Desarrollo de Software I

Facultad de Ingeniería
Escuela de Ingeniería de Sistemas y Computación

Mauricio Gaona
mauricio.gaona@correounivalle.edu.co
Profesor

2025-I

Objetivos del curso

Objetivo general

Proporcionar al estudiante las bases conceptuales fundamentales de la ingeniería de software y los elementos metodológicos necesarios para llevar a cabo el desarrollo de aplicaciones de software.

Objetivos específicos

- Entender los **conceptos fundamentales** de la ingeniería de software.
- Entender y usar **metodologías ágiles** de desarrollo de software para ser un miembro efectivo y eficiente en un equipo de desarrollo ágil.
- Entender los conceptos principales de las **metodologías tradicionales** de desarrollo de software.
- Entender y aplicar los **roles** y las actividades que se realizan en las etapas de **análisis, diseño, codificación, pruebas y despliegue** de una aplicación de software.

Objetivos del curso

Objetivos específicos

Identificar **buenas prácticas** aplicables a cada momento de un proceso de desarrollo de aplicaciones web con el fin de facilitar la creación colectiva de software.

Aplicar algunos modelos, herramientas, lenguajes y plataformas, para implementar una aplicación de software usando **tecnologías de tendencia**.

Entender como **estimar** un producto de software (tiempo y valor en $$$ de una aplicación de software)

Desarrollar una aplicación de software.

Desarrollo de Software I

Desarrollo del curso

Metodología
Clase presenciales y virtuales preparadas por el profesor y con participación de los estudiantes

Evaluación (Por competencias)
Examen, Quices, Tareas, Exposiciones, Proyecto, informes escritos y comunicación oral

Examen	30
Quices	10
Tareas	10
Exposiciones	10
Proyecto	35
Informes escritos	2.5
Comunicación oral	2.5

Materiales del curso
Campus Virtual y Enlaces web

Contenido
Temas a tratar en el curso

Bibliografía
Libros y Artículos recomendados

Desarrollo de Software I

Aspectos a tratar

Contenido

1	Conceptos Generales. El proceso de desarrollo de software: Ciclo de vida de desarrollo del software, modelos de ciclo de vida del software, conceptos sobre metodologías de desarrollo de software, otros modelos
2	El Proceso de Realización de un Producto de Software. - Metodologías Ágiles para el desarrollo de software (**Scrum**, XP, Kanban y otras) - Metodologías tradicionales (Artefactos de las metodologías tradicionales)
3	Análisis y especificación de requerimientos (funcionales y no funcionales) mediante el Product Backlog . Arquitecturas para el diseño de sistemas de software.
4	Planeación y desarrollo de un producto de software Usando prácticas ágiles: Prácticas de gestión y prácticas técnicas Sprint cero (Una idea, Visión del producto, necesidades del producto, equipo de trabajo, fechas críticas, Vistas de diseño del sistema, Tecnología a usar, Release plan, Priorización, Estimación y criterios de aceptación).
5	Aspectos conceptuales del world wide web (www) Protocolos web, anatomía de la URL El rol de los Servidores y Clientes web, peticiones Get y Post Front-end y Back-end

Aspectos a tratar

Contenido

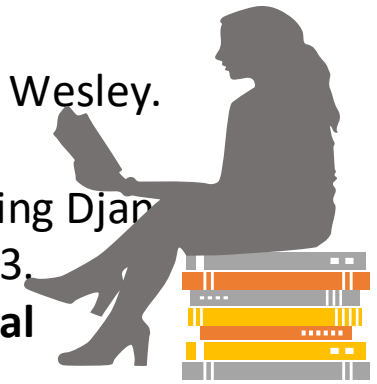
5	Prácticas para el agilismo • Uso de prácticas ágiles como (reunión diaria, iteraciones cortas, planeación de la iteración, pruebas unitarias, incrementos de software (pequeñas entregas) y retrospectivas, entre otras.)
6	Implementación • Diseño de APIs • Mapeo de los diseños para codificación según la arquitectura definida
7	Pruebas de software • Conceptos sobre pruebas • Pruebas unitarias, pruebas funcionales y pruebas End-to-End • Construcción de pruebas para los diferentes componentes de un proyecto de software
8	Estimación de tiempos y costos de una aplicación de software
9	Estrategias de despliegue de aplicaciones de software • Servidores físicos, maquinas virtuales, contenedores y plataformas • Despliegue local, despliegue en la nube
10	[Conceptos sobre: Aplicaciones en la Nube (IaaS, PaaS, SaaS)]
11	[Tendencias emergentes en las aplicaciones web: **Ingeniería de Software Aumentada con IA**]

Bibliografía

- The AI-Augmented Developer: Leveraging AI Tools to Enhance Software Development, 2024, ALBERT TETTEH ADJEI, ISBN B0D6Z2L23B.
- Concise Guide to Software Engineering: From Fundamentals to Application Methods, Gerard O´Regan, 2017.
- The Agile Samurai: How agile masters delivery great software. Jonathan Rasmusson, ISBN 13-978- 1-934356-58-6, 2017.
- User Story Mapping, Jeff Patton and Peter Economy, 2014.
- Ingeniería del Software (8a edición), Ian Sommerville, ISBN: 978-0-07-802212-8, Adisson Wesley, 2015.
- Adam Przybyłek and Miguel Ehécatl Morales-Trujillo. Advances in Agile and User-Centred Software Engineering: Third International Conference on Lean and Agile Software Development, LASD 2020
- Making Sense of Agile Project Management: Balancing Control and Agility, Charles G. Cobb, Jonh Wiley & Sons, ISBN: 978-0-470-94336-6 (2021)
- Booch G; Rumbaugh J; Jacobson I. The Unified Modeling Language, Reference Manual. Addison Wesley. 1996.

Web Development with Django: A definitive guide to building modern Python web applications using Djan
de Ben Shaw (Author), Saurabh Badhwar (Author), Chris Guest (Author), Bharath Chandra K S. 2023.

- **Recursos online: Artículos y direcciones web (URL) que estarán disponibles en el campus virtual**
- Tutorial Django: https://www.youtube.com/watch?v=7XO1AzwkPPE



Desarrollo de Software I

Por que estudiar ingeniería de software ?

01 Ingeniería de software

APLICAR CONOCIMIENTO

La Ingeniería de Software aplica el conocimiento y la comprensión teórica obtenidos a través de la computación para la creación de aplicaciones de software de calidad.

02 Ingeniería de software

ALTA DEMANDA

Necesidad creciente de desarrolladores de software con talento. A medida que avanza la tecnología, se busca la capacidad de crear software de calidad teniendo en cuenta el diseño, el desarrollo, la integración, aspectos éticos entre otros.

03 Ingeniería de software

DISCIPLINA MADURA

Como una disciplina que madura, el software se vuelve cada vez más importante en nuestra vida cotidiana.

04 Ingeniería de software

RECONOCIMIENTO INTERNACIONAL

Es una profesión de clase mundial. En un futuro dominado por la IA, los ingenieros de software asumirán roles más estratégicos y especializados, como diseñadores de sistemas, auditores éticos, integradores tecnológicos y supervisores de IA.

Herramientas

Algunas herramientas requeridas

Ambiente de trabajo: AV o MV

Herramientas de desarrollo : **Front end**
HTML5
JavaScript y CSS3

Back-end

Python
Django

Servidor Web: Apache, nginx

Bases de datos: PostgreSQL, Oracle

Herramientas de Gestión ([Google Docs, Jira, Taiga])
Manejador de versiones : GitHub , Bitbucket

IDE

Sonar: SonarLint

https://marketplace.visualstudio.com

https://aws.amazon.com/es/education/awseducate/

https://education.github.com/

https://azure.microsoft.com/es-es/free/students/

Conceptos

¿Qué es software ?

El software es un conjunto estructurado de instrucciones codificadas en un lenguaje de programación, que, al ser ejecutadas por un sistema informático (hardware), le permiten realizar tareas específicas y procesar información.

I Instrucciones (programas de computador) que cuando se ejecutan proveen la funcionalidad y rendimiento requerido.

D Documentos que describen la operación y uso de los programas.

E Estructuras de datos que permiten a los programas manipular la información adecuadamente.

Desarrollo de Software I

Conceptos

Que es Ingeniería de Software?

La ingeniería de software es una disciplina que aplica principios, metodologías y prácticas de la ingeniería al diseño, desarrollo, implementación, pruebas, despliegue, monitoreo y mantenimiento de software. Se enfoca en la creación sistemática y eficiente de software de alta calidad que satisfaga las necesidades y expectativas de los usuarios.

Conceptos: Aspectos clave de la Ingeniería de software

5. Gestión de proyectos

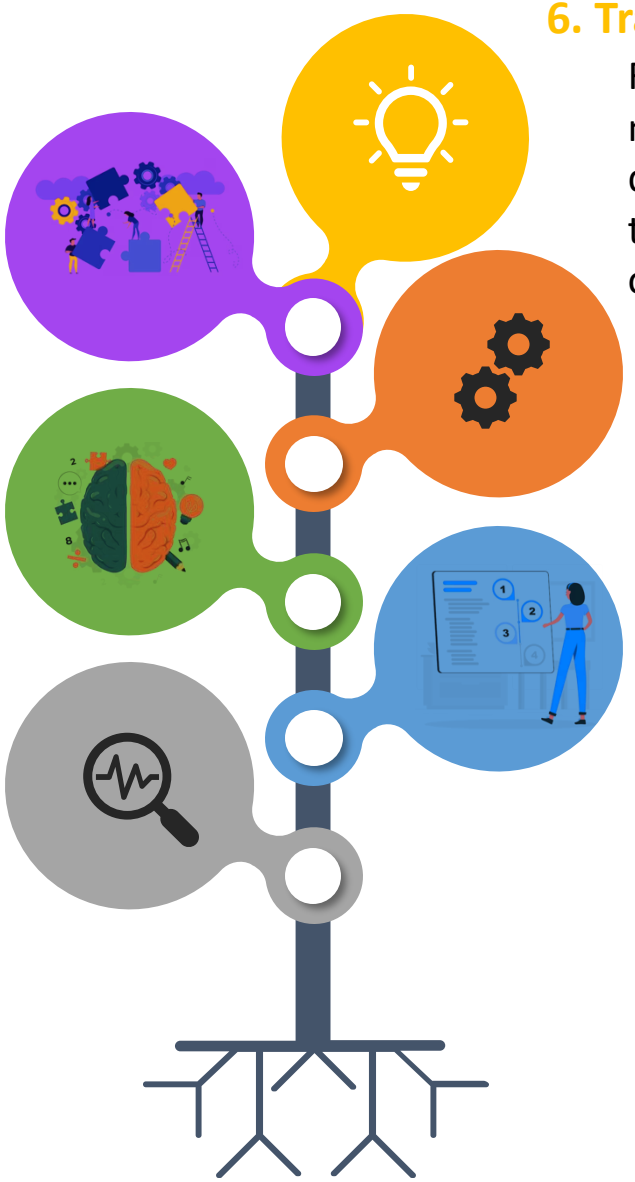
Aplica técnicas de gestión de proyectos para planificar, organizar y controlar el desarrollo de software, asegurando que se entregue a tiempo y dentro del presupuesto.

3. Metodologías y prácticas

Emplea diversas metodologías (ágiles, cascada, etc.) y prácticas (control de versiones, integración continua, etc.) para gestionar el desarrollo de software de manera efectiva.

1. Aplicación de principios de ingeniería

La ingeniería de software utiliza conceptos de ingeniería como el análisis de requisitos, el diseño, la construcción, las pruebas y el mantenimiento para garantizar que el software sea confiable, eficiente y escalable.



6. Trabajo en equipo

Fomenta la colaboración entre equipos multidisciplinarios, incluyendo desarrolladores, analistas, diseñadores, testers y otros profesionales, para lograr los objetivos del proyecto.

4. Calidad del software

Se centra en la creación de software que sea funcional, confiable, usable, eficiente, mantenible y portable.

2. Ciclo de vida del desarrollo de software

Sigue un proceso estructurado para guiar el desarrollo del software desde la concepción hasta la implementación y el mantenimiento, incluyendo fases como planificación, análisis, diseño, codificación, pruebas y despliegue.

Conceptos

Principales problema de los proyectos de software

Problemas de calidad

Entregas tardías

Sobrecosto (fuera del presupuesto estimado)

Conceptos

Programación Vs Desarrollo de software

La programación es el proceso de trasladar un problema de su ambiente físico a un lenguaje que el computador pueda entender y obedecer.

- Uno o dos desarrolladores
- Aplicaciones pequeñas
- Vida corta
- Uno o pocos participantes
- Normalmente construido desde cero
- Mantenimiento mínimo

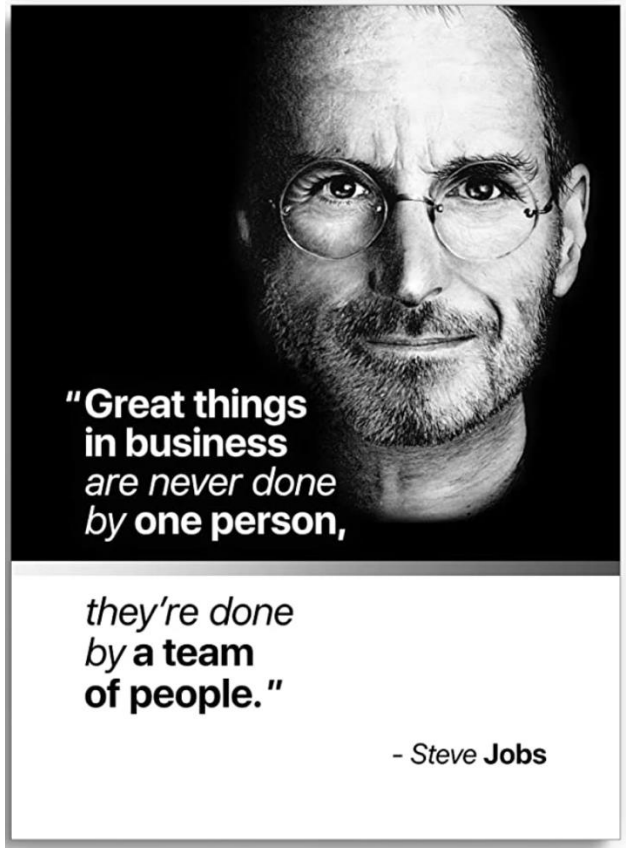
Conceptos

Programación Vs Ingeniería de software

- Equipos de desarrolladores con múltiples roles.
- Sistemas complejos.
- Infinito crecimiento.
- Muchos participantes.
- Reutilización para reducir costos.
- Requieren mantenimiento.
- Nuevas tecnologías para evitar obsolescencia

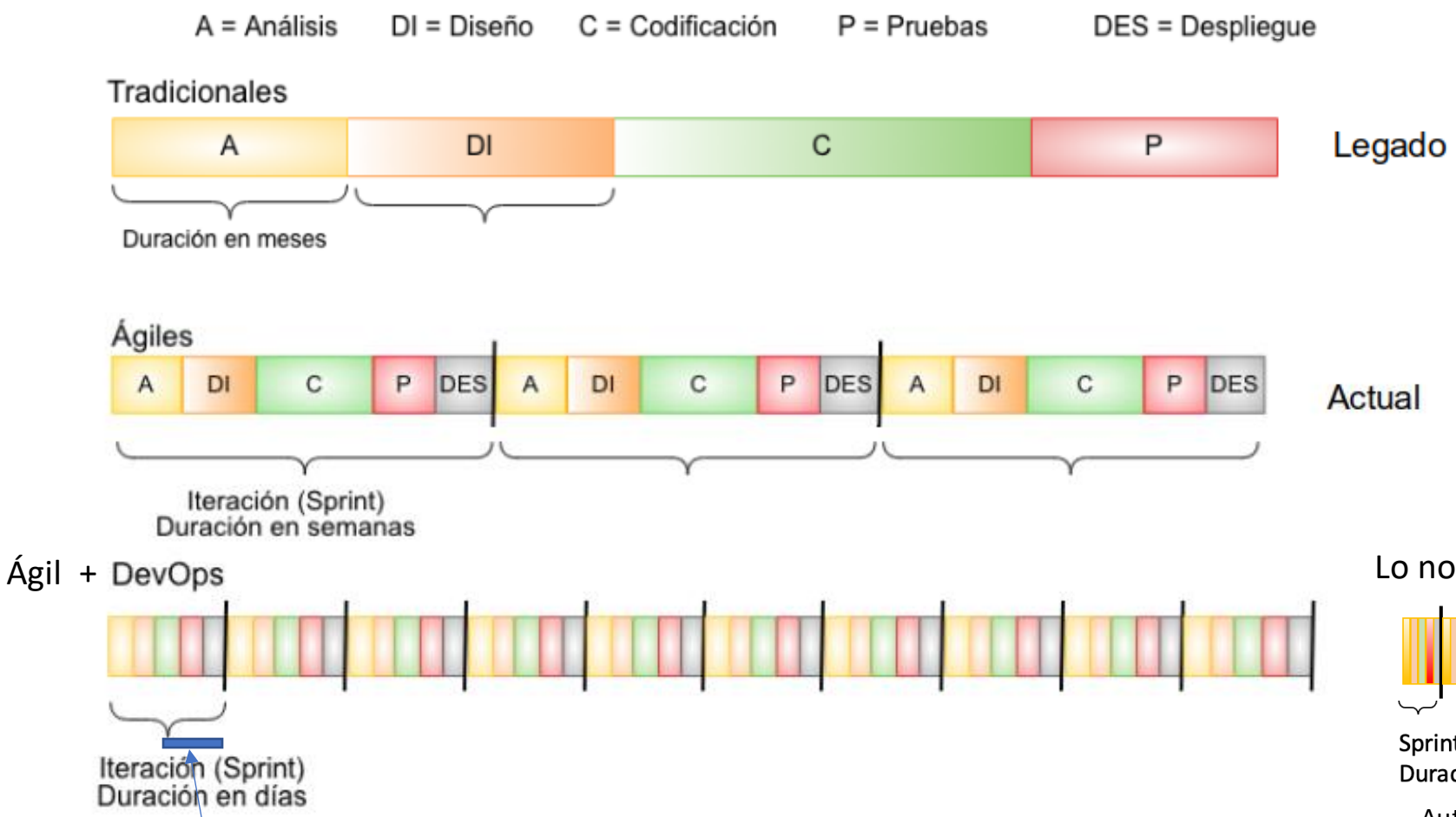
Conceptos

Un buen equipo de desarrolladores es la clave del éxito en un proyecto

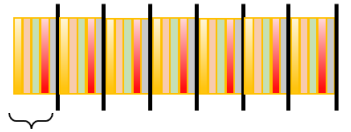


Conceptos

Ciclo de vida del desarrollo de software evoluciona



Lo novedoso: ágil + DevOps + IA



Sprint
Duración en días o horas.

Automatización de todas las etapas

Ingeniería de software aumentada con IA

Desarrollo de Software I

Concepto: Ciclo de vida del software hoy día

IDEACIÓN

Lluvia de ideas que resuelven un problema particular que enfrentan los usuarios.

REQUERIMIENTOS

Interacturar con los stakeholdes y usuarios para entender los procesos y recolectar los requerimientos del Proyecto.

DISEÑO

Crear la arquitectura de un sistema de software es decir sus componentes, sus interfaces y su forma de interactuar.

DESARROLLO

Crear el sistema de software usando una metodología de desarrollo y lenguajes de programación.

PRUEBAS

Evaluar la calidad del sistema para encontrar y corregir posibles errores.

DESPLIEGUE

Colocar el sistema en los servidores o ambiente donde va a funcionar.

MONITOREO

Conocer el estado del sistema y su infraestructura por si se requiere actualizaciones y soporte del software que esta en producción.

Metodologías de desarrollo de software

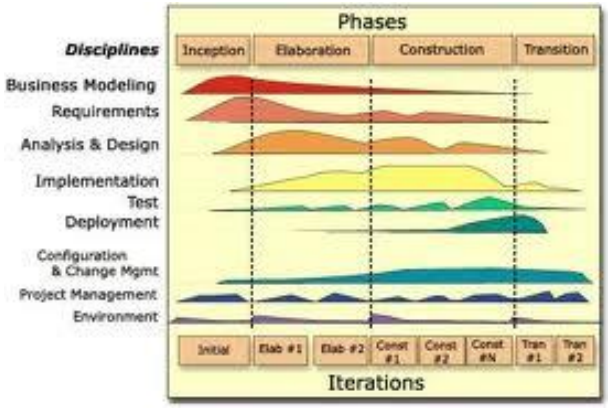
- Una metodología de desarrollo de software es un conjunto estructurado de prácticas, principios y procedimientos que guían y regulan el proceso de creación, implementación, despliegue y mantenimiento de sistemas de software.

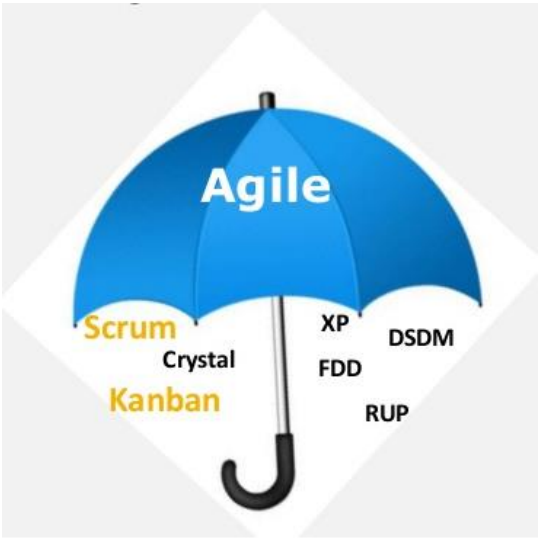
- Estas metodologías establecen un marco para organizar y gestionar las tareas involucradas en cada fase del ciclo de vida del software, desde la recolección de requisitos hasta el despliegue y soporte, con el fin de mejorar la calidad del producto final, optimizar los recursos disponibles y minimizar los riesgos asociados al desarrollo.

Conceptos

Metodologías de desarrollo de software

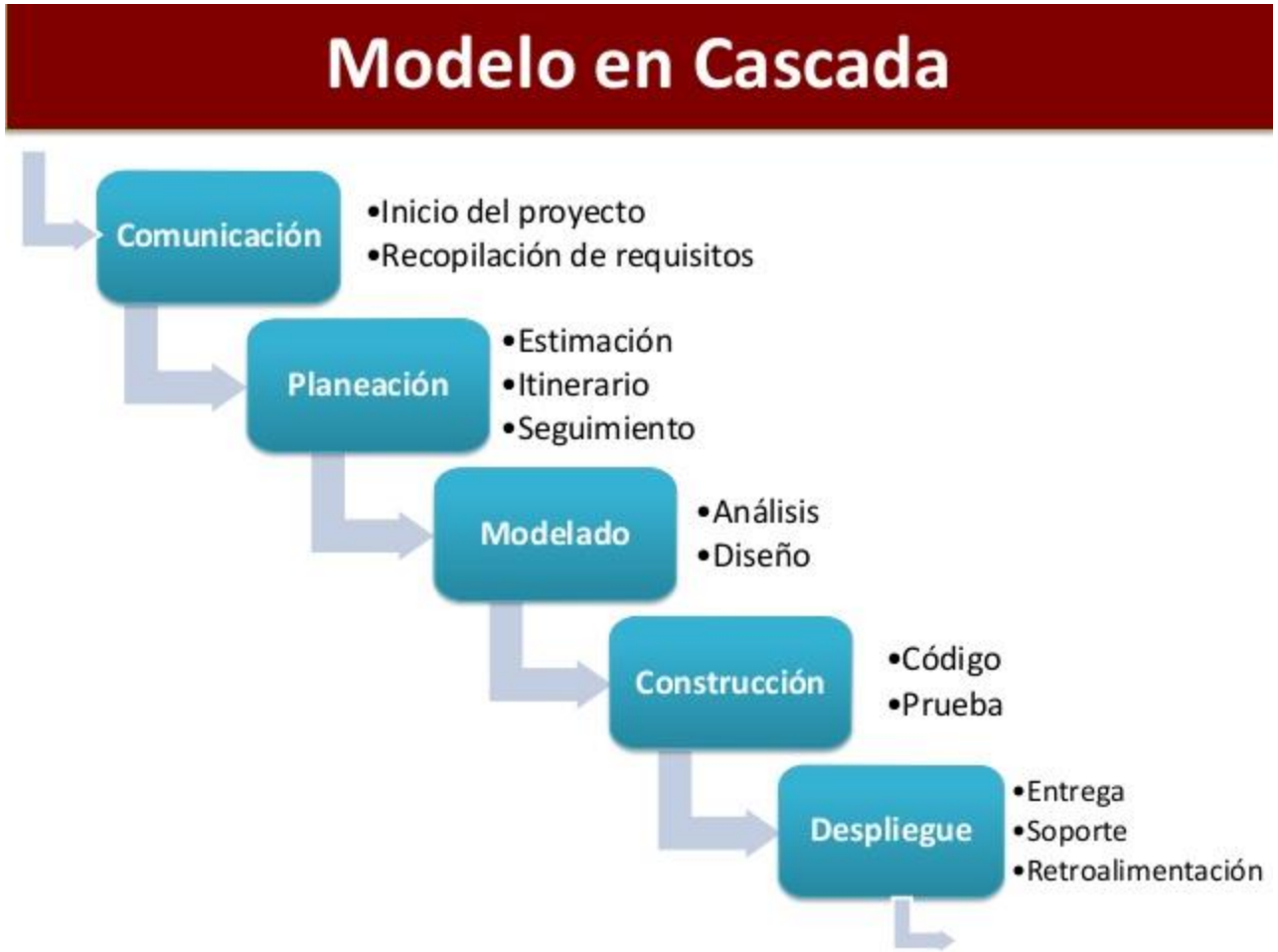
- Ejemplo: Metodologías Tradicionales RUP, MSF, ... Metodologías Ágiles SCRUM, XP, Kanban ...





Conceptos: El legado

Metodologías tradicionales



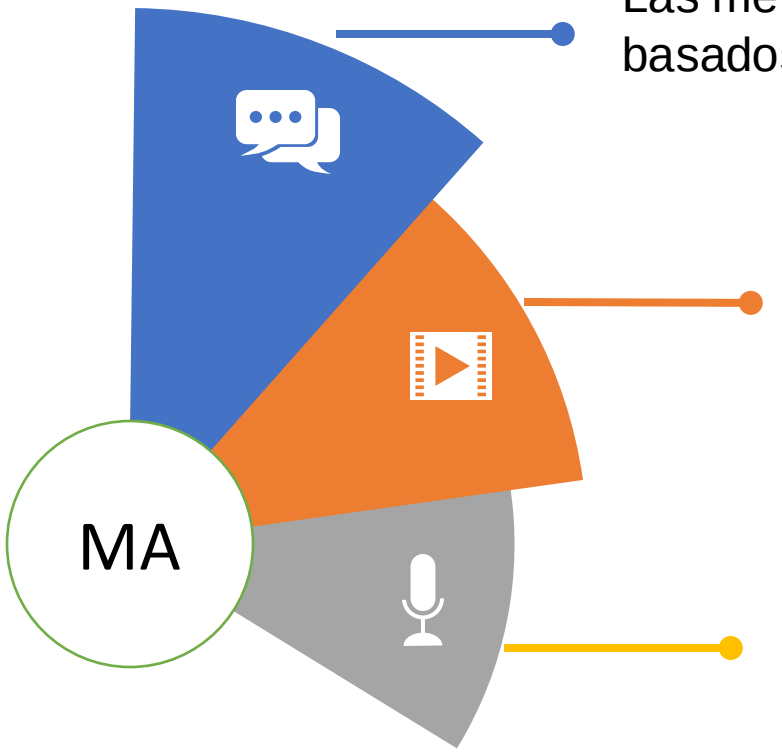
Desarrollo en cascada

Dependía de una planeación muy grande al inicio de los proyectos y una captura muy detalla de los requerimientos.

- Requiere procesos bien definidos
- Dificultades para incluir cambios
- Todo se entrega al final del proyecto

Conceptos: Metodologías actuales

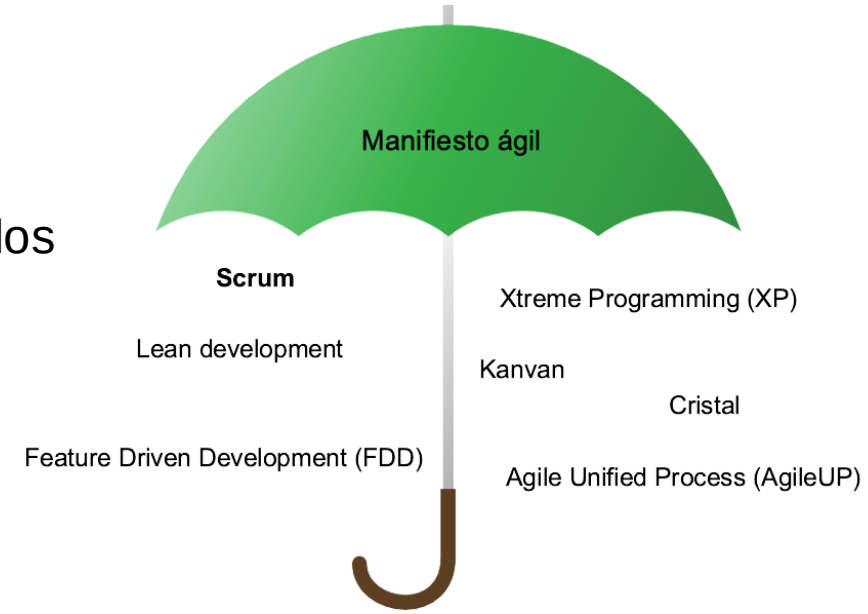
Metodologías ágiles



Las metodologías ágiles se refieren a métodos de ingeniería del software basados en el desarrollo iterativo e incremental.

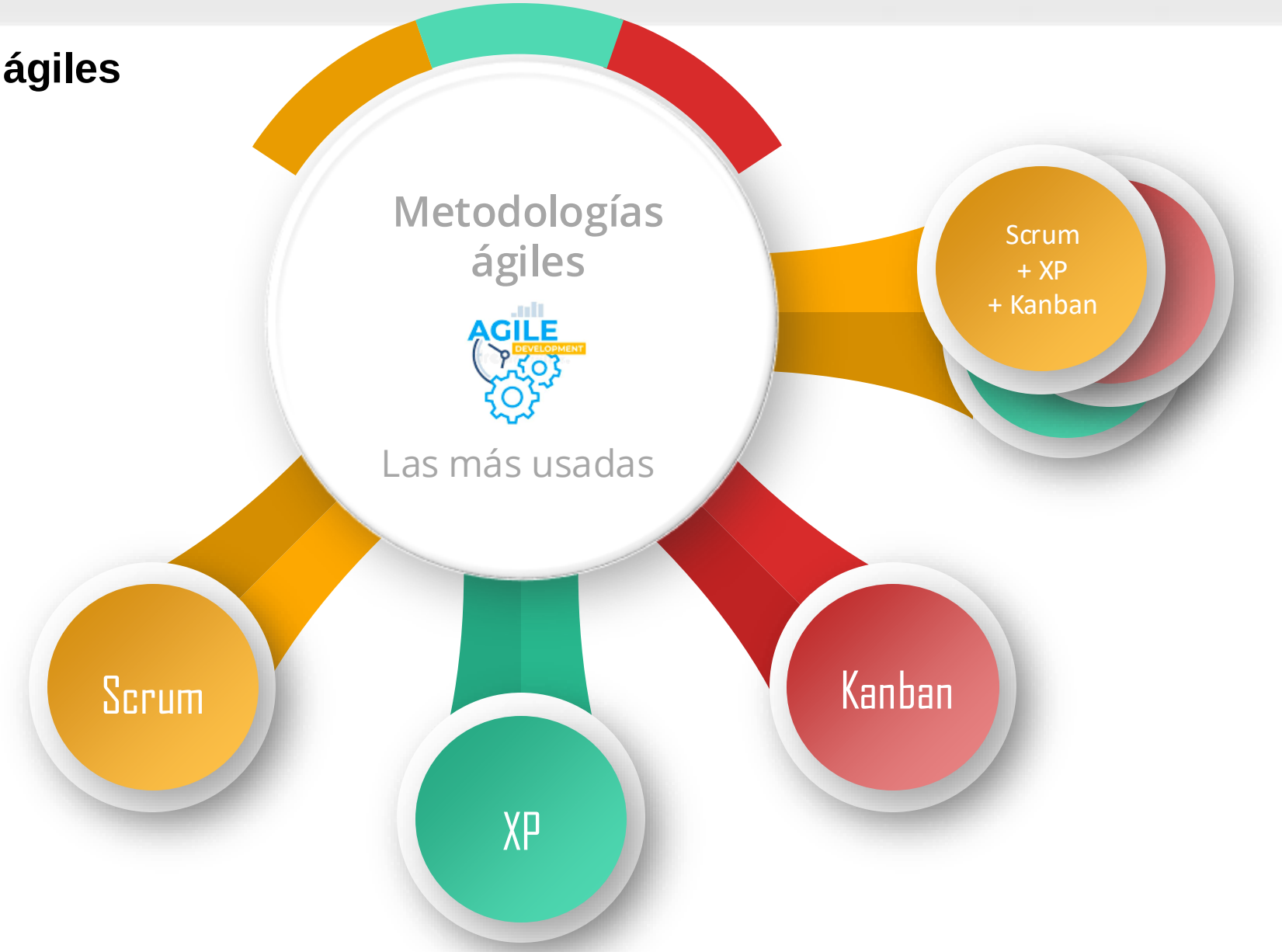
Las metodologías ágiles se basan en **prácticas ágiles** donde una práctica ágil es una actividad que se hace durante el proceso de desarrollo de software

Ejemplos



Conceptos

Metodologías ágiles

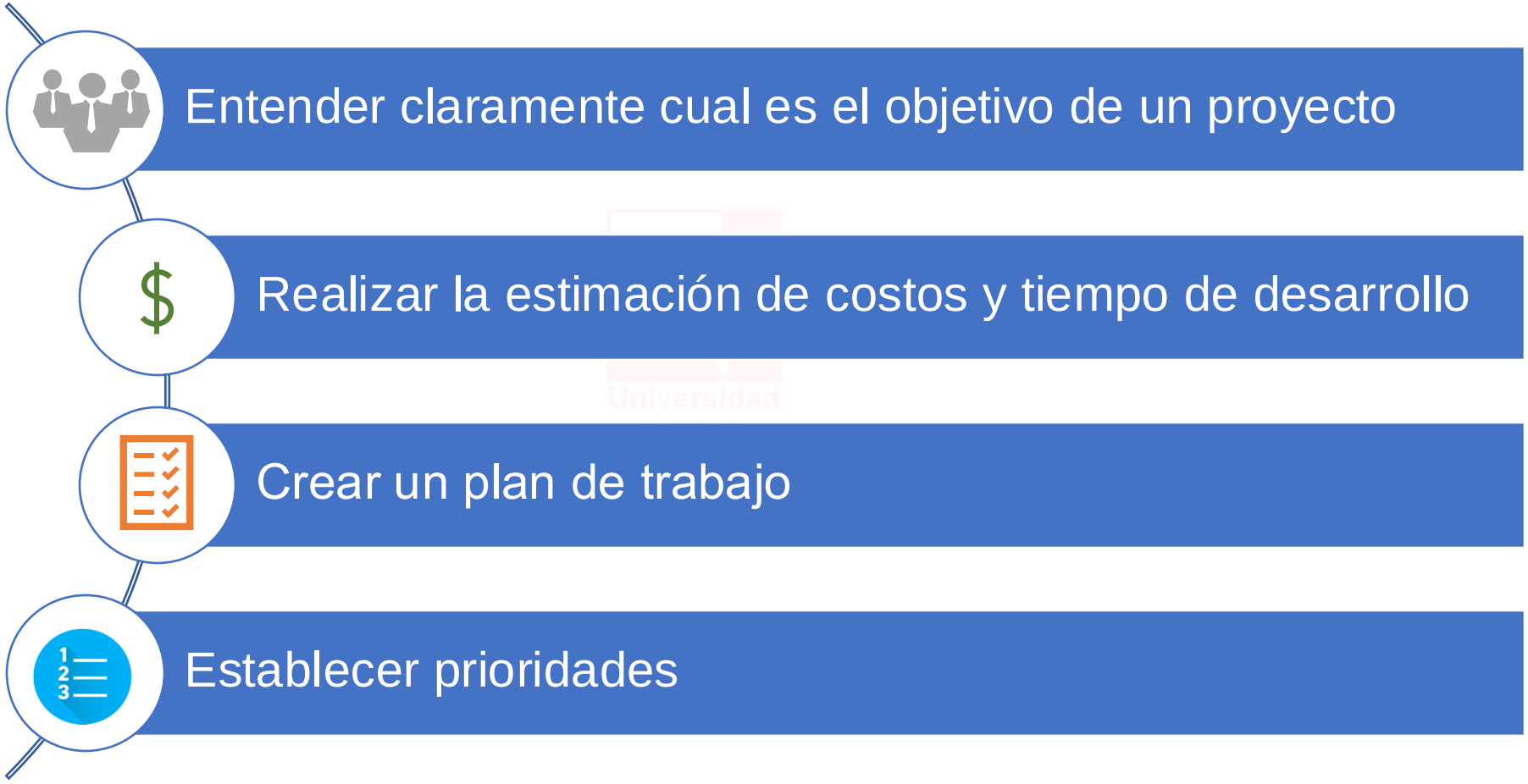


REQUERIMIENTOS DEL SOFTWARE

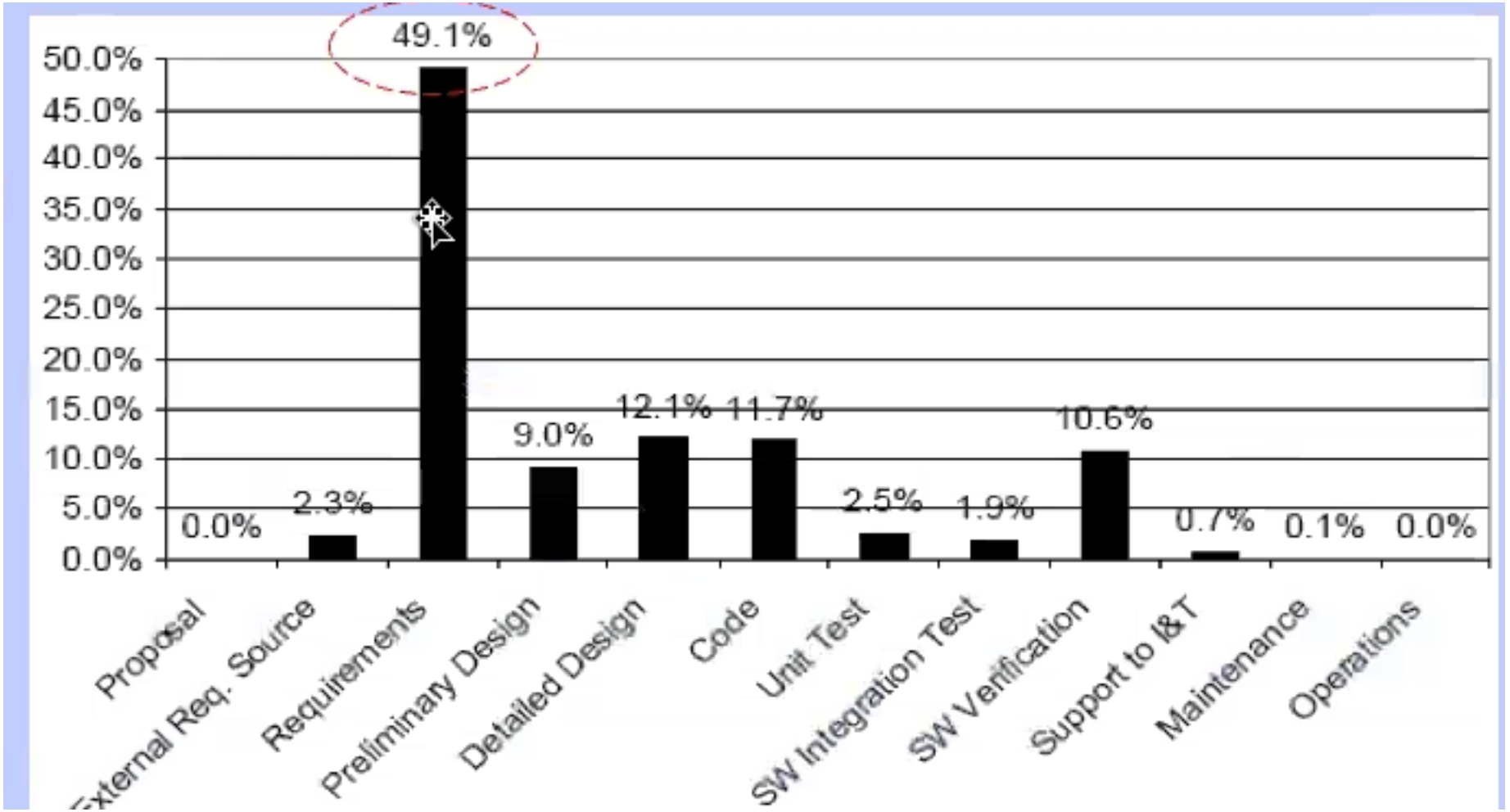
Un requisito o requerimiento es una declaración que identifica una característica o restricción operativa, funcional o de diseño de un producto o proceso, que es no ambigua, comprobable o medible, y necesaria para la aceptación del producto o proceso.

Los requerimientos especifican lo que desea el cliente y definen las acciones o actividades que debe hacer o cumplir un sistema de software.

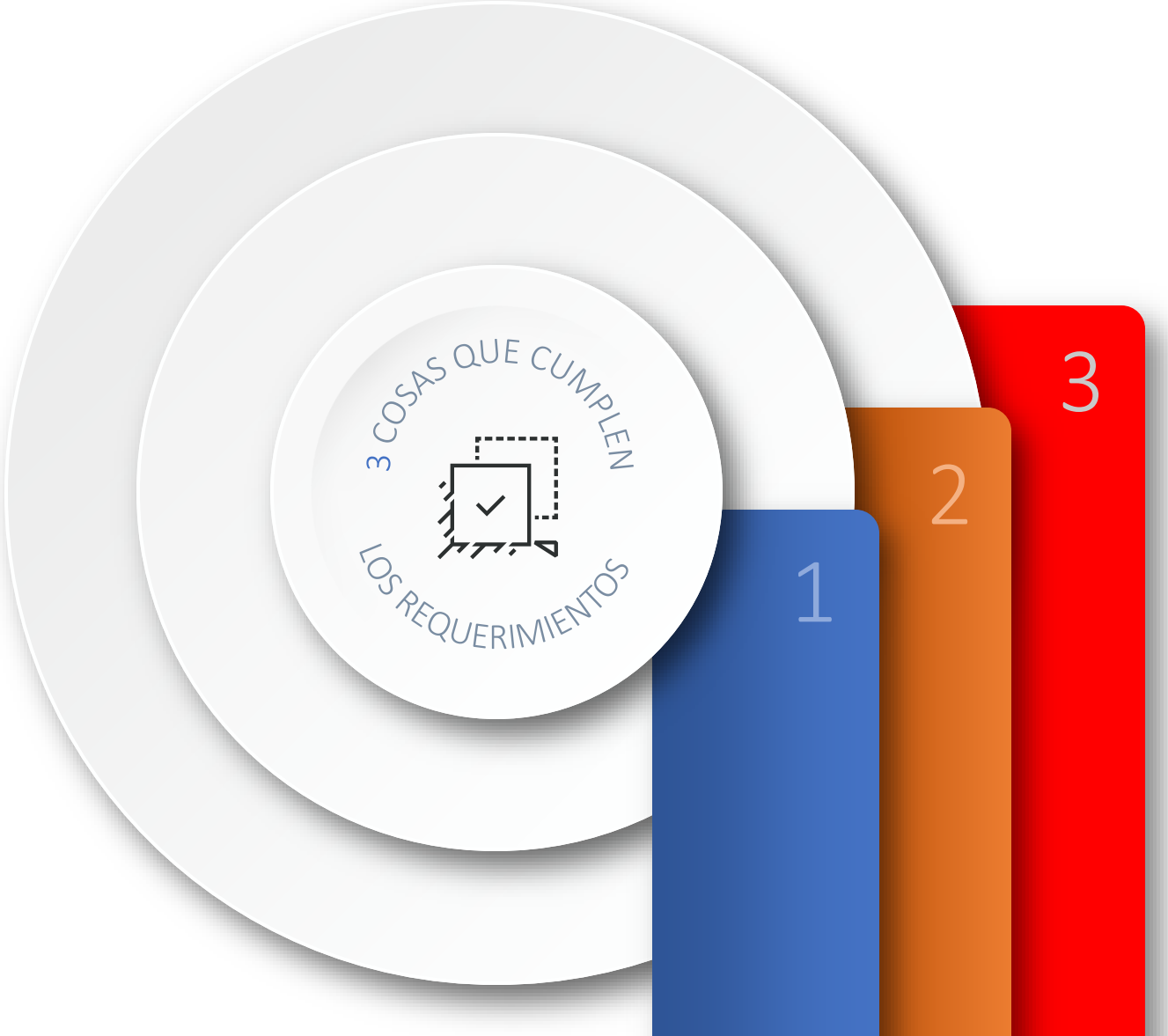
Definir adecuadamente los requerimientos permitirá:



¿De donde provienen los errores en el Desarrollo de software?



¿Qué es un requerimiento?



Resuelve una necesidad

Una condición o necesidad de un usuario para resolver un problema o alcanzar un objetivo.

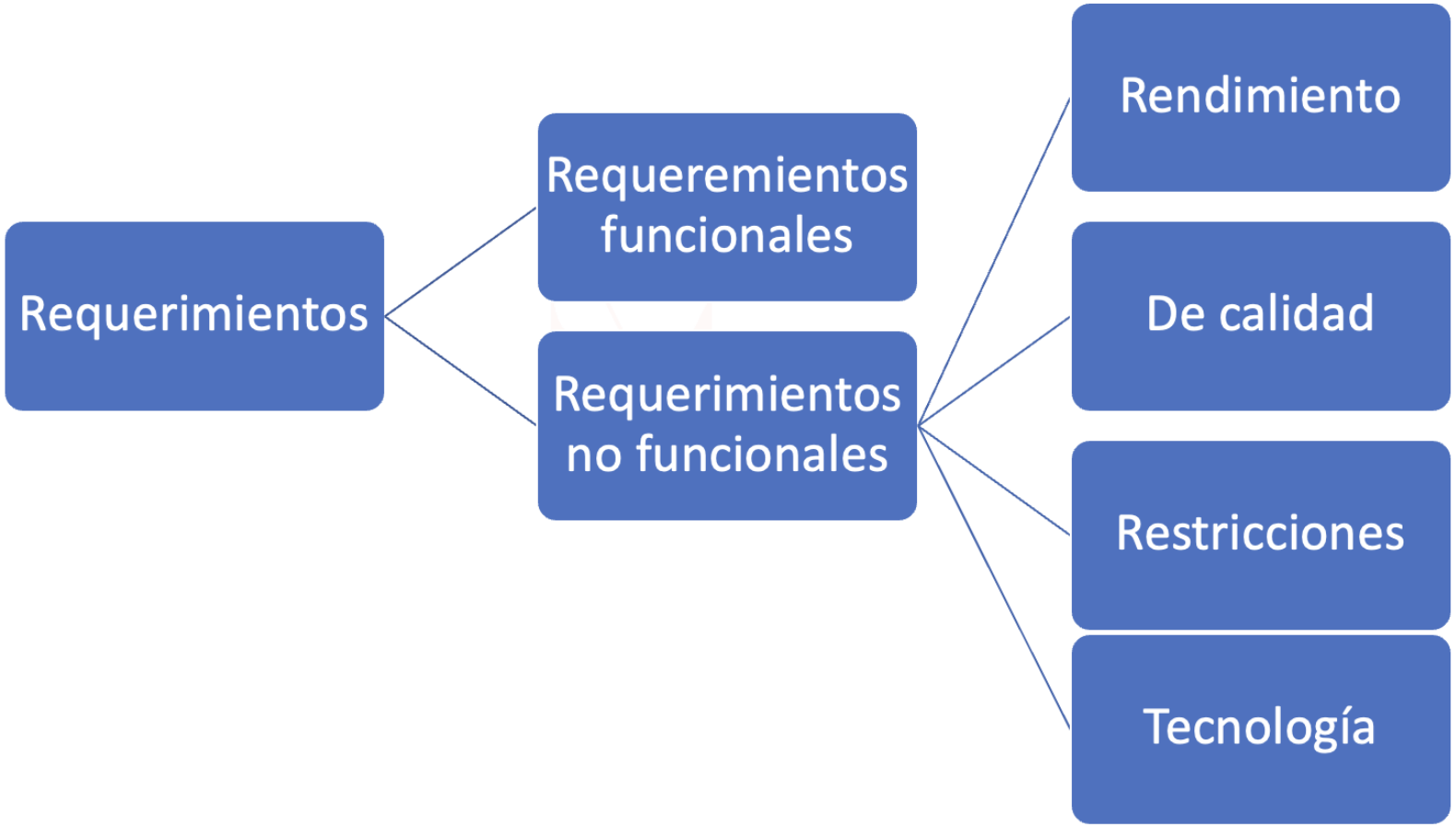
Satisface condiciones

Capacidad que debe estar presente en un sistema o componentes del sistema para satisfacer un contrato, un estándar, una especificación u otro documento formal.

Define lo que se necesita

Definen lo que el sistema debe hacer en cuanto a sus procesos, consultas, reportes, ...

Clasificación de los requerimientos.



Clasificación de los requerimientos.

Los requerimientos funcionales

Declaraciones de los servicios que debe proporcionar el sistema, la forma en que el sistema debe reaccionar a las entradas y la forma en que el sistema debe comportarse en situaciones particulares.

Requerimientos no funcionales

Limitaciones en los servicios o funciones ofrecidas por el sistema como de tiempo, limitaciones en el proceso de desarrollo, de rendimiento, de calidad, normas, tecnología, etc

Requerimientos funcionales.

Los requisitos o requerimientos funcionales son declaraciones de los servicios que prestará el sistema, en la forma en que reaccionará a determinados insumos.

Generalmente, los requerimientos funcionales describen el comportamiento del sistema en condiciones específicas.

En algunos casos, los requerimientos funcionales de los sistemas también establecen explícitamente lo que el sistema no debe hacer.

Requerimientos funcionales.

Los requerimientos.

se suelen especificar en lenguaje natural,

se expresan de forma individual y

se organizan de forma jerárquica (a distintos niveles de detalle),

a menudo, se numeran (para facilitar su gestión)

Requerimientos funcionales.

Requerimientos en las metodologías tradicionales

Ejemplos de requerimientos funcionales.

Descripción de requerimientos

El sistema debe permitir crear un usuario del sistema.

El sistema debe permitir registrar participantes a un evento.

El sistema debe permitir registrar la venta de un artículo.

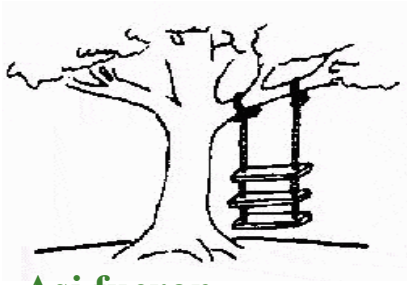
El sistema debe permitir anular una nota débito.

El sistema debe permitir a los usuarios contratistas poder registrar sus pagos de certificación al sistema de seguridad social junto al valor pagado por cada concepto como son: ARL, Pensión y EPS junto con el número de la planilla que lo soporta.

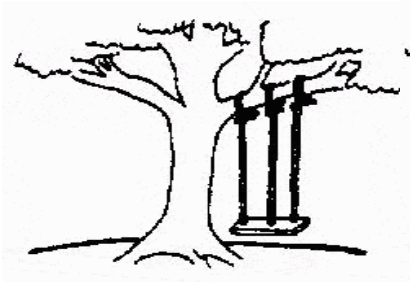
Imprecisión en los requerimientos.

Imprecisión en los requerimientos

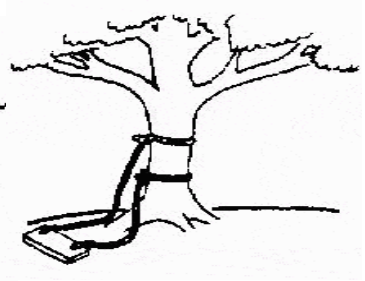
Problemas de comunicación



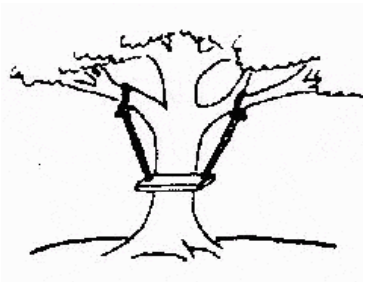
Asi fueron definidos los requerimientos



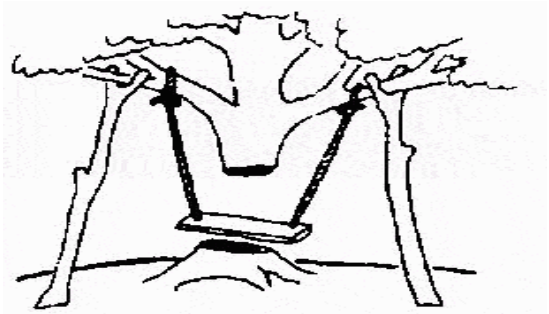
Asi lo entendieron los desarrolladores



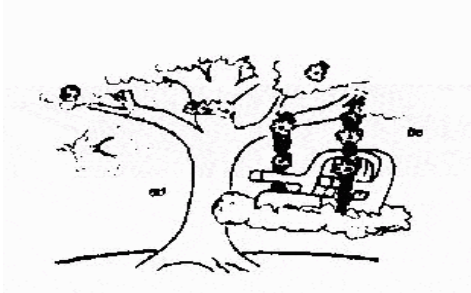
Asi fué resuelto el problema anteriormente



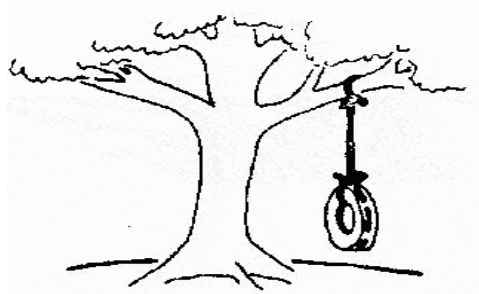
Asi fué resuelto el problema ahora



Este es el problema despues de la depuración



Así es como el problema es descrito por ventas



Esto es lo que el cliente realmente quería.

Imprecisión en los requerimientos.

Los problemas surgen cuando los requerimientos no son declarados con precisión.

Requerimientos ambiguos pueden interpretarse de diferentes maneras por los desarrolladores y usuarios.

Deben estar redactados de tal forma que sean comprensibles para usuarios sin conocimientos técnicos avanzados.

Características de los requerimientos.

Necesario: Lo que pida un requisito debe ser necesario para el producto.

Correcto: sí y solo sí, cada requisito especificado es un requisito que el software debe hacer.

No ambiguo: El texto debe ser claro, preciso y tener una única interpretación posible.

Conciso: Debe redactarse en un lenguaje comprensible por los clientes en lugar de uno de tipo técnico y especializado, aunque aún así debe referenciar los aspectos importantes

Consistente: Ningún requisito debe entrar en conflicto con otro requisito diferente. Asimismo, el lenguaje empleado entre los distintos requisitos debe ser consistente también.

Completo: Los requisitos deben contener en sí mismos toda la información necesaria, y no remitir a otras fuentes externas que los expliquen con más detalle.

Alcanzable: Un requisito debe ser un objetivo realista, posible de ser alcanzado en el tiempo y los recursos disponibles.

Verificable: Se debe poder verificar con absoluta certeza, si el requisito fue satisfecho o no. Esta verificación puede lograrse mediante inspección, análisis, demostración o testeo.

Requerimientos no adecuados.

El sistema debe ser fácil de utilizar.

El sistema proporcionará una respuesta rápida al usuario.

El sistema se recuperará automáticamente tras producirse un fallo.

Requerimientos adecuados.

Cuando haya hasta 100 usuarios accediendo simultáneamente al sistema, su tiempo de respuesta no será en ningún momento superior a 5 segundos.

Ante un fallo en el software del sistema, no se tardará más de 10 minutos en restaurar los datos del sistema (en un estado válido) y volver a poner en marcha el sistema.

Condiciones verificables

Recomendaciones para redactar requerimientos.

Usar un formato estándar y asegurar la adherencia al mismo para todos los requerimientos.

Utilizar el lenguaje de manera consistente.

Resaltar el texto para distinguir las partes clave del requerimiento.

Evite el uso de jerga informática.

Un requerimiento se especifica en dos pasos:

1. Realizar una **descripción** del requerimiento lo más precisa posible
2. Realizar la **especificación de los detalles** del requerimiento (requerida al momento de implementar)

Formato para registrar detalles de un requerimiento.

- Detalles de los requerimientos que le sirven al desarrollador para entender con más precisión lo que hay que programar.

Descripción	El sistema debe permitir registrar participantes a un evento
Identificador	**Rq-02**
Tipo de Requerimiento	**Tipo de requerimiento:** Funcional
Datos de Entrada	**Nombre, cédula, edad, género, nivel de formación, correo electrónico, evento**
Proceso	Dado los datos de entrada el sistema debe permitir almacenar cada uno de los datos ingresados y enlazar a un participante con el evento al cual desea asistir.
Datos de salida	**Mensaje: "El participante se registró con éxito"**
Resultados esperados	El sistema tendrá un nuevo participante en uno de sus eventos en caso de que el proceso de registro sea exitoso.
Origen	**Necesidades del cliente.**
Dirigido a	**Operadores.**
Prioridad	**5**
Requerimientos asociados	**Rq-01**

Ejercicio

El sistema debe permitir crear un usuario del sistema.

Elaborar la descripción de un requerimiento funcional para el proceso de registro de un curso en el sistema de la universidad.

Proceso para especificar requerimientos : Sirve para determinar los requerimientos

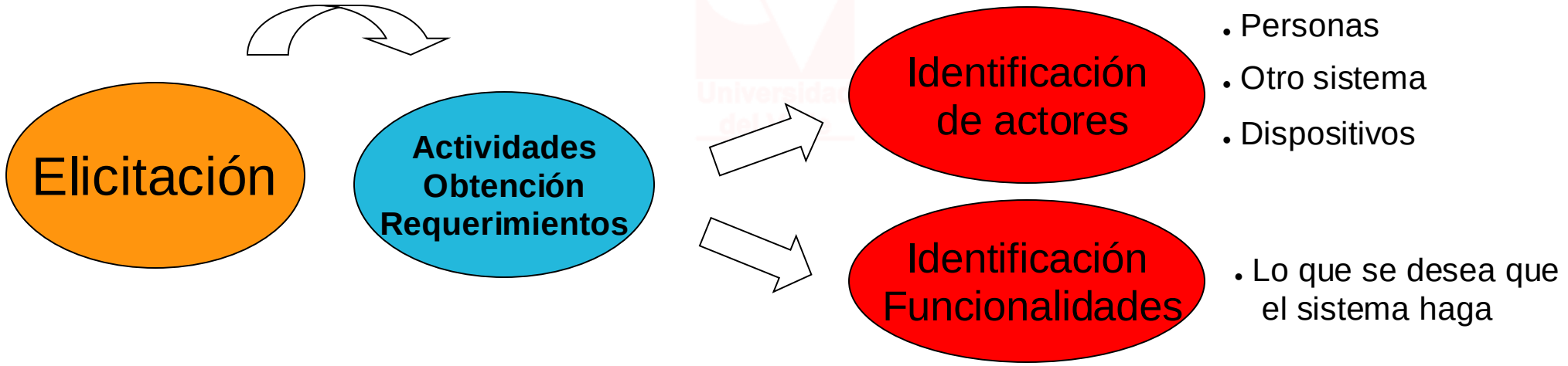
Elicitación: Es el proceso de recopilar y descubrir los requerimientos de un sistema de software a través de la comunicación directa con los interesados, como usuarios finales, clientes y otros actores clave. (Identificación de actores y funcionalidades)

Análisis de requerimientos: Es el proceso de identificar y entender las necesidades y expectativas de los usuarios y otras partes interesadas para un sistema de software.

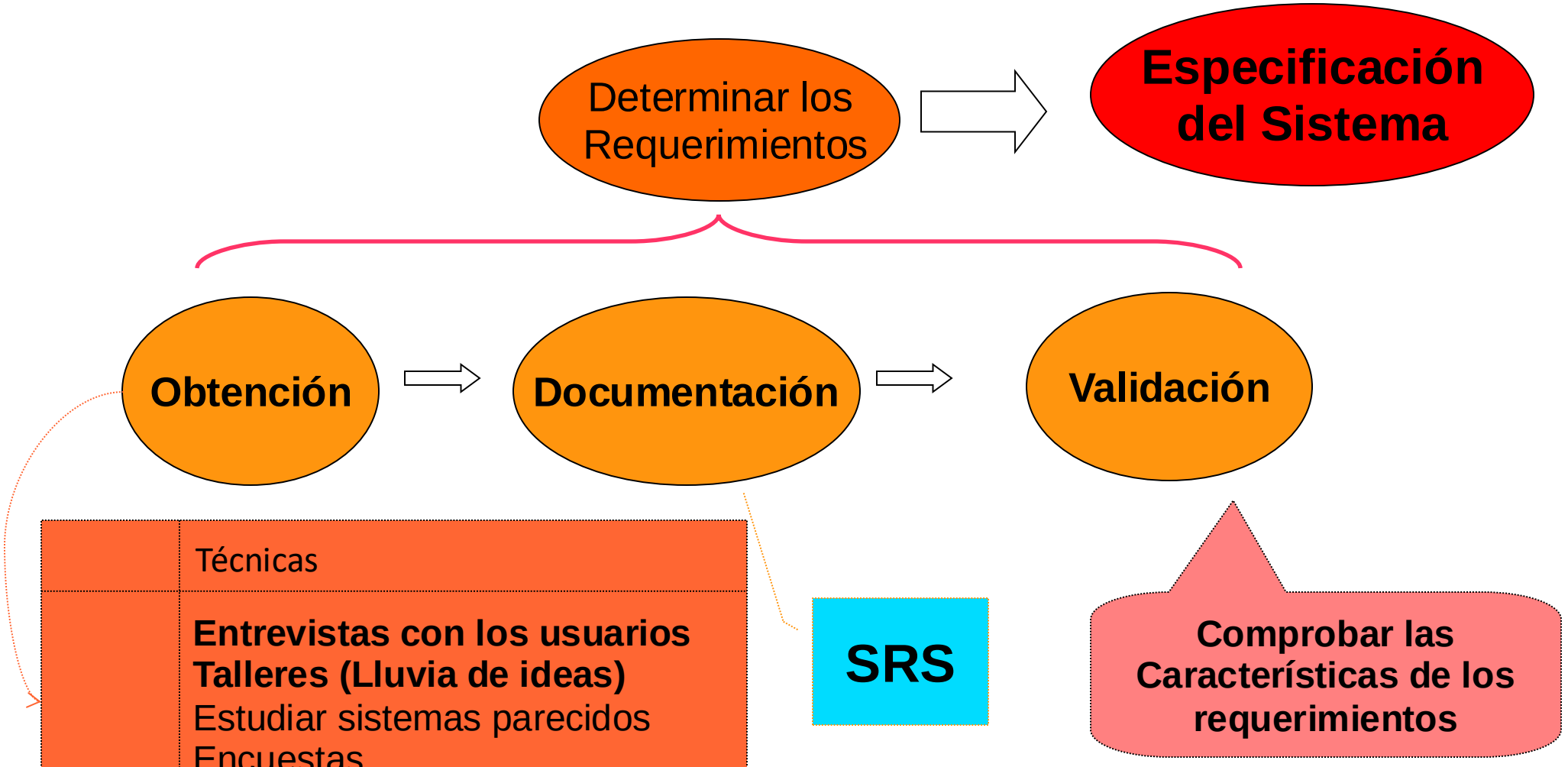
Especificación: Definir qué funciones y características debe tener el software, así como sus restricciones y condiciones de operación para realizar del documento de especificación de requerimientos.

Validación: Validación mediante pruebas para asegurar que los requerimientos fueron entendidos con el fin de garantizar que el producto final cumpla con los objetivos del proyecto y satisfaga las necesidades del cliente.

Proceso para especificar requerimientos



Proceso para especificar requerimientos



Requerimientos no funcionales

Técnicas para obtener requerimientos

Entrevistas

Las entrevistas de los usuarios y las partes interesadas son importantes para crear un software adecuado.

La entrevista es un diálogo formal o informal con personas, donde se busca respuesta a un conjunto de preguntas planeadas.

- Identificar el propósito de la entrevista : preguntas que permitan entender con claridad cada cosa que el cliente desea
- Identificar posibles entrevistados : funcionarios de diferentes niveles
- Estudiar el problema planteado
- Familiarizarse con el vocabulario del negocio
- Tomar apuntes o grabar la reunión.

Requerimientos no funcionales

Técnicas para obtener requerimientos

Entrevistas con los usuarios:

Recomendaciones: Evitar

Criticar la información emitida por la otra persona	Lenguaje inadecuado (muy técnico, muy formal o informal)
Completar las frases del otro o interrumpir su discurso	Demostrar la falta de preparación en el tema a tratar
Ser arrogante, o dar la impresión de que Usted sabe más que el otro	Corregir el otro, ya sea en alguna información o en su forma de expresarse
No manifestar interés en la información recibida, en el otro o en su problema	La falta de cortesía, amabilidad, contacto visual o puntualidad
Dar la solución antes de escuchar el problema	Lugar, tiempo o duración inadecuada
Dar un paso en falso (broma o comentario inapropiado)	Presentación personal inadecuada (formal / informal)

Requerimientos no funcionales

Técnicas para obtener requerimientos

Lluvia de ideas: Reunión donde se proponen ideas para solucionar un problema.

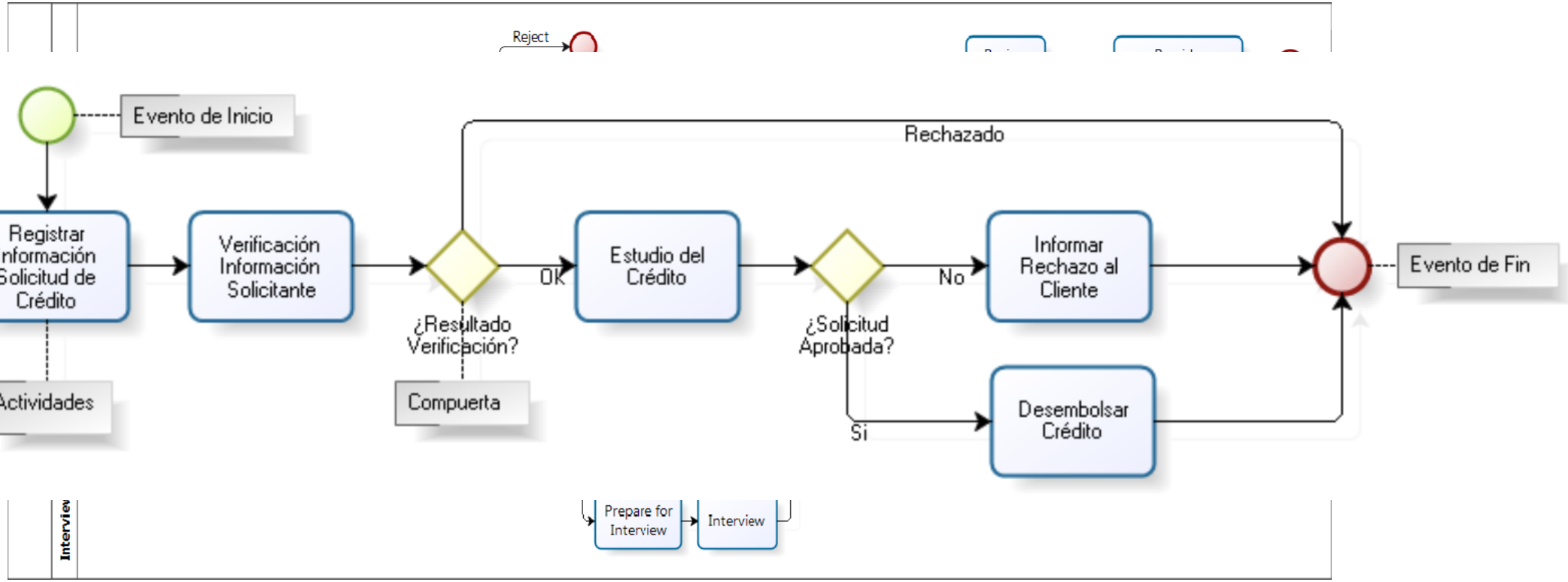
Por lo general, la lluvia de ideas se utiliza para identificar posibles soluciones a los problemas, para aplicaciones nuevas donde hay pocos antecedentes.

- Tener un plan : dividir el problema en partes pequeñas. Ej: hacer diagramas en papel o tableros ayuda a aclarar cosas y capturar colaboración.
- Proponer varias alternativas (ir preparado).
- Combinar ideas.
- Tenga un moderador y limite el tiempo de discusión de cada parte.
- Tomar apuntes o grabar la reunión.

Requerimientos no funcionales

Técnicas para obtener requerimientos

BPM: Business Process Model. **BPMN** Business Process Model Notation



Requerimientos no funcionales

Técnicas para validación de requerimientos

1. **Validación de expertos**

Personas con experiencia revisan los requerimientos y aprueban o rechazan el requerimiento

2. **Prototipado de interfaz de usuario**

El prototipado de interfaz de usuario es una técnica de representación aproximada de la interfaz de usuario

Los dos tipos principales de prototipos de interfaz de usuario son:

- **Desechables**: se utilizan sólo para la validación de los requisitos y posteriormente se desechan. Pueden ser prototipos en papel o en software.
- **Evolutivos:** una vez utilizados para la validación de los requisitos, se mejora su calidad y se convierten progresivamente en el producto final.

3. **Recorrido de BPM**: Hacer un BPM que muestre todo el proceso, algoritmo de alto nivel

Requerimientos no funcionales en las metodologías tradicionales

Requerimientos no funcionales

Especifican "qué tan bien" y "como" debe comportarse un sistema

Imponen **restricciones** que típicamente limitan los requerimientos funcionales

También conocidos como "requisitos técnicos", "atributos de calidad "o" requisitos de calidad de servicio "

Tipos de requerimientos no funcionales

Eficiencia:

- Toda funcionalidad del sistema y transacción de negocio debe responder al usuario en menos de 5 segundos.
- El sistema debe ser capaz de operar adecuadamente con hasta 100 usuarios con sesiones concurrentes.

Seguridad de lógica y de datos

- Los permisos de acceso al sistema podrán ser cambiados solamente por el administrador del sistema.
- Todas las comunicaciones externas entre servidores de datos, aplicación y cliente del sistema deben estar encriptadas utilizando el algoritmo RSA.

Usabilidad

- El sistema debe contar con manuales de usuario estructurados por cada módulo y funcionalidad.
- El sistema debe contar con un módulo de ayuda en línea.

Tecnología

- El sistema debe ser desarrollado usando el lenguaje Python 3.13.2
- El sistema debe funcionar en el sistema operativo Linux, Android, Windows y macOS

Requerimientos no funcionales

Son aquellos requerimientos que no se refieren directamente a las funciones específicas que entrega el sistema, sino a las propiedades emergentes de éste como la fiabilidad, la respuesta en el tiempo y la capacidad de almacenamiento.

De forma alternativa, definen las **restricciones** del sistema como la capacidad de los dispositivos de entrada/salida y la representación de datos que se utiliza en la interface del sistema.

Factores internos: Los requerimientos no funcionales surgen de la necesidad del usuario, debido a las restricciones en el presupuesto, a las *políticas* de la organización, a la necesidad de interoperabilidad con otros sistemas de software o hardware.

Factores externos: como los reglamentos de seguridad, las políticas del govierno, entre otros.

Preguntas ?

Desarrollo de Software I

Ideas para recordar de la clase de hoy

1

Presentación de curso

Características del curso

2

Qué es software

Definiciones

3

¿Qúe es ingeniería de software?

Aplicar conceptos, métodos y herramientas para crear aplicaciones de software.

4

Diferencias

Escribir programas
vs
Desarrollar software

5

Ciclo de vida

Proceso de creación de software

6

Metodologías

Proceso o marco estructurado para el desarrollo de software

7

REQUERIMIENTOS

...

Funcionales y no funcionales

8

ELICITACIÓN

Técnicas para obtener requerimientos, ...

Requerimientos no funcionales

Técnicas para obtener requerimientos

Tarea: Próxima clase control de lectura.

1. Ver video: https://youtu.be/2KZKMY75cJM

2. Leer documento: **Requirements Gathering Techniques**
   https://www.linkedin.com/pulse/requirements-gathering-techniques-samgra-malik/

Gracias

Universidad del Valle