



Politécnico
Internacional

INGENIERIA DE SOFTWARE - I



Harol Hernán Torres Neuta
Magister en Educación en Tecnología
Licenciado en Informática

**“Cualquier tecnología suficientemente
avanzada es equivalente a la magia..”**

Arthur C. Clarke

ACUERDOS PEDAGÓGICOS

ACUERDOS PEDAGÓGICOS

FECHAS DE LAS SESIONES									
Semana 1	Semana 2	Semana 3	Semana 4	Semana 5	Semana 6	Semana 7	Semana 8	Semana 9	Semana 10
29/04/22	06/05/22	13/05/22	20/05/22	27/05/22	03/06/22	10/06/22	17/06/22	24/06/22	01/07/22

ACUERDOS PEDAGÓGICOS

ACUERDOS	
1. PUNTUALIDAD Y ASISTENCIA:	
1.1	<i>Hora de inicio:</i> 6:15 pm
1.2	<i>Día y hora encuentros sincrónicos:</i> Viernes, 6:00 pm
1.3	<i>Trabajo autónomo:</i> El estudiante es responsable por el trabajo autónomo.
1.4	<i>Foro de dudas académicas:</i> En este foro el estudiante presentará las inquietudes que tenga con el desarrollo de la asignatura. El docente responderá dentro de las 12 horas hábiles siguientes. En ningún caso el tiempo de respuesta de la docente puede ser superior a 24 horas después de la solicitud del estudiante. Los estudiantes revisarán las preguntas consignadas por sus compañeros, para identificar si la inquietud que tiene ya fue planteada por un compañero y respondida por el docente. En caso que en el foro no haya una pregunta igual, registrará su pregunta allí. Esto para hacer más eficiente el tiempo de respuesta.
1.5	<i>Vídeo conferencia:</i> La vídeo conferencia es el encuentro sincrónico (personal, en línea). En cada sesión se realizarán dos vídeo conferencias, la cuales se aclaran en el desarrollo de la cada sesión.

ACUERDOS PEDAGÓGICOS

6. CÓDIGO DE CONDUCTA

6.1 Se trabajará en un clima de respeto, orden y cordialidad entre alumnos y docente. Se considera falta de respeto una agresión verbal de cualquier tipo.

La docente se compromete a cumplir los horarios acordados tanto para encuentros sincrónicos, como para retroalimentaciones.

6.2 Las entregas de trabajos se han pautadas con anticipación.

6.3 Es necesario que en las vídeo conferencias los estudiantes mantengan los micrófonos y cámaras apagados. La participación debe ser por el chat, excepto que la docente les indique que pueden participar por audio o vídeo.

6.4 El cometer faltas en las actividades, como plagio (adueñarse de propiedad intelectual sin referenciar al autor), realización de tareas a otros compañeros, duplicar actividades elaboradas por otros compañeros, entre otros, implica un 0.0 en la nota, al igual que en la nota cualitativa del profesor y se iniciará un proceso disciplinario.

6.5 Este acuerdo pedagógico se socializa en la primera sesión. Es susceptible de modificaciones según lo acordado entre estudiantes y docente. Una vez quede definitivo en sesión 1, la docente lo publicará en el foro “Noticias”, en el cual debe escribir cada estudiante indicando su conformidad o no, durante los siguientes 3 días hábiles. Para los estudiantes que no participen en el foro durante este tiempo, se entenderá que lo aceptan



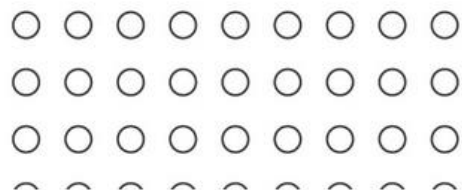
Transformamos **la evaluación** para mejorar

A partir de este ciclo los **porcentajes en las evaluaciones de cada corte** han cambiado de la siguiente forma:

1^{er} corte **15%**
2^{do} corte **35%**
3^{er} corte **50%**



Si tienes alguna inquietud
consulta con tu mentor



Tercer corte (Final): semana 10. Las notas obtenidas durante las semanas 9 y 10 más la nota de la evaluación parcial tienen un peso del 30% en la nota final, cumpliendo así con el 100% de la nota de la asignatura.

Primer corte (Parcial 1): semana 4. Las notas obtenidas a esta fecha más la nota de la evaluación parcial tienen un peso del 35% de la nota final.

Segundo corte (Parcial 2): semana 8. Las notas obtenidas entre las semanas 5 y 8 más la nota de la evaluación parcial tienen un peso del 35% en la nota final.



ESPACIOS DE FORMACIÓN

1	Viernes	6 PM a 10:30 PM - 1 Hora
2	Break	Por acordar.

A TENER EN CUENTA



1. Cerrar los micrófonos y desactivar las cámaras.



2. Solicitar la palabra mediante el chat.



3. Preguntar

CONTENIDOS

ASIGNATURA:	INGENIERÍA DE SOFTWARE I
CICLO	V
CODIGO	1647
CREDITOS	3
INTENSIDAD HORARIA	4.5 Horas semanales
PRERREQUISITOS	Ninguno
CORREQUISITOS	Ninguno
COMPETENCIA DE ENTRADA	El estudiante clasifica vulnerabilidades utilizando los estándares de matriz de riesgo y vulnerabilidades para apoyar el sistema de gestión de seguridad de la información.
COMPETENCIA DE SALIDA	El estudiante clasifica vulnerabilidades utilizando los estándares de matriz de riesgo y vulnerabilidades para apoyar el sistema de gestión de seguridad de la información.
PERFIL DOCENTE	Ingeniero de sistemas, electrónicos con experiencia en desarrollo de software
VERSIÓN	02
FECHA DE IMPLEMENTACIÓN	2020-5T

COMPETENCIA DE SALIDA	
El estudiante elabora modelos de prototipos de software teniendo en cuenta el lenguaje UML y la metodología RUP para cumplir con los requerimientos del cliente.	
INDICADORES DE DESEMPEÑO	
1.1 Recolecta información sobre el software utilizando técnicas de elicitación para el análisis de los requerimientos establecidos por el cliente.	
1.2 Construye diagramas de un proyecto de software, enmarcado en el estándar UML.	
1.3 Diligencia las plantillas RUP de un producto software tomando como punto de partida los requerimientos funcionales establecidos por el cliente.	
CONOCIMIENTOS ASOCIADOS/CONTENIDOS	<ul style="list-style-type: none"> • Introducción al modelado a objetos • Conceptos Básicos del modelado • Lenguaje de Modelado Unificado (UML) • Elementos estructurales, de comportamiento, de agrupación, de anotación. • Relaciones • Diagramas: Casos de uso, clases, secuencia • Herramientas CASE para el modelado de software • Metodologías Tradicionales de Implementación de Software (RUP) • Fase de planificación y especificación de requisitos
REQUERIMIENTOS DE INFRAESTRUCTURA O MEDIOS EDUCATIVOS	<ul style="list-style-type: none"> • Sala de Software. Computadores con mínimo procesador Core I3 o equivalente y 4 GB en RAM. Proyector. • Herramienta de Software de diagramación UML, como astah (https://astah.net/) o StarUML (http://staruml.io). • Herramienta de diagramación online como Lucidchart (https://www.lucidchart.com/)

SESIÓN	CONTENIDO	OBJETIVO DE SESIÓN
1	<ul style="list-style-type: none"> • Conceptos Generales de Metodología <ul style="list-style-type: none"> ○ Por qué utilizar una metodología ○ Requisitos de una metodología ○ Metodologías más utilizadas 	Identifica las características principales de las etapas del ciclo de vida de software con el fin de alcanzar la calidad del software en ambientes de desarrollo.
	<ul style="list-style-type: none"> • Ciclo de Vida del Software <ul style="list-style-type: none"> ○ Análisis preliminar ○ Análisis de requerimientos <ul style="list-style-type: none"> ▪ Requerimiento ▪ Requisito ○ Diseño ○ Desarrollo y Construcción ○ Pruebas <ul style="list-style-type: none"> ▪ Caja Negra ▪ Caja Blanca ▪ Versiones Alfa ▪ Versiones Beta ○ Documentación <ul style="list-style-type: none"> ▪ Interna ▪ Externa ○ Mantenimiento • Tipos de Cliente <ul style="list-style-type: none"> ○ Atención al cliente • Levantamiento de información • Procesos de análisis de requerimientos • Técnicas de Elicitación 	

2	<ul style="list-style-type: none">• Perspectiva general de UML<ul style="list-style-type: none">○ Breve resumen de UML○ Historia de UML○ Objetivos de UML○ Complejidad de UML○ Valoración de UML○ Áreas conceptuales de UML• Vistas de UML• Vista de casos de uso<ul style="list-style-type: none">○ Descripción○ Actores○ Relaciones<ul style="list-style-type: none">▪ Asociación▪ Generalización▪ Extensión▪ Inclusión○ Casos de uso	<p>Emplea las técnicas y las herramientas disponibles para el levantamiento de información, de tal forma que le permita al estudiante, la obtención y análisis de datos, enmarcados en el desarrollo de software.</p> <p>Construye diagramas de casos de uso que cumplan con el estándar UML, además, se acoplen a las mejores prácticas de desarrollo y representen eficientemente los requerimientos funcionales definidos por un cliente</p>
---	--	---

	<ul style="list-style-type: none"> ○ Representación gráfica de casos de uso 	
3	<ul style="list-style-type: none"> • Vista de clases <ul style="list-style-type: none"> ○ Clases ○ Operaciones (Métodos) ○ Cardinalidad de las Relaciones ○ Relaciones entre Clases (Clasificación) ○ Polimorfismo ○ Flujo de objetos ○ Asociaciones ○ Simbología 	Construye diagramas de clases que cumplan con el estándar UML, además, se acoplen a las mejores prácticas de desarrollo y representen eficientemente los requerimientos funcionales definidos por un cliente
4	Primer parcial	Utiliza técnicas de elicitación que facilitan el levantamiento y análisis de los requerimientos establecidos por el cliente.
5	<ul style="list-style-type: none"> • Vista interacción (Secuencia) <ul style="list-style-type: none"> ○ Descripción ○ Flujo de datos ○ Asociaciones ○ Simbología • Análisis de requerimientos <ul style="list-style-type: none"> ○ Representación gráfica de diagramas de secuencia. • Metodología RUP <ul style="list-style-type: none"> ○ Conceptos generales ○ Características ○ Marco histórico ○ Configuración y personalización 	Construye diagramas de secuencia que permitan representar objetos, interfaces y mensajes con el objeto de describir el comportamiento de un sistema, subsistema u operación, teniendo como referencia el patrón MVC y los lineamientos del estándar UML.
6	<ul style="list-style-type: none"> • Vista de actividades (estados) <ul style="list-style-type: none"> ○ Descripción ○ Flujo de objetos ○ Asociaciones ○ Simbología • Análisis de requerimientos 	Representa eficientemente la naturaleza dinámica de un sistema, subsistema, proceso u operación, mediante el modelado del flujo ocuriente de actividades.

	<ul style="list-style-type: none"> Representación gráfica de diagramas de actividades 	
7	<ul style="list-style-type: none"> Fases de Desarrollo <ul style="list-style-type: none"> Inicio Elaboración Construcción Transición Disciplinas Artefactos <ul style="list-style-type: none"> Visión Casos de uso Glosario Pruebas 	Identifica las fases de desarrollo, disciplinas y artefactos que hacen parte de la metodología RUP, al momento de documentar un producto software.
8	Segundo parcial	Construye diagramas de un proyecto software, enmarcado en el estándar UML.
9	<ul style="list-style-type: none"> Desarrollo e implementación de plantillas RUP <ul style="list-style-type: none"> Documento Visión Casos de Uso Plan de Desarrollo 	Documenta un proyecto software, utilizando las plantillas estandarizadas RUP.
10	Evaluación final	Construye prototipos de productos software, tomando como referencia las plantillas RUP y los requerimientos funcionales establecidos por el cliente.

PRESENTACIÓN DE ESTUDIANTES

1. Nombre
2. A que se dedica
3. ¿Que espera de esta asignatura?
4. ¿Que experiencia previa tiene?
5. En que proyectos de desarrollo ha trabajado.



<https://padlet.com/harol003/u80d6k5laexarjht>

Todos deberían aprender a programar

<https://www.youtube.com/watch?v=Y1HHBXL9bg>

"Everybody in this country should
learn how to program a computer..."

"Todo el mundo en este país debería aprender a programar un

**¿Por qué es importante la Ingeniería de Software
en la labor de un ingeniero?**

PAUSA ACTIVA

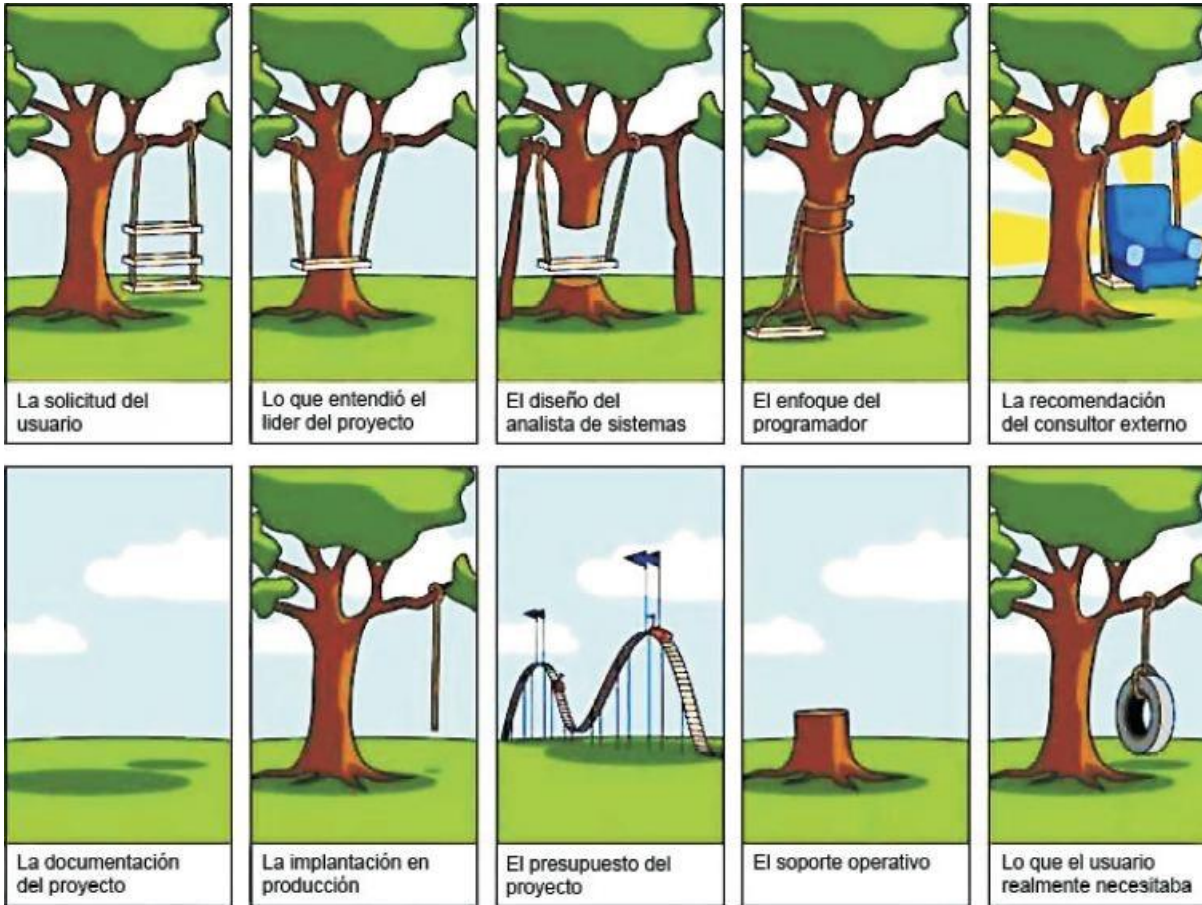
10 Min

MANOS A LA OBRA



REQUISITOS DE SOFTWARE

REQUISITOS DE SOFTWARE



Los requisitos software son la descripción de las características y las funcionalidades del sistema 'target'. Los requisitos nos comunican las expectativas de los consumidores de productos software. Los requisitos pueden ser obvios o estar ocultos, conocidos o desconocidos, esperados o inesperados, des del punto de vista del cliente.

Este término se utiliza para referirse a cualquier persona que tiene influencia directa o indirecta sobre los requisitos del sistema. Entre los **STAKEHOLDERS** se encuentran los usuarios finales que interactúan con el sistema y todos aquellos en la organización se que verán afectados por dicho sistema.

REQUISITOS DE SOFTWARE



REQUISITOS DE SOFTWARE

Los **STAKEHOLDERS** también son los ingenieros que desarrollan o dan mantenimiento a otros sistemas relacionados, los Administradores del negocio, los expertos en el dominio del sistema, los representantes de los trabajadores, etc.



REQUISITOS DE SOFTWARE

Los **STAKEHOLDERS** o grupos interesados:

- Dueños, accionistas, inversionistas.
- Bancos y acreedores.
- Socios y proveedores.
- Compradores, clientes actuales y potenciales.
- La Administración de la empresa.
- Empleados, consejos de trabajo, sindicatos.
- Competidores.
- Gobierno (local, estado, nacional, internacional y reguladores.
- Asociaciones profesionales, grupos comerciales de la industria.
- Prensa.
- Organizaciones no gubernamentales.
- Opinión pública, grupos de intereses sociales, políticos, ambientales, religiosos, comunales.



REQUISITOS DE SOFTWARE

Pasos para identificar los **STAKEHOLDERS**:

1. Identificar a todos los interesados.
2. Entender las necesidades, intereses y expectativas de cada uno.
3. Balancear, reconciliar y sintetizar esos aspectos.
4. Integrar las necesidades de los interesados con las estrategias de la organización.
5. Verificar relaciones existentes entre los interesados.
6. Identificar la importancia de cada interesado: poder, influencia, interés.



REQUISITOS DE SOFTWARE

ERS → Especificación de requisitos de software

El ERS es un documento creado por los analistas de sistema después de recoger los requisitos.

El ERS define cómo va a interactuar el software que quiere crearse con el hardware, las interfaces externas, la velocidad operativa, el tiempo de respuesta del sistema, la portabilidad del software en las diversas plataformas, el mantenimiento, la velocidad de reponerse después de estropearse, su seguridad, calidad, limitaciones, etc.



Los requisitos recibidos por parte del cliente se escriben en lenguaje natural. Es responsabilidad del analista de sistemas documentar sobre los requisitos en lenguaje tecnológico para que puedan ser útiles y comprendidos por el equipo de desarrollo de software.

REQUISITOS DE SOFTWARE



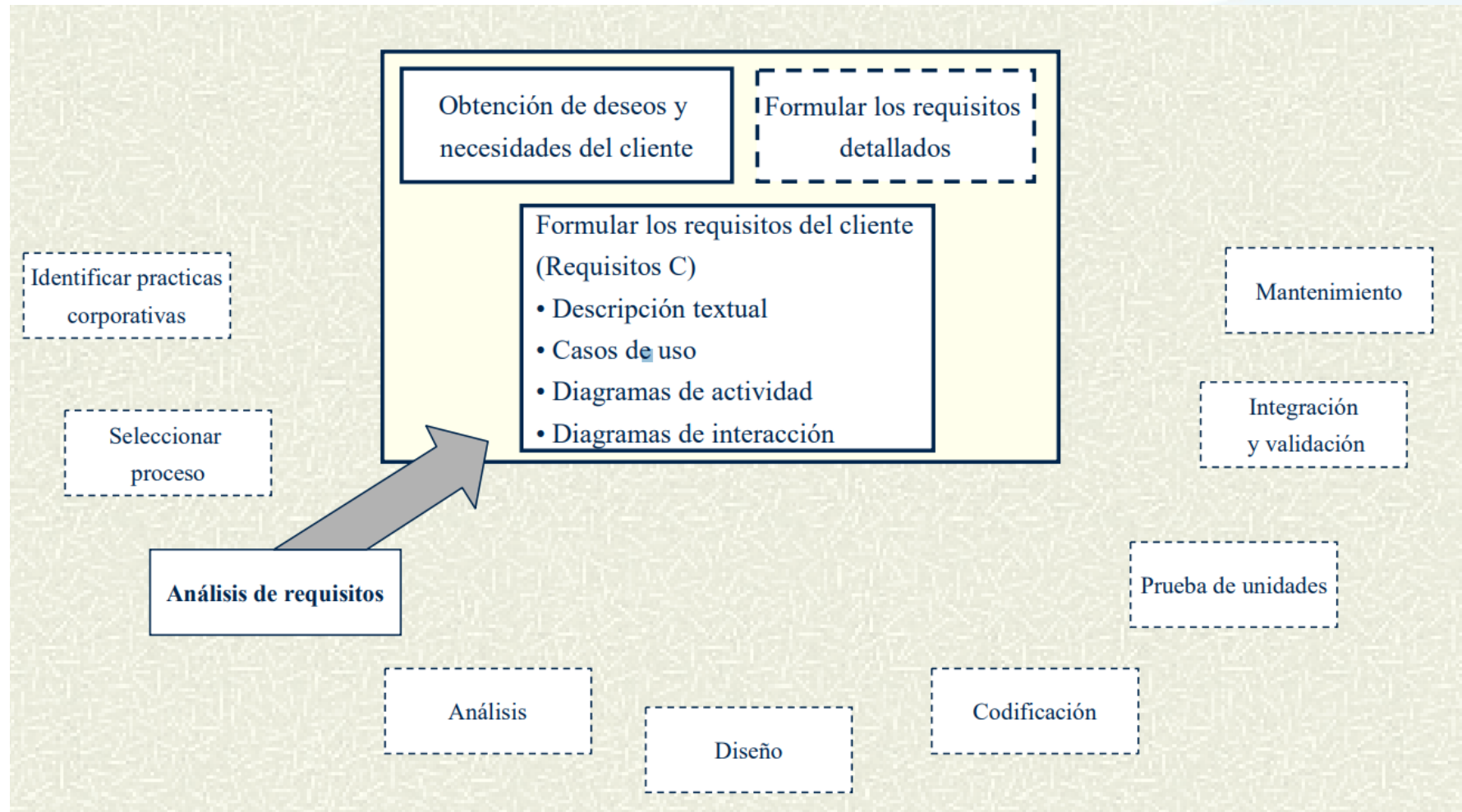
REQUISITOS DE SOFTWARE

El ERS debe venir con las siguientes características:

- Los requisitos del usuario se deben expresar en lenguaje natural.
- Los requisitos técnicos se deben expresar en lenguaje estructurado, el cual se usará dentro de la organización.
- La descripción del diseño se debe escribir en pseudocódigo.
- El formato de Forms y GUI impresiones de pantalla.
- Anotaciones condicionales y matemáticas para DFDs etc.

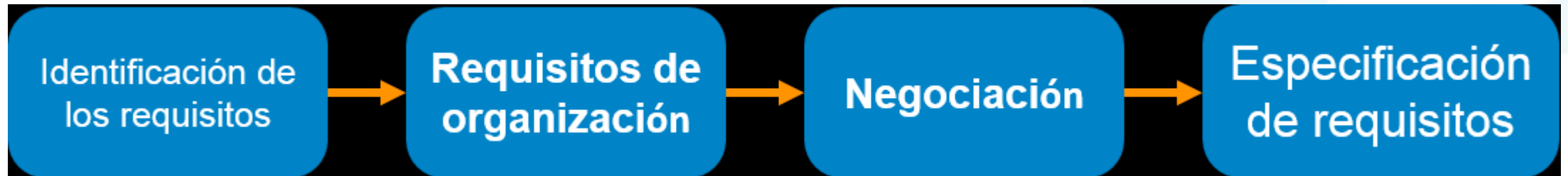


REQUISITOS DE SOFTWARE



REQUISITOS DE SOFTWARE

PROCESO REQUISITOS DE SOFTWARE

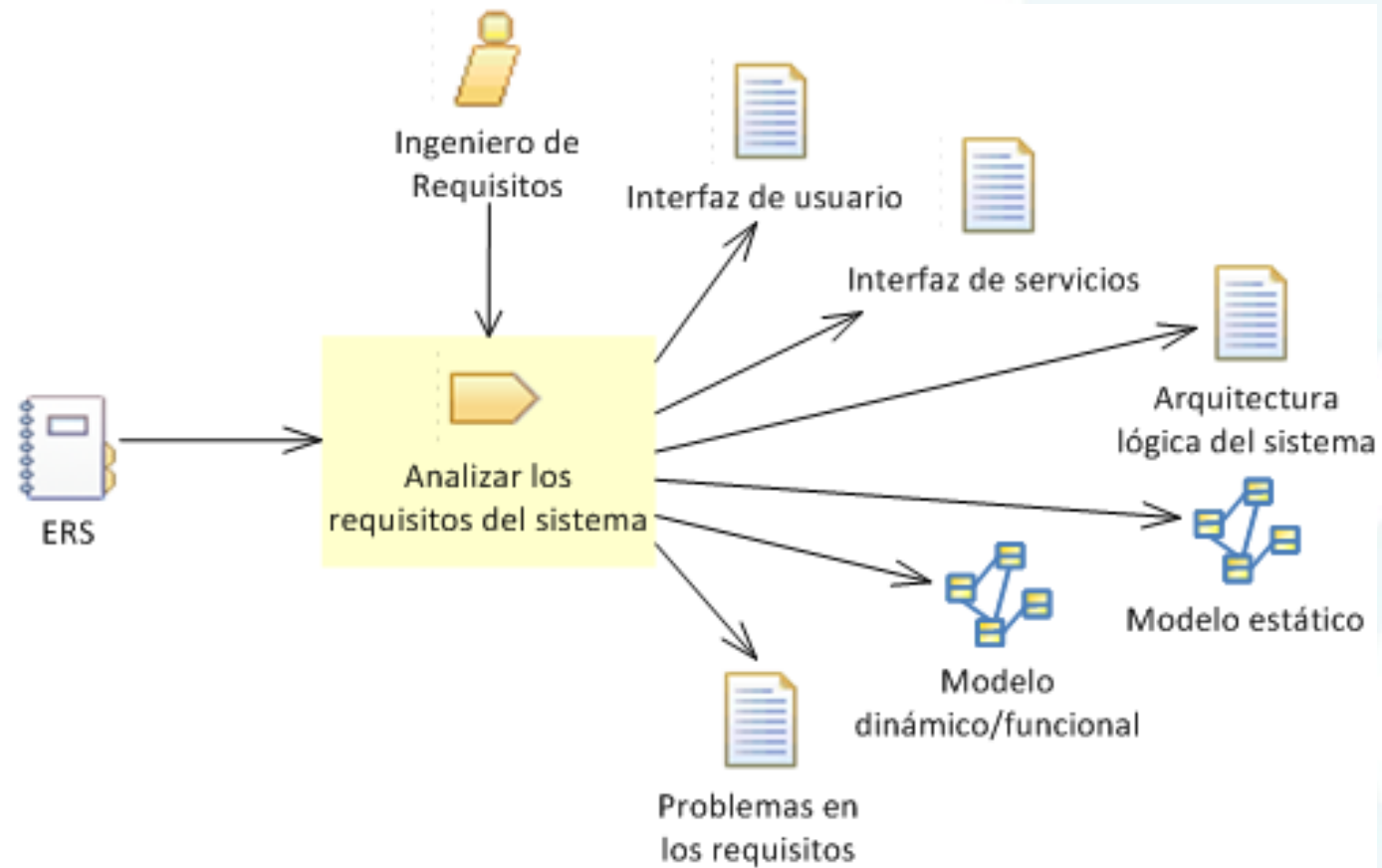


REQUISITOS DE SOFTWARE

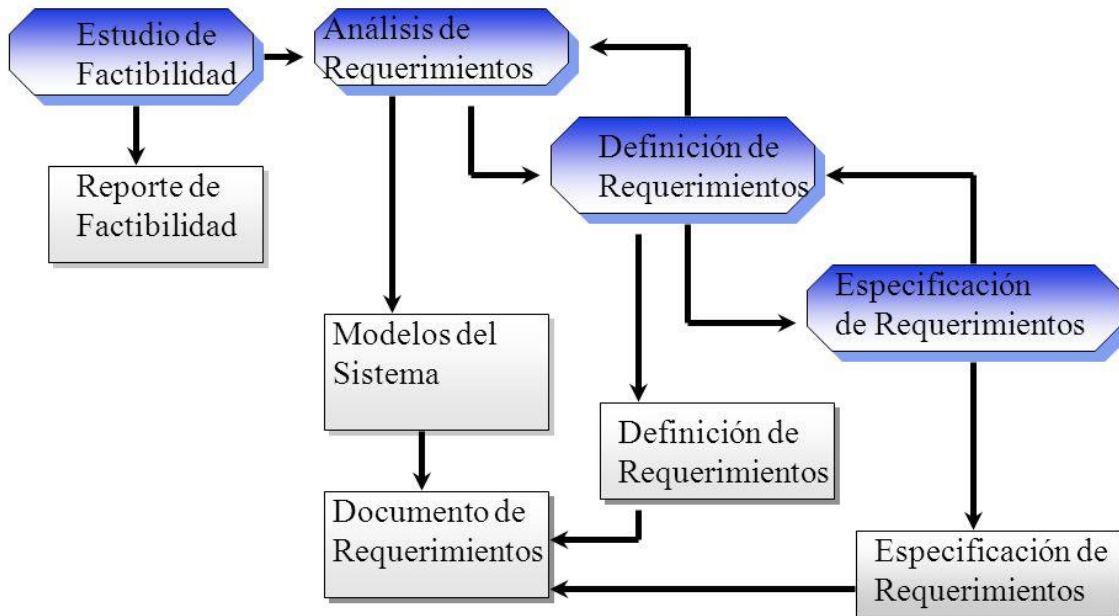
PROCESO REQUISITOS DE SOFTWARE



PROCESO REQUISITOS DE SOFTWARE + UML



El Proceso de Ingeniería de Requerimientos



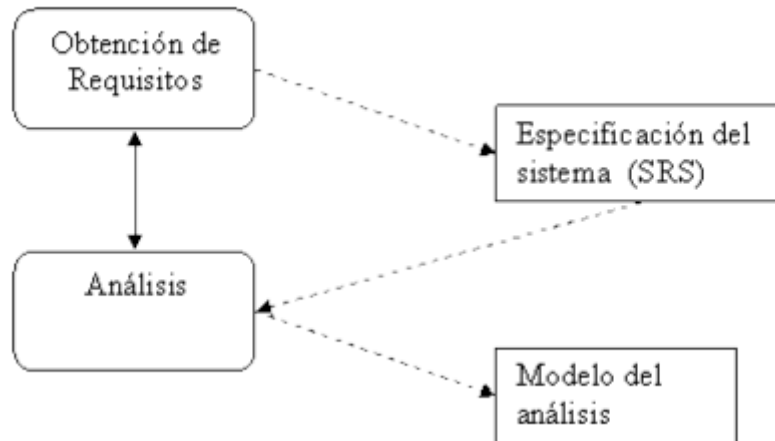
©Ian Sommerville 1995

Ingeniería de Software, 5a. edición Capítulo 4

Diapositiva 12

REQUISITOS DE SOFTWARE

REQUISITOS DE SOFTWARE

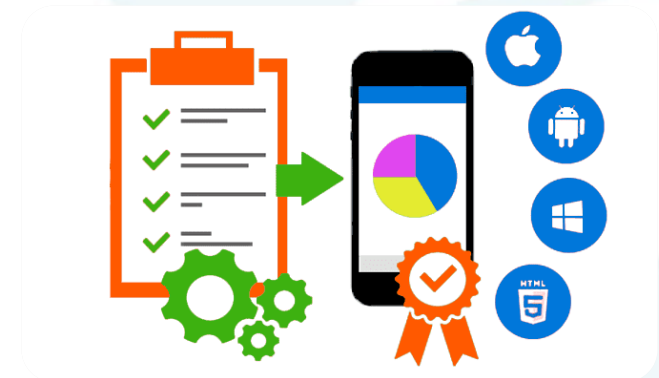


PAUSA ACTIVA

10 Min

- Funcionales
- No funcionales

REQUISITOS DE SOFTWARE



REQUISITOS FUNCIONALES: Describen las interacciones entre el sistema y su ambiente, en forma independiente a su implementación. El ambiente incluye al usuario y cualquier otro sistema externo con el cual interactúe el sistema.

REQUISITOS DE SOFTWARE



REQUISITOS DE SOFTWARE

REQUISITOS NO FUNCIONALES: Describen atributos sólo del sistema o del ambiente del sistema que no están relacionados directamente con los requisitos funcionales. Los requisitos no funcionales incluyen restricciones cuantitativas, como el tiempo de respuesta o precisión, tipo de plataforma (lenguajes de programación y/o sistemas operativos, etc.)



1. Correcta
2. No ambigua
3. Completa
4. Consistente
5. Calificada de acuerdo a la importancia y/o estabilidad
6. Verificable
7. Modificable
8. Rastreable

REQUISITOS DE SOFTWARE

CARACTERISTICAS REQUISITOS DE SOFTWARE
IEEE STD 830-1998



1. Correcta

Una SRS es correcta, sí y solo sí, cada requisito especificado es un requisito que el software debe cumplir.

2. No ambigua

Una SRS no es ambigua sí y solo sí cada requisito especificado tiene sólo una interpretación.

REQUISITOS DE SOFTWARE

CARACTERISTICAS REQUISITOS DE SOFTWARE
IEEE STD 830-1998



3. Completa

Una SRS es completa, sí y solo sí, incluye los siguientes elementos:

- a) Todos los requisitos significativos, ya sea que se relacionen a funcionalidad, desempeño, restricciones de diseño, atributos o interfaces externas. En particular cualquier requisito externo impuesto por una especificación del sistema debe ser reconocido y tratado.
- a) Definición de las respuestas del software a todos los tipos posibles de clases de datos de entrada en todos los tipos posibles de clases de situaciones. Notar que es importante especificar las respuestas tanto para valores de entrada válidos como inválidos.
- b) Etiquetas y referencias completas a todas las figuras, tablas y diagramas en la SRS así como la definición de todos los términos y unidades de medida.

REQUISITOS DE SOFTWARE

**CARACTERISTICAS REQUISITOS DE SOFTWARE
IEEE STD 830-1998**



4. Consistente

Una SRS es consistente, sí y solo sí, no se contradice a sí misma, es decir, si ningún subconjunto de requisitos ahí descritos se contradicen o entran en conflicto.

5. Jerarquizada de acuerdo a la importancia y/o estabilidad

Una SRS está calificada de acuerdo a la importancia y/o estabilidad si cada requisito tienen un identificador que indique la importancia o estabilidad del requisito.

REQUISITOS DE SOFTWARE

CARACTERISTICAS REQUISITOS DE SOFTWARE
IEEE STD 830-1998



6. Verificable Una SRS es verificable, sí y solo sí, cada requisito especificado es verificable

Un requisito es verificable sí y solo sí existe un proceso finito de costo/efectivo con el cual una persona o una máquina puede verificar que el producto de software cumple el requisito. En general cualquier requisito ambiguo no es verificable.

REQUISITOS DE SOFTWARE

CARACTERISTICAS REQUISITOS DE SOFTWARE
IEEE STD 830-1998



7. Modificable

Una SRS es modificable, sí y solo sí, su estructura y estilo son tales que, cualquier cambio a los requisitos pueden ser hechos fácil, completa y consistentemente sin perder la estructura y el estilo. Generalmente requiere que una SRS:

- a) Tenga una organización coherente y fácil de usar con una tabla de contenido, un índice y referencias cruzadas explícitas.
- b) No sea redundante (esto es, el mismo requisito no debe aparecer en más de una parte en la SRS).
- c) Expresa cada requisito de manera separada, en vez de hacerlo mezclado con otros requisitos.

REQUISITOS DE SOFTWARE

**CARACTERISTICAS REQUISITOS DE SOFTWARE
IEEE STD 830-1998**



8. Rastreable

Una SRS es rastreable si el origen de cada uno de sus requisitos es clara y si facilita la referencia de cada requisito en el desarrollo futuro o mejora de la documentación.

Se recomiendan los siguientes dos tipos de rastreabilidad:

- a) Rastreabilidad hacia atrás (esto es, a estados previos del desarrollo). El requisito tiene referencias explícitas a sus fuentes en documentos anteriores.
- b) Rastreabilidad hacia enfrente (esto es, a todos los documentos derivados del SRS). Depende de que cada requisito en la SRS tenga un nombre único o número de referencia. Es particularmente importante cuando el software entra en operación y mantenimiento. Cuando el código y los documentos de diseño son modificados, es esencial contar con la capacidad para conocer el conjunto completo de requisitos que pueden ser afectados por esas modificaciones.

REQUISITOS DE SOFTWARE

**CARACTERISTICAS REQUISITOS DE SOFTWARE
IEEE STD 830-1998**





... la **Licitación Pública** debe tener como mínimo un cartel, en donde se indiquen:

- las condiciones generales,
- las especificaciones técnicas,
- financieras y de calidad...

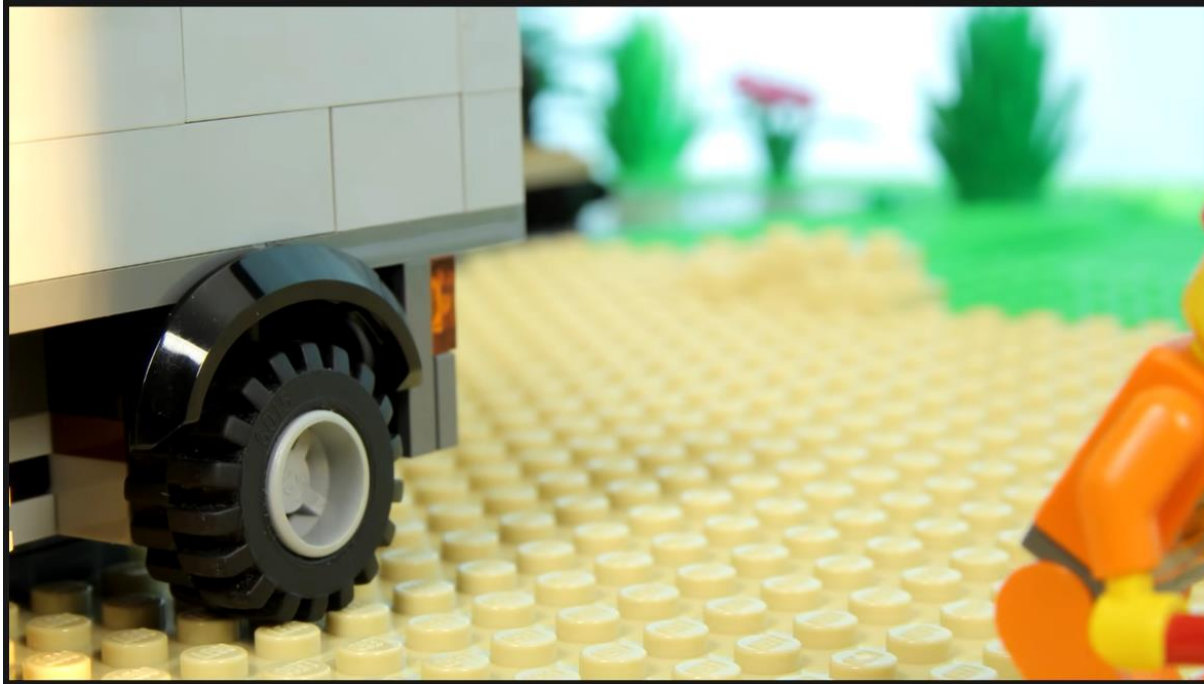
... pudiendo programar audiencias previas con potenciales oferentes para recibir observaciones.

REQUISITOS DE SOFTWARE

PROCESO REQUISITOS DE LICITACIÓN PÚBLICA

¿Y entonces como vamos a trabajar?



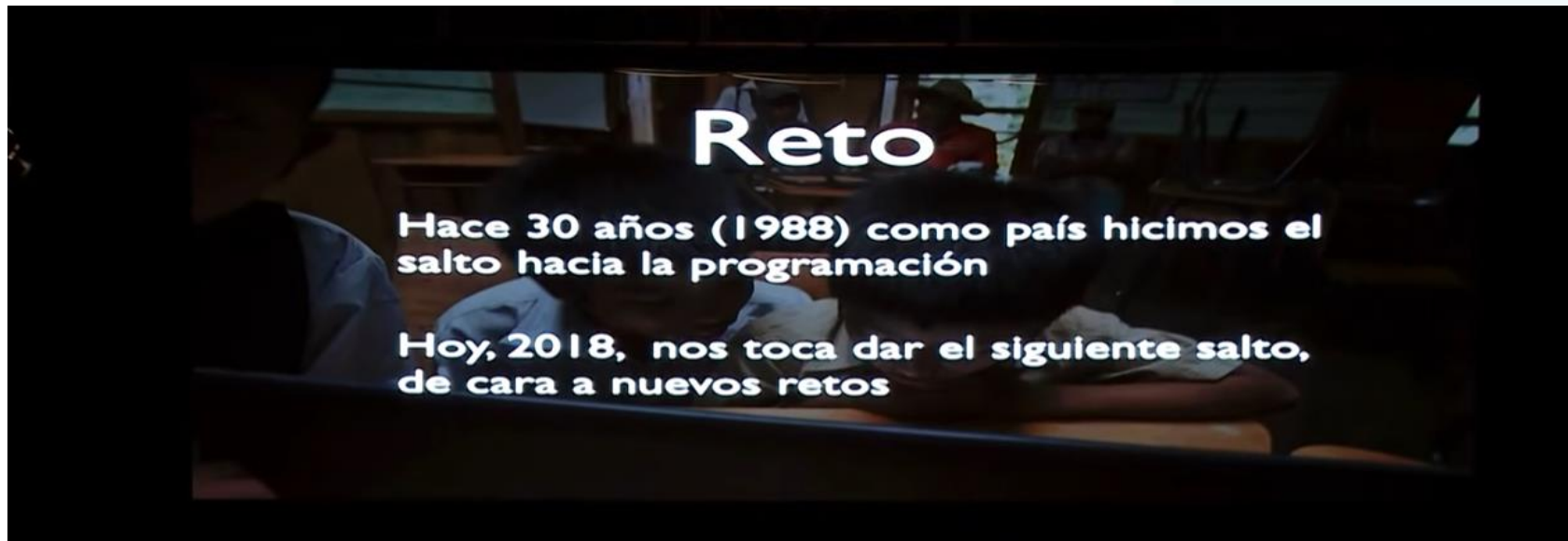


<https://www.youtube.com/watch?v=UDFgCo68tJs>

Lego Construction Site (Skyscraper Building, Mobile Crane, Excavator)

<https://www.youtube.com/watch?v=0TFZfwB67Y&t=848s>

[Pensamiento computacional | Alberto Cañas | TEDxPuraVidaED](#)



¿preguntas?

CONCLUSIONES