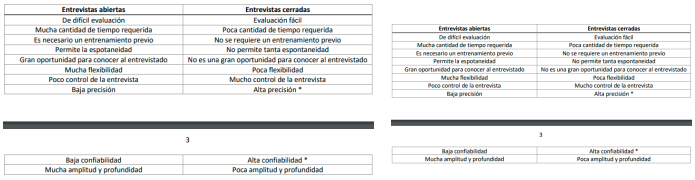
INGENIERIA DE SOFTWARE 1 UNLP- RESUMEN DE PROMOCIÓN

Requerimiento	característica del sistema o una descripción de algo que el sistema es capaz de hacer con el objetivo de satisfacer el propósito del mismo
Impacto de los errores en la etapa de requerimientos	El software resultante puede no satisfacer a los usuarios Las interpretaciones múltiples de los requerimientos pueden causar desacuerdos entre clientes y desarrolladores Gastar tiempo y dinero construyendo un sistema erróneo
Elicitación de requerimientos	Proceso mediante el cual se adquiere todo el conocimiento relevante necesario para producir un modelo de los requerimientos de un dominio de problema
Fuentes de requerimientos	Stakeholders Documentación Especificación de sistemas similares
Stakeholder	Personas o grupos de personas que se verán afectados por el sistema de manera directa o indirecta
Puntos de vista	Interactuadores: Representa a las personas u otros sistemas que interactúan de forma directa con el sistema. Influyen en los requerimientos del sistema Inidrecto: Representa a los stakeholders que no usan el sistema ellos mismos pero que influyen en los requerimientos del sistema Del Dominio: Representa las características y restricciones del dominios que que influyen en los requerimientos del sistema.
Características del Software	Es un elemento lógico Se desarrolla, no se fabrica

	No se desgasta
¿Por qué el software no sigue la curva clásica del envejecimiento?	El software no sigue la curva clásica del envejecimiento porque su deterioro no es físico. Sin embargo, puede "envejecer" debido a cambios en el entorno, las necesidades del usuario o la acumulación de deuda técnica, lo que exige mantenimiento y actualizaciones constantes, es decir, **el problema no está en el tiempo de operación, sino en los cambios.**
¿Cuáles son los objetivos de la elicitación de requerimientos?	• Conocer el dominio del problema para así poder comunicarse con clientes y usuarios y entender sus necesidades • Ver como se están manejando actualmente, con que sistema ya sea manual o informatizado • Entender las necesidades implícitas y explícitas de los clientes y lo usuarios (saber qué esperan ellos del sistema)
Requerimiento	Característica o descripción de algo que el sistema es capaz de hacer con el objetivo de satisfacer el propósito del mismo
Diferencia entre requerimiento funcional y no funcional	La diferencia entre un requerimiento funcional y un requerimiento no funcional radica en lo que describen del sistema. Mientras que los requerimientos funcionales describen lo que debe y no debe hacer el sistema y cómo debe comportarse ante determinados estímulos, los requerimientos no funcionales describen restricciones sobre el sistema que limitan nuestras elecciones en la construcción de una solución al problema.
Ejemplo de requerimiento funcional y no funcional	Un ejemplar de requerimiento funcional sería que desde el moodle de la facultad pueda acceder a la funcionalidad de calendario y que no pueda acceder al plan de estudios Un ejemplo de requerimiento no funcional sería que el oodle tiene que andar las 24 horas los 7 días de la semana, que las contraseñas de los alumnos se guarden de manera encriptada o que la página respete los colores del logo

Propiedades de un requerimiento	• Necesario: Su omisión provoca una deficiencia en el sistema • Conciso: Fácil de entender y leer • Completo: No necesita ampliarse • Consistente: No se contradice con otro • No ambiguo: Tiene una sola implementación • Verificable: Puede testearse a través de inspecciones, pruebas
Validación de requerimientos	Es el proceso de certificar la correctitud del modelo de requerimientos contra las intenciones del usuario. Trata de mostrar que los requerimientos definidos son los que estipula el sistema
Diferencia entre validar y verificar	La validación se hace al final del desarrollo en donde se evaluar al software y se asegura que cumple con los requisitos mientras que la verificación es asegurar se que e software cumple con los requerimientos de forma correcta
Métodos de elicitación discretos	Son los que menos perturban al sistema pero se consideran insuficientes par recopilar información cuándo se utilizan por si solos • Muestreo de la documentación, los formularios y los datos existentes: Me permiten conocer el historial que origino el proyecto • Investigación y visitas al sitio: Investigar el dominio y consultar otras organizaciones • Observación del ambiente de trabajo
Métodos de elicitación interactivos	Se basan en hablar con las persona, escuchar y comprender lo que están diciendo • Cuestionarios: permiten analizar y recabar información y opiniones y son útiles cuando las personas están dispersas geográficamente, cuando son muchos los involucrados o cuando queremos obtener opiniones generales.

	• Entrevistas: se recolecta información de las personas mediante la ínteraccion cara a cara. Básicamente es un conversación con un propósito especifico que se basa en un formato de preguntas y respuestas general • Planeación conjunta de requerimientos (JRP): Proceso mediante el cual se conducen reuniones de grupo altamente estructuradas con el propósito de analizar problemas y definir requerimientos • Brainstorming: Tecnica en la cual se alienta a los participantes apara ofrecer tantas ideas como sea posible en un corto tiempo, sin ningún análisis
¿Cuándo debo aplicar cuestionarios?	son útiles cuando las personas están dispersas geográficamente, cuando son muchos los involucrados o cuando queremos obtener opiniones generales.
¿Cuándo debo aplicar entrevistas?	• Cuando necesito información detallada y especifica • Cuando los requerimientos no estan claros o son ambiguos • Cuando el grupo de stakeholders es reducido
4 ventajas y 4 desventajas de los cuestionarios y de las entrevistas	CUESTIONARIOS: Ventajas: • Útil para un gran numero de personas • Es económico y fácil de estructurar • Obtiene sentimientos generalizados y respuestas cuantificadas • Lleva poco tiempo Desventajas • No analizo el lenguaje corporal • Preguntas tienden a ser rígidas • No todos contestan • Difíciles de preparar ENTREVISTAS

	Ventajas • El entrevistador se siente incluido • Hay posibilidad de retroalimentación • Puedo adaptar las preguntas al entrevistado • Puedo obtener la información no verbal o corporal Desventajas • Son costosas • Requieren de tiempo y recursos humanos • No son aplicables a distancia • Dependen de la habilidad del entrevistador
Ventajas y desventajas de las preguntas cerradas y abiertas en los cuenstionarios	 	Características	ABIERTAS	CERRADAS
---	---	---
Velocidad de conclusión	Lenta	Rápida
Naturaleza exploratoria	Alta	Poca
Amplitud y profundidad	Alta	Poca
Facilidad de preparación	Fácil	Difícil
Facilidad de análisis	Difícil	Fácil | | Características	ABIERTAS	CERRADAS
---	---	---
Velocidad de conclusión	Lenta	Rápida
Naturaleza exploratoria	Alta	Poca
Amplitud y profundidad	Alta	Poca
Facilidad de preparación	Fácil	Difícil
Facilidad de análisis	Difícil	Fácil	
Ventajas y desventajas de las preguntas cerradas y abiertas en las entrevistas	 	Entrevistas abiertas	Entrevistas cerradas
De difícil evaluación	Evaluación fácil
Mucha cantidad de tiempo requerida	Poca cantidad de tiempo requerida
Es necesario un entrenamiento previo	No se requiere un entrenamiento previo
Permite la espotaneidad	No permite tanta espontaneidad
Gran oportunidad para conocer al entrevistado	No es una gran oportunidad para conocer al entrevistado
Mucha flexibilidad	Poca flexibilidad
Poco control de la entrevista	Mucho control de la entrevista
Baja precisión	Alta precisión *	 3 	Baja confiabilidad	Alta confiabilidad *
Mucha amplitud y profundidad	Poca amplitud y profundidad	 	Entrevistas abiertas	Entrevistas cerradas
De difícil evaluación	Evaluación fácil
Mucha cantidad de tiempo requerida	Poca cantidad de tiempo requerida
Es necesario un entrenamiento previo	No se requiere un entrenamiento previo
Permite la espotaneidad	No permite tanta espontaneidad
Gran oportunidad para conocer al entrevistado	No es una gran oportunidad para conocer al entrevistado
Mucha flexibilidad	Poca flexibilidad
Poco control de la entrevista	Mucho control de la entrevista
Baja precisión	Alta precisión *	 3 	Baja confiabilidad	Alta confiabilidad *
Mucha amplitud y profundidad	Poca amplitud y profundidad	
Ventajas y desventajas de JRP	VENTAJAS -ahorro de tiempo -involucración activa de los usuarios y la gerencia -reducción del tiempo en el que se toman los requerimientos DESVENTAJAS -dificultad para organizar los horarios de todos los involucrados -es complejo planear las sesiones -es complejo encontrar un grupo de participantes integrados y un organizador
Modelo de software	Es una representación abstracta de un proceso de software que presenta una visión de ese proceso.
¿Que es un SRC?	Es un documento de definición de requerimientos, en donde se lista todo lo que el cliente espera que se haga en el sistema propuesto

Aspectos básicos de un SRC	• Funcionalidad: qué debe hacer el software • Interfaces externas: como interactúa el software con el medio externo • Rendimiento: velocidad, disponibilidad, tiempo de respuesta • Atributos: portabilidad, seguridad, mantenibilidad, eficiencia • Restricciones de diseño: estándares requeridos, lenguaje, limite de recursos
Estudio de viabilidad	Etapa en la cual se analiza si estamos seguros de realizar el software. Esta etapa es importante para los sistemas nuevos
¿Es suficiente validar después del desarrollo del software?	No, porque mientras mas tarde se detecta un problema, mas cuesta corregirlo creando una bola de nieve de defectos
¿Contra que se validan los requerimientos?	Se validan contra el usuario. Se alcanza una convicción por parte del cliente. Básicamente, si el cliente piensa que así está bien es lo que se hace
Técnicas de especificación de requerimientos	Estáticas: Describen el sistema a través de las entidades u objetos, sus atributos y relaciones con otros. No se describe como cambian con el tiempo. Es una descripción útil cuando el tiempo no es un factor mayor en la operación del sistema Dinamicas: Se considera al sistema en función de los cambios que ocurren a lo laro del tiempo. Se considera que el sistema esta en un estado particular hasta que un estimulo particular lo obliga a cambiar de estado
Ejemplos de técnicas de especificación de requerimientos dinámicas y estáticas	Estaticas: Referencia indirecta, Relaciones de recurrencia, Definición axiomática, Expresiones regulares, Abstracciones de datos, entre otras Dinámicas: Tablas de decisión, Diagramas de transición de estados, Tablas de transición de estados, Diagramas de persianas, Diagramas de transición extendidos, Redes de Petri, entre otras.

Historias de usuario	Son utilizadas en las metodologías ágiles y es una representación de un requisito de software escrito en una o dos frases utilizando el lenguaje común del usuario
Criterios que debe cumplir una historia de usuario	• Debe ser limitada o sea debe poder escribirse en una nota adhesiva pequeña • Permiten responder rápidamente a los requisitos cambiantes • Al momento de implementar las historias, los desarrolladores deben tener la posibilidad de discutirla con los clientes • Son una forma rápida de administrar los requisitos de los usuarios, sin tener que elaborar gran cantidad de documentos formales y sin requerir de mucho tiempo para administrarlos
Características de una Historia de Usuario	• Independientes unas de otras • Negociables • Deben ser valoradas por los clientes o usuarios • Estimables en tiempo • Pequeñas • Verificables
Ventajas de las Historias de Usuario	• Como son muy cortas, representan requisitos del modelo que pueden implementarse rápidamente en días o semanas • Necesitan poco mantenimiento • Mantienen una relación cercana con el cliente, ya que se escriben en conjunto con este • Permite dividir los proyectos en pequeñas entregas • Permite estimar fácilmente el esfuerzo de desarrollo • Es ideal para proyectos con requisitos volátiles o no muy claros

Desventajas de las Historias de Usuario	• Sin criterios de aceptación pueden quedar abiertas a distintas interpretaciones haciendo difícil utilizarlas como base para un contrato. • Se requiere un contacto permanente con el cliente durante el proyecto lo cual puede ser difícil o costoso. • Podría resultar difícil escalar a proyectos grandes. • Requiere desarrolladores muy competentes.
¿Qué es una épica?	Es un conjunto de historias de usuario que se agrupan por algún denominador común. Un proyecto puede tener varias épicas
Casos de Uso	Proceso de modelado de las “funcionalidades” del sistema en término de los eventos que interactúan entre los usuarios y el sistema. Es aplicable a cualquier metodología de desarrollo
Ventajas de los Casos de Uso	• Herramienta para capturar requerimientos funcionales. • Descompone el alcance del sistema en piezas más manejables. • Medio de comunicación con los usuarios. • Utiliza lenguaje común y fácil de entender por las partes. • Permite estimar el alcance del proyecto y el esfuerzo a realizar. • Define una línea base para la definición de los planes de prueba. • Define una línea base para toda la documentación del sistema. • Proporciona una herramienta para el seguimiento de los requisitos.
Componentes de un CU	Diagrama de Casos de Uso: Ilustra las interacciones entre el sistema y los actores. Escenarios (narración del CU): Descripción de la interacción entre el actor y el sistema para realizar la funcionalidad.

Relaciones entre caso de uso y actores	• Asociaciones. Es una relación entre un actor y un CU que interactúan entre sí. • Extensiones. Es la relación entre CU, dónde uno extiende (implementa) la funcionalidad que tiene otro CU. Un caso de uso puede tener muchas extensiones. Estos sólo son iniciados por otro CU, no por un actor. • Uso o inclusión. Entre CU. Reduce la redundancia entre dos o más CU al combinar los pasos comunes que hay entre ambos • Dependencia. Relación entre CU, dónde un caso de uso no puede utilizarse hasta que no termine otro. • Herencia. Relación entre actores, dónde un actor hereda las funcionalidades de uno o varios.
Características que debe cumplir un CU	• Un CU debe representar una funcionalidad concreta. • La descripción de los pasos en los escenarios debe contener más de un paso, para representar la interacción entre los componentes. • El uso de condicionales en el curso normal, es limitado a la invocación de excepciones, ya que este flujo representa la ejecución del caso sin alteraciones. • Las pre-condiciones no deben representarse en los cursos alternativos, ya que al ser una pre-condición no va a ocurrir. • Los “uses” deben ser accedidos por lo menos desde dos CU.
Maquinas de estado finito	Describen al sistema como un conjunto de estados dónde el sistema reacciona a ciertos eventos posibles (externos o internos). De un estado pasa a otro. Esta técnica tiene tanto una notación gráfica como una notación matemática
¿Cómo se pueden representar las maquinas de estado finito?	• Diagramas de persianas: cada estado está representado con una línea horizontal y las transiciones como flechas verticales entre estado y estado.

	• Diagrama de transición de estados:
Diagrama de Transición de Estados	Un **diagrama de transición de estados** es una herramienta gráfica que muestra cómo un sistema o componente cambia de un estado a otro en respuesta a eventos o condiciones. Es parte de la técnica de modelado de máquinas de estados, utilizada para representar sistemas dinámicos con un número finito de estados.
Características de un diagrama de transición de estados	• Se pueden alcanzar todos los estados • Se pueden salir de todos los estados • En cada estado, el sistema responde a todas las condiciones posibles (normales y anormales)
Redes de petri	Utilizadas para especificar sistemas de tiempo real en los que son necesarios representar aspectos de concurrencia. Los sistemas concurrentes se diseñan para permitir la ejecución simultánea de componentes de programación, llamadas tareas o procesos, en varios procesadores o intercalados en un solo procesador. Permite la ejecución simultanea de componentes y trabaja con la idea de proceso-tarea, que sería el programa en ejecución
Caractesiticas que debe cumplir una red de petri	• Las tareas concurrentes deben estar sincronizadas para permitir la comunicación entre ellas pueden operar a distintas velocidades • Deben prevenir la modificación de datos compartidos condiciones de bloqueo • Pueden realizarse varias tareas en paralelo, pero son ejecutados en un orden impredecible
¿Qué elementos tiene una red de petri?	Eventos, las acciones que quiero hacer. Los eventos se representan como transiciones (T) Estados son las condiciones del sistema las cuál me hacen cambiar. Se presentan como lugares o sitios (P) En el diagrama, los arcos nos van a indicar, a

	través de una flecha, la relación entre sitios y transiciones, y viceversa. A los lugares se les asignan tokens (representación de recursos) que se representan mediante un número o punto dentro del sitio. Esta asignación de tokens a lugares constituye la marcación. Luego de una marcación inicial, se puede simular la ejecución de la red. El número de tokens asignados a un sitio es ilimitado.
¿Cómo se habilitan las transiciones en las redes de petri?	La ejecución de una Red de Petri se realiza disparando transiciones habilitadas. Una transición está habilitada cuándo cada lugar de entrada tiene al menos tantos tokens como arcos hacia la transición.
Patrones en las redes de petri	Paralelismo - Sincronización - Exclusión mutua - Inanición
Tablas de decisión	Son herramientas que permiten presentar de forma concisa las reglas lógicas que hay que utilizar para decidir acciones a ejecutar en función de las condiciones y la lógica de decisión de un problema específico. Es mejor elegirlas cuándo hay mucha condición lógica en el enunciado.
Analisis estructurado	Forma parte de las técnicas de especificación de requerimientos, pero es una actividad de construcción de modelos mucho mas amplia que uns técnica. Mediante una notación se crean modelos que representan el contenido y flujo de de información (datos y control)
Objetivos del análisis estructurado	• Describir lo que requiere el cliente (requerimientos) • Establecer una base para la creación de un diseño de software • Definir un conjunto de requisitos que se pueda validar una vez que se construye el software (construir pruebas)
¿Cómo es el modelamiento del análisis estructurado?	Es una amalgama de métodos Para modelar el comportamiento del sistema podemos encontrar las máquinas de estado finito

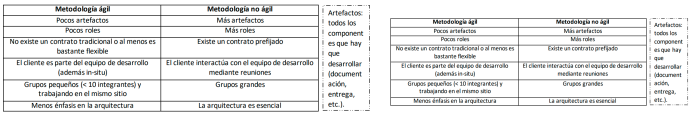
	Para modelar las funciones del sistema el diagrama de flujo de datos Para modelar los datos del sistema diagramas de entidad-relación Estos tres modelamientos trabajan juntos conjuntamente para formar un diccionario de datos
¿Qué es un DFD?	Es una herramienta que permite visualizar un sistema como una red de procesos funcionales, conectados entre si por “conductos” y almacenamientos de datos. Representa la transformación de entradas a salidas y es también llamado diagrama de burbujas. Se usa en sistemas operacionales, en los cuales las funciones del sistema son de gran importancia y son más complejas que los datos que este maneja. Hace foco a la funcionalidad del sistema. Se utiliza un rectángulo para representar una entidad externa, esto es, un elemento del sistema u otro sistema que produce información para ser transformada por el software, o recibe información producida por el software. Un círculo (también llamado burbuja) representa un proceso o transformación que es aplicado a los datos (o al control) y los modifica. Una flecha representa uno o más elementos de datos (objetos de dato), indica a dónde va la información.
¿Qué es un proceso de software?	Es un conjunto de actividades y resultados asociados que producen un producto de software. Que tengo que hacer para escribir un software.

¿Qué es un modelo de proceso de software?	Es una representación simplificada de un proceso de software que presenta una visión de ese proceso. Estos modelos pueden incluir actividades que son partes de los procesos y productos de software, y el papel de las personas involucradas.
¿Cuáles son los tipos de modelos de procesos?	Modelos prescriptivos Prescriben un conjunto de elementos del proceso: actividades del marco de trabajo, acciones de la ingeniería del software, tareas, aseguramiento de la calidad y mecanismos de control. Cada modelo de proceso prescribe también un “flujo de trabajo”, es decir de qué forma los elementos del proceso se interrelacionan entre sí. Teoricos. Modelos descriptivos Descripción en la forma en que se realizan en la realidad. Aplicación de los teóricos. Ambos modelos deberían ser iguales
Modelo en cascada	Las etapas se representan cayendo en cascada. Cada etapa de desarrollo se debe completar antes que comience la siguiente. Es un modelo útil para diagramar lo que se necesita hacer, su simplicidad hace que sea fácil explicarlo a los clientes.
Desventajas del modelo en cascada	• No existen resultados concretos hasta que todo esté terminado. • Las fallas más triviales se encuentran al comienzo del período de prueba y las más graves al final. • La eliminación de fallas suele ser extremadamente difícil durante las últimas etapas de prueba del sistema. • Deriva del mundo del hardware y presenta una visión de manufactura sobre el desarrollo de software. • La necesidad de pruebas aumenta exponencialmente durante las etapas finales. • "Congelar" una fase es poco realista. • Existen errores, cambios de parecer, cambios en el ambiente

Modelo en cascada con prototipos	La diferencia de este con el tradicional, es que a partir del análisis de requerimientos se genere un prototipo. Ese prototipo es evaluado con el usuario y si hay algún problema se corrigen los errores. También se genera un prototipo para las siguientes etapas de diseño del sistema y diseño del programa. Entonces cuando se llegan a las últimas etapas, los grandes errores ya han sido corregidos o al menos hay menos errores de los que podrían generarse con el anterior modelo.
Modelo en V	A diferencia del modelo en cascada este muestra un diseño donde los pasos formando una "V", donde por un lado tenemos los análisis y diseño y por el otro las pruebas de cada uno de ellos. Demuestra cómo se relacionan las actividades de prueba con las de análisis y diseño. Sugiere que la prueba unitaria y de integración también sea utilizada para verificar el diseño del programa. La vinculación entre los lados derecho e izquierdo implica que, si se encuentran problemas durante la verificación y validación, entonces el lado izquierdo de la V puede ser ejecutado nuevamente para solucionar el problema
Modelo de prototipos	Un prototipo es un producto parcialmente desarrollado que permite que clientes y desarrolladores examinen algunos aspectos del sistema propuesto, y decidan si éste es adecuado o correcto para el producto terminado. Esta es una alternativa de especificación para tratar mejor la incertidumbre, la ambigüedad y la volubilidad de los proyectos reales. Un sistema inicial se desarrolla rápidamente a partir de una especificación abstracta, generalmente el cliente propone ideas más ambiguas que deben ser evaluadas y llevadas a un prototipo. Éste se refina basándose en las peticiones del cliente.
Tipos de modelos de prototipos	Evolutivos: El objetivo es obtener el sistema a entregar. Permite que todo el sistema o alguna

	de sus partes se construyan rápidamente para comprender o aclarar aspectos y asegurar que el desarrollador, el usuario y el cliente tengan una comprensión unificada tanto de lo que se necesita como de lo que se propone como solución. Descartables: No tiene funcionalidad, se utilizan herramientas de modelado.
Modelo de desarrollo por fases	Se desarrolla el sistema de tal manera que puede ser entregado en piezas. Esto implica que existen dos sistemas funcionando en paralelo: el sistema operacional y el sistema en desarrollo
Tipos de modelos de desarrollo por fases	Incremental: El sistema es particionado en subsistemas de acuerdo con su funcionalidad. Cada entrega agrega un subsistema. Inicia desde una funcionalidad inicial que está acotada y luego se irá agregando funcionalidad. Iterativo: Entrega un sistema completo desde el principio y luego aumenta la funcionalidad de cada subsistema con las nuevas versiones. Se inicia con un sistema “completo” con todas las funcionalidades pero sin estar implementadas y a medida que va habiendo nuevas versiones, se le agrega una nueva operatividad.
Modelo en espiral	Se trabaja por ciclos, cada ciclo está divido en sectores. El primer sector donde se establecen los objetivos y se establecen restricciones, todo lo básico a tener en cuento. El segundo sector establece alternativas y la evaluación y análisis de los riesgos. El tercer sector tiene que ver con el desarrollo en sí y la validación. El último sector se evalúa si se está yendo por buen camino, si se tiene que modificar algo o si hay algún problema. Combina las actividades de desarrollo con la gestión del riesgo. Trata de mejorar los ciclos de vida clásicos y prototipos. Incorpora objetivos de calidad y elimina errores y alternativas no atractivas al comienzo. Permite iteraciones, vuelta atrás y finalizaciones rápidas. Cada ciclo empieza identificando: Los objetivos de la porción correspondiente y las alternativas. Como restricciones tenemos que cada ciclo se

	completa con una revisión que incluye todo el ciclo anterior y el plan para el siguiente.
¿Qué es la calidad? ¿Cuál es la visión holística de la calidad?	Se puede definir como propiedad o conjunto de propiedades inherentes a algo que permiten juzgar su valor. La visión holística dice que es el software quien le da calidad a la empresa, por lo tanto la calidad de la empresa va a depender de la del software. El software a su vez, se compone de 6 componentes de los cuales tiene que asegurar su calidad: • Calidad de software: de las aplicaciones de software construidas o mantenidas • Calidad de la información: está relacionada con la calidad de los datos • Calidad de los datos: que ingresan al sistema • Calidad de la gestión: presupuesto, plinifación y programación • Calidad del servicio: incluye los procesos de atención al cliente • Calidad de la infraestructura: incluye, por ejemplo, la calidad de las redes y el sistema de software
Metodología ágiles	Son un enfoque iterativo incremental de desarrollo de software en el cual se le da prioridad a los resultados directos y que reducen la burocracia tanto como sea posible, adaptandose rapidamente al cambio de los proyectos
¿Cuales son los objetivos de las metodologías ágiles ?	• Producir software de alta calidad con un costo efectivo y en el tiempo apropiado. • Esbozar los valores y principios que deberían permitir a los equipos desarrollar software rápidamente y respondiendo a los cambios que puedan surgir a lo largo del proyecto. • Ofrecer una alternativa a los procesos de desarrollo de software tradicionales, caracterizados por ser

	rígidos y dirigidos por la documentación que se genera en cada una de las actividades desarrolladas.
¿Cuales son los valores de las metodologías ágiles?	• Individuos e interacciones más que procesos y herramientas. • Software operante más que documentaciones completas. • Colaboración con el cliente más que negociaciones contractuales. • Respuesta al cambio más que apegarse a una rigurosa planificación
Diferencias entre metodologias áglies y no ágiles	
Desventajas de las metodologias ágiles	• Involucrar al cliente en todas las etapas cuándo este no tiene el tiempo para asistir en la mismas. • En sistemas con muchos participantes estar atentos a los cambios puede ser complicado. • Mantener la simplicidad requiere un esfuerzo extra • Las grandes compañías pasan años cambiando su cultura de trabajo hacía los valores de la metodología ágil • El SRC forma parte del contrato entre el cliente y el proveedor, en las metodologías ágiles donde la documentación es mínima suele ser complejo reglamentar dicho contrato. • Es una enorme cantidad de esfuerzo en ingeniería de software el mantener y evolucionar softwares existentes, es algo que la metodología esta no contempla pues habla de nuevos sistemas. Los cambios y el mantenimiento son difíciles de implementar cuándo no está la documentación.
Tipos de metodoloias ágiles	• extreme programming • SCRUM • DSDM • Crystal method • FDD

Extreme Programming	Es una disciplina de desarrollo de software basado en los valores de la sencillez, la comunicación, la retroalimentación, la valentía y el respeto. Su acción consiste en llevar a todo el equipo reunido en la presencia de prácticas simples, con suficiente información para ver dónde están y para ajustar las prácticas a su situación particular
Bases de XP	• El desarrollo es iterativo e incremental: pequeñas mejoras, unas tras otras. Cada pieza es integrada apenas está lista. • Pruebas unitarias continuas, frecuentemente repetidas y automatizadas, incluyendo pruebas de regresión. • Programación en parejas, dos programadores en una sola máquina • Frecuente integración con el cliente. • Corrección de todos los errores antes de añadir nueva funcionalidad. • Refactorización del código, se restructura el código con el objetivo de remover duplicaciones, mejorar la legibilidad, simplificarlo y hacerlo más flexible a futuros cambios. • Propiedad del código colectiva, cualquiera puede cambiar código en cualquier parte del sistema en cualquier momento. Motiva a contribuir con nuevas ideas, evitando a la vez que algún programador sea imprescindible. • Simplicidad en el código
Caracteristicas escenciales de XP	• Historias de usuario • Roles (6) • Proceso (6) • Prácticas
Roles que hay en XP	Entre los roles podemos encontrar: al **programador** (responsable de las decisiones técnicas, analistas, diseñadores y codificadores), al **jefe de proyecto** (guía las reuniones y asegura que todo esté en condiciones óptimas), **cliente** (determina qué construir y cuándo, establece las pruebas funcionales), al

	entrenador (responsable del proceso), al **tester** (ayuda al cliente con las pruebas funcionales) y al rastreador (conserva datos históricos y observa sin molestar)
Ciclo de vida de XP	1. Exploración: • Los clientes plantean las historias de usuario que son de interés para la primera entrega del producto. • El equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. • Se construye un prototipo. La fase de exploración toma de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología. 2. Planificación: • El cliente establece la prioridad de cada historia de usuario. • Los programadores realizan una estimación del esfuerzo. • Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Esta fase dura unos pocos días. 3. Iteraciones: • El Plan de Entrega está compuesto por iteraciones de no más de tres semanas. • El cliente es quien decide qué historias se implementarán en cada iteración • Al final de la última iteración el sistema estará listo para entrar en producción. Esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. 4. Producción • Esta fase requiere de pruebas adicionales y revisiones de rendimiento antes de que el sistema sea trasladado al entorno del cliente. • Al mismo tiempo, se deben tomar decisiones sobre la inclusión de

	nuevas características a la versión actual, debido a cambios durante esta fase. 5. Mantenimiento • Mientras la primera versión se encuentra en producción, el proyecto XP debe mantener el sistema en funcionamiento al mismo tiempo que desarrolla nuevas iteraciones. • La fase de mantenimiento puede requerir nuevo personal dentro del equipo y cambios en su estructura. 6. Muerte • Es cuando el cliente no tiene más historias para ser incluidas en el sistema. • Se genera la documentación final del sistema y no se realizan más cambios en la arquitectura. • La muerte del proyecto también ocurre cuando el sistema no genera los beneficios esperados por el cliente o cuando no hay presupuesto para mantenerlo.
Practicas de XP	**Testing:** Los programadores continuamente escriben pruebas unitarias, las cuales deben correr sin problemas para que el desarrollo continúe. Los clientes escriben pruebas demostrando la funcionalidad. **Refactoring**: Actividad constante de reestructuración del código con el objetivo de remover duplicación de código, mejorar su legibilidad, simplificarlo y hacerlo más flexible para facilitar los posteriores cambios. **Programación de a Pares**: Todo el código de producción es escrito por dos programadores en una máquina. **Propiedad Colectiva del Código**: Cualquiera puede cambiar código en cualquier parte del sistema en cualquier momento. Motiva a contribuir con nuevas ideas, evitando a la vez que algún programador sea imprescindible.

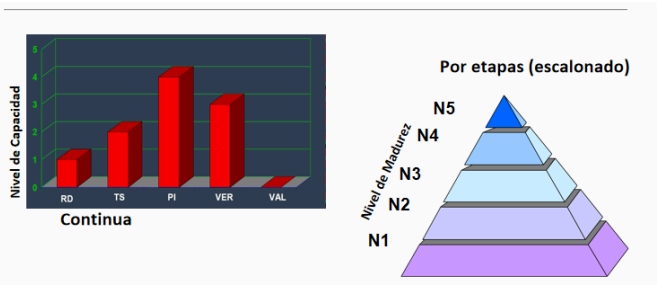
	Integración Continua: Cada pieza de código es integrada en el sistema una vez que esté lista. Así, el sistema puede llegar a ser integrado y construido varias veces en un mismo día. Reduce la fragmentación de los esfuerzos de los desarrolladores por falta de comunicación sobre lo que puede ser reutilizado o compartido. **Semana de 40-horas**: Se debe trabajar un máximo de 40 horas por semana. El trabajo extra desmotiva al equipo. Los proyectos que requieren trabajo extra para intentar cumplir con los plazos suelen al final ser entregados con retraso. **Cliente en el lugar de desarrollo**: El cliente tiene que estar presente y disponible todo el tiempo para el equipo. **Estándares de Codificación:** Los programadores escriben todo el código de acuerdo con reglas que enfatizan la comunicación a través del mismo.
Scrum	Scrum es un proceso en el que se aplican, de manera regular, un conjunto de mejores prácticas para trabajar en equipo y obtener el mejor resultado posible de un proyecto. Estas prácticas se apoyan unas a otras y su selección tiene origen en un estudio de la manera de trabajar de equipos altamente productivos. En Scrum se realizan entregas parciales y regulares del resultado final del proyecto, priorizadas por el beneficio que aportan al receptor del proyecto.
Principios del SCRUM	
Principios del SCRUM	- Eliminar el desperdicio, es decir, no generar artefactos o cosas que no son de valor para el cliente - Construir la calidad con el producto, desde el momento 0 se busca la calidad - Crear conocimiento, en la práctica se hace el conocimiento - Diferir las decisiones, hasta que no sea hora de tomar una decisión, no tomarla. Recién se

	decide cuándo sea necesario - Entregar rápido: Debe ser una de las ventajas competitivas más importantes - Respetar a las personas - Optimar el todo, el proceso y todo lo relacionado
Roles en SCRUM	Los roles en scrum son tan sólo 4: el **propietario** (conoce y marca las prioridades del producto), **scrum master** (el jefe, es quién maneja a las personas y asegura que trabajen bien, las protege de las presiones ext4ernas), **scrum team** (el equipo, responsables de implementar la metodología scrum en su trabajo) y los **usuarios o clientes**.
Artefactos del SCRUM	• Product Backlog: es la lista maestra que contiene toda la funcionalidad deseada en el producto. La característica más importante es que la funcionalidad se encuentra ordenada por un orden de prioridad. • Sprint Backlog (lista de Sprint): es la lista que contiene toda la funcionalidad que el equipo se comprometió a desarrollar durante un Sprint determinado. • Burndown Chart: muestra un acumulativo del trabajo hecho, día-a-día.
Proceso del SCRUM	• Scrum es iterativo e incremental • Se busca poder atacar todos los problemas que surgen durante el desarrollo del proyecto. • El nombre Scrum se debe a que durante los Sprints, lo que serían las fases de desarrollo, se solapan, de manera que no es un proceso de cascada por cada iteración, si no que tenemos todas éstas etapas juntas que se ejecutan una y otra vez, hasta que se crea suficiente. • Este solapamiento de fases se puede asemejar a un scrum de rugby, en el

	cual todos los jugadores (o roles, en nuestro caso), trabajan juntos para lograr un objetivo
Cuando usar SCRUM y cuando usar XP	Scrum está pensado para ser aplicado en proyectos en donde el "caos" es una constante, aquellos proyectos en los que tenemos requerimientos dinámicos, y que tenemos que implementar tecnología de punta. Esos proyectos difíciles, que con los enfoques tradicionales se hace imposible llegar a buen puerto. XP es mejor para proyectos pequeños y no tan apresurados
Desarrollo de Software Basado en Modelos (MBD)	Es un intermedio entre las metodologías tradicionales y las ágiles. Este modelo apunta a que la construcción de un sistema de software debe ser precedida por la construcción de un modelo. Un modelo del sistema consiste en una conceptualización del dominio del problema y actúa como una especificación precisa de los requerimientos que el sistema de software debe satisfacer (Abstracción de elementos del problema, comunicación, negociación con el usuario). Se necesita generar modelos para dominios de problemas (abstracción del problema, de los elementos y cómo se comunica con el usuario). Se tiene un problema real, luego su modelo y finalmente su implementación
Desarrollo de Software Dirigido por Modelos (MDD)	El adjetivo «dirigido» en MDD, a diferencia de «basado» (MBD), enfatiza que este paradigma asigna a los modelos un rol central y activo: son al menos tan importantes como el código fuente. El modelo me dirige el desarrollo. • Mayor nivel de abstracción en la especificación tanto del problema a resolver como de la solución correspondiente. • Aumento de confianza en la automatización asistida por computadora para soportar el análisis, el diseño y la ejecución.

	• Uso de estándares industriales como medio para facilitar las comunicaciones, la interacción entre diferentes aplicaciones y productos, y la especialización tecnológica. • Los modelos son los conductores primarios en todos los aspectos del desarrollo de software. • Los modelos pasan de ser entidades contemplativas (es decir, artefactos que son interpretados por los diseñadores y programadores) para convertirse en entidades productivas a partir de las cuales se deriva la implementación en forma automática.
PIM y PSM	Del MDD surge los PIM y los PSM, dos modelos que se usan al mismo tiempo. • PIM (platform independent model), es un modelo independiente de la plataforma, no contiene información acerca de la misma o la tecnología que es usada para implementarlo. • PSM (platform specific model) este modelo incluye información acerca de la tecnología específica que se usará para su implementación sobre una plataforma específica. Y estos dos modelos pasan por una transformación, en la cual se especifica el proceso de conversión de un modelo a otro. Cada transformación incluye, al menos, un PIM, un modelo de la plataforma, una transformación y un PSM. La transformación consiste en una colección de reglas, las cuáles son especificaciones no ambiguas de las formas en que un modelo (o parte de él) puede ser usado para crear otro modelo (o parte de él). El patrón MDD es normalmente utilizado sucesivas veces para producir una sucesión de transformaciones.
Clidad de producto y calidad de proceso	Producto: La estandarización del producto define las propiedades que debe satisfacer el producto software resultante.

	Proceso: La estandarización del proceso define la manera de desarrollar el producto software. Sin un buen proceso de desarrollo es casi imposible obtener un buen producto.
¿Cómo se mide la calidad del software?	La calidad del software se divide en dos, las cuales son dependientes entre si (sin un buen proceso de desarrollo es imposible obtener un buen producto) : • Calidad del producto obtenido: Un producto es de buena calidad si satisface las necesidades del usuario para los objetivos para lo que fue concebido. Para medir la calidad del producto existe la norma ISO 25000 • Calidad del proceso de desarrollo: Un proceso mal concebido va a generar como resultado productos de mala calidad. Para medir la calidad del proceso de desarrollo existen varias normas como por ejemplo la ISO12207, la ISO 15504 y la CMMI
CMM	CMM (Capability Maturity Model): Es un modelo que evalua y mejora los procesos de desarrollo de software dentro de una organización y ayuda a las empresas a medir la madurez y capacidad de de sus procesos:
CMMI	CMMI (Capability Maturity Model Integrated): Es una evolución de CMM, diseñada para ser más flexible y aplicable no solo al desarrollo de software, sino también a otros sectores como manufactura, servicios, e ingeniería de sistemas.Posee 2 enfoques diferentes según las necesidades de quien vaya a implementarlo • Enfoque escalonado: Centra su foco en la **madurez** de la organización, al igual que CMM. Es global • Enfoque continuo: Enfoca las actividades de mejora y evaluación en la **capacidad** de los diferentes

	procesos. Presenta 6 niveles de capacidad que indican qué tan bien se desempeña la organización en un área de procesos individual. 
¿Que son los niveles de madurez?	Los niveles de madurez son una clasificación que evalúa la capacidad y calidad de los procesos de una organización. Representan el grado de formalización, estandarización y control que tiene una empresa sobre sus procesos, especialmente en el desarrollo de software o la gestión organizacional. El concepto de **niveles de madurez** proviene principalmente del **Modelo de Madurez de Capacidades (CMM)** y su evolución, el **Modelo Integrado de Madurez de Capacidades (CMMI)**. Nivel 1: Inicial • Proceso impredecible, poco controlado y reactivo • Los procesos no están definidos ni documentados Nivel 2: Gestionado • Proceso caracterizado por proyectos y frecuentemente reactivo • Algunos procesos básicos están definidos y documentados para proyectos específicos Nivel 3: Definido • Proceso caracterizado por la organización y proactivo • Los procesos están documentados, estandarizados y aplicados consistentemente en toda la organización. Nivel 4: Gestionado cuantitativamente:

	• El proceso es controlado cuantitativamente • Los procesos son medidos y controlados con métricas cuantitativas. • Se enfocan en mejorar la calidad mediante el análisis de datos. Nivel 5: Óptimo • Enfoque en la mejora del proceso • La organización se centra en la mejora continua mediante la innovación y las lecciones aprendidas
¿Cómo se relacionan los conceptos de calidad con los de CMM y CMMI?	CMM y CMMI son marcos que se relacionan directamente con los conceptos de calidad, ya que proporcionan un enfoque sistemático para mejorar los procesos de desarrollo de software, lo que a su vez garantiza productos de alta calidad y procesos eficientes.
¿Qué es una familia de normas?	Una **familia de normas** es un conjunto de estándares relacionados que abordan diferentes aspectos de un tema común. Estas normas son desarrolladas por organizaciones internacionales como la **ISO (International Organization for Standardization)** y buscan proporcionar lineamientos, métricas y mejores prácticas para garantizar calidad, interoperabilidad, y eficiencia en procesos, productos o servicios. Cada familia se centra en un área específica y las normas que la componen están organizadas para cubrir múltiples facetas del tema, desde requisitos hasta guías para implementación.
Ejemplos de familias de normas	Por ejemplo la familia de normas ISO/IEC 25000, también conocida como SQuaRE (Software Quality Requirements and Evaluation) es un conjunto de estándares internacionales que define un marco para evaluar y gestionar la **calidad del software**. Otro ejemplo es la familia ISO 9000 es un conjunto de normas de “gestión de la calidad” aplicables a cualquier tipo de organización

	con el objetivo de obtener mejoras en la organización y, eventualmente arribar a una certificación, punto importante a la hora de competir en los mercados globales.
Jrp	Proceso mediante el cual se conducen reuniones en grupo altamente estructuradas con el propósito de entender un problema y definir requerimientos