importante | Muy importante |
| documentación | No existe | Existe: Usuario y proyecto |
| Confiabilidad y robustez | No importante | Fundamental |
| Inversión | No Existe | Fuerte |
| Portabilidad | No importante | Clave |

En software las *fallas* **NO** son consecuencia del uso y el deterioro. Las fallas ocurren como consecuencia de errores introducidos durante el desarrollo.

Mantenimiento:

- **Correctivo** (updates) errores.
- **Adaptativo** (upgrade) funcionalidad.

Ingeniería de Software: Aplicación de un enfoque sistemático, disciplinado, y cuantificable al desarrollo, operación, y mantenimiento del software.

Enfoque sistemático: Metodología y prácticas existentes para solucionar un problema dentro de un dominio determinado. Esto permite repetir el proceso y da la posibilidad de predecirlo (independientemente del grupo de personas que lo lleva a cabo).

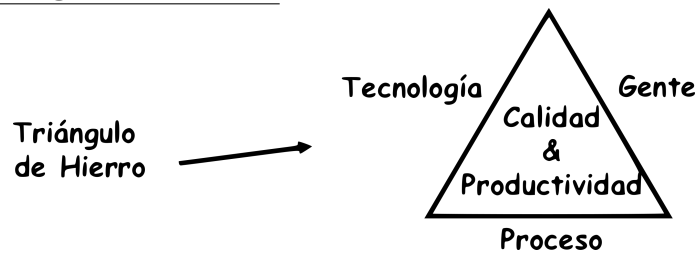
Factor principal: Satisfacer necesidades cliente/usuario

Factores de impacto:

- **Escala:** Debe funcionar para entradas pequeñas y grandes.
- **Productividad:** Reducir las *KLOC/PM*.
- **Calidad:** Densidad de defectos ($\#defectos / tamaño$) (+ fallas \Rightarrow - confiable)
 - **Funcionalidad** Capacidad de proveer funciones que cumplen las necesidades establecidas o implicadas.
 - **Confiabilidad** Capacidad de realizar las funciones requeridas bajo las condiciones establecidas durante un tiempo específico.
 - **Usabilidad** Capacidad de ser comprendido, aprendido y usado.
 - **Eficiencia** Capacidad de proveer desempeño apropiado relativo a la cantidad de recursos usados.
 - **Mantenibilidad** Capacidad de ser modificado con el propósito de corregir, mejorar, o adaptar.
 - **Portabilidad** Capacidad de ser adaptado a distintos entornos sin aplicar otras acciones que las provistas a este propósito en el producto.

- **Consistencia y repetitividad:** Sucesiva producción de sistemas de alta calidad y con alta productividad que pueda ser repetido.
- **Cambio:** Adaptarse a las necesidades.

Triángulo de hierro:



Fases del proceso de desarrollo:

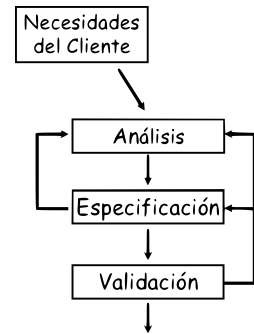
- Análisis de requisitos y especificación
- Arquitectura y Diseño
- Codificación
- Testing
- Entrega e instalación

2 Requerimientos de Software

La *SRS* **especifica** lo que el sistema propuesto debe hacer.
Establece bases de un acuerdo entre el *cliente / usuario* y el desarrollador.

Pasos para crear una *SRS*:

1. **Análisis del problema o requerimientos.**
2. **Especificación de los requerimientos.**
3. **Validación.**



2.1 Análisis del problema o requerimientos.

Objetivo: Lograr una buena comprensión de las necesidades, requerimientos y restricciones del software. Trata con el dominio del problema. Se utilizan técnicas de diagramas de flujo de datos (*DFD*), diagramas de objetos, etc.

Principio básico: *Divide y conquistarás* para comprender cada subproblema y la relación entre ellos con respecto a:

- **Funciones** (Análisis estructural)
- **Objetos** (Análisis objetos)
- **Eventos del sistema** (particionado de eventos)

2.1.1 Modelado de flujo de datos

Un *DFD* es una representación del flujo de datos a través del sistema.

- Ve al sistema como una transformación de I / O
- La transformación se realiza a través de “*Transformadores*”
- Captura la manera en que ocurre la transformación de la entrada en la salida a medida que los datos se mueven a través de los transformadores.
- No se limita al software.

2.1.2 Modelado orientado a objetos

Un sistema es visto como un conjunto de objetos interactuando entre sí (o con el usuario) a través de servicios que cada uno provee.

Objetivo:

- Identificar los objetos en el dominio del problema
- Definir las clases identificando cual es la información del estado que esta encapsula
- Identificar las relaciones entre los objetos de las distintas clases, ya sea en la jerarquía o a través de llamadas a métodos.

2.1.3 Prototipado

Cientes, usuarios y desarrolladores lo utilizan para comprender mejor el problema y las necesidades. Dos enfoques:

1. **Descartable:** (*más adecuado*) El prototipo se construye con la idea de desecharlo luego de culminada la fase de requerimientos.
2. **Evolucionario:** Se contruye con la idea de que evolucionará al sistema final.

2.2 Especificación de requerimientos:

2.2.1 Características de una SRS

- **Correcta:** Cada requerimiento representa precisamente alguna característica deseada en el sistema final.
- **Completa:** Todas las características deseadas están descritas.
- **No ambigua** Cada requerimiento tiene exactamente un significado.
- **Consistente** Ningún requerimiento contradice a otro.
- **Verificable** Cada requerimiento se debe poder verificar.
- **Rastreable** Se debe poder determinar el origen de cada requerimiento y cómo éste se relaciona a los elementos del software.
Requerimiento \Rightarrow parte de código, Parte de código \Rightarrow Requerimiento.
- **Modificable** Incorporar cambios fácilmente preservando completitud y consistencia. No redundancia.
- **Ordenada en aspectos de importancia y estabilidad** Orden de prioridades para reducir riesgos debido a cambios de requerimientos.

2.2.2 Una SRS debe especificar requerimientos sobre

1. **Funcionalidad:** Especifica toda la funcionalidad que el sistema debe proveer, las salidas que debe producir para cada entrada y las relaciones entre ellas, todas las operaciones que el sistema debe realizar. Las entradas válidas y el comportamiento del sistema para entradas inválidas, errores u otras situaciones anormales.
2. **Requerimientos de desempeño:** Todas las restricciones en el desempeño del sistema de software.
 - **Requerimientos dinámicos:** Especifican restricciones sobre la ejecución.
 - **Requerimientos estáticos:** No imponen restricción en la ejecución.

Todos los requisitos se especifican en términos medibles, verificables.

3. **Restricciones de diseño:** Existen factores en el entorno del cliente que pueden restringir las elecciones de diseño. Ej: Ajustarse a estándares.
4. **Requerimientos de interfaces externas:** Todas las interacciones del software con gente, hardware y otros softwares deben especificarse claramente. La interfaz con el usuario debe recibir la atención adecuada.

2.2.3 Casos de uso

Conceptos básicos:

- **Actor:** Persona o sistema que interactúa con el sistema propuesta para alcanzar un objetivo.
- **Actor primario:** Actor principal que inicia el caso de uso.
- **Escenario:** Conjunto de acciones realizadas con el fin de alcanzar un objetivo bajo determinadas condiciones.
- **Escenario exitoso principal:** Cuando todo funciona normalmente y se alcanza el objetivo.
- **Escenario alternativos (de extensión/ de excepción):** Cuando algo sale mal y el objetivo no puede ser alcanzado.

2.3 Validación

2.3.1 Errores más comunes

- Omisión
- Inconsistencia
- Hechos incorrectos
- Ambigüedad

La *SRS* la revisan un grupo de personas, conformado por: Autor, cliente, representantes de usuarios y de desarrolladores.

2.4 Métricas

Para poder estimar costos y tiempos y planear el proyecto se necesita “medir” el esfuerzo que demandará.

2.4.1 Punto función

Es una métrica como las *LOC*, se determina solo con la *SRS*, define el tamaño en términos de funcionalidad

Tipos de Funciones:

- Entradas externas
- Salidas externas
- Archivos de interfaz externa
- Archivos lógicos internos
- Transacciones externas

Punto función no ajustado (UFP): $\sum_{i=1}^5 \sum_{j=1}^3 W_{ij} C_{ij}$

Características:

1. Comunicación de datos
2. Procesamiento distribuido
3. Objetivos de desempeño
4. Carga en la configuración de operación
5. Tasa de transacción
6. Ingreso de datos online
7. Eficiencia del usuario final
8. Actualización online
9. Complejidad del procesamiento lógico
10. Reusabilidad
11. Facilidad para la instalación
12. Facilidad para la operación
13. Múltiples sitios
14. Intención de facilitar cambios

Factor de ajuste de complejidad (CAF): $0.65 + 0.01 \sum_{i=1}^{14} P_i$

Puntos función: CAF * UFP

2.4.2 Métrica de calidad

- Directa: Evalúan la calidad del documento estimando el valor de los atributos de calidad de la SRS.
- Indirecta: Evalúan la efectividad de las métricas del control de calidad usadas en el proceso en la fase de requerimientos.