¿Qué es la ingeniería de software?

La ingeniería de software es una disciplina de la ingeniería que se interesa por todos los aspectos de la producción de software. Sus actividades fundamentales son la especificación, desarrollo, validación y evolución del software. Para todo esto hay un conjunto de teorías, métodos y estrategias para llevar estas disciplina a cabo. Los aspectos de la producción no sólo comprenden los procesos técnicos del desarrollo de software, sino también se realizan actividades como la gestión de proyectos y el desarrollo de herramientas, métodos y teorías de apoyo a la producción de software.

Definición de Ingeniería de Software de la IEE:

- **Usa métodos sistemáticos cuantificables:**
  La cuantificación rigurosa de recursos, procesos y productos es una precondición para optimizar productividad y calidad. La "metrificación" y el control estadístico de procesos son claves en Ingeniería de Software.
- **Procede dentro de tiempos y costos estimados:**
  Un Ingeniero de Software debe cumplir contratos en tiempo y costos como es normal en obras de Ingeniería. Ello presupone la capacidad de medir, estimar, planificar y administrar proyectos.
- **Es utilizada para el "Desarrollo, operación y mantenimiento":**
  La Ingeniería de Software se ocupa de todo el ciclo de vida de un producto, desde su etapa inicial de planificación y análisis de requerimientos hasta la estrategia para determinar cuándo y cómo debe ser retirado de servicio.

¿Qué conocimiento debe tener un ingeniero de software?

Los ingenieros hacen que las cosas funcionen. Aplican teorías, métodos y herramientas donde es adecuado. Sin embargo, los usan de manera selectiva y siempre tratan de encontrar soluciones a problemas, incluso cuando no hay teorías ni métodos aplicables. Los ingenieros también reconocen que deben trabajar ante restricciones organizacionales y financieras, de modo que buscan soluciones dentro de tales limitaciones.

El Ingeniero debe conocer las tecnologías y productos: sistemas operativos, lenguajes, bases de datos, sistemas generadores de interfaces, bibliotecas de código. Así también debe conocer técnicas de administración de proyectos: planificación, análisis de riesgos, control de calidad, seguimiento de proyectos, control de subcontratistas, etc.

La Ingeniería de Software se desarrolla en un marco económico, social y legal. Los IS deben aceptar responsabilidades más amplias que las responsabilidades técnicas. No debe utilizar su capacidad y habilidades de forma deshonesta, o de forma que deshonre la profesión.

Responsabilidad y ética laboral

- **Confidencialidad:** Respetar la confidencialidad de sus empleados y clientes.
- **Competencia:** No falsificar el nivel de competencia y aceptar responsabilidades fuera de su capacidad.
- **Derechos de la propiedad intelectual:** Conocer las leyes vigentes sobre las patentes y copyright.
- **Uso inapropiado de las computadoras:** No debe utilizar sus habilidades técnicas para utilizar de forma inapropiada otras computadoras.

Requerimientos

¿Qué es un requerimiento?

Los requerimientos para un sistema son descripciones de lo que el sistema debe hacer: el servicio que ofrece y las restricciones en su operación. Tales requerimientos reflejan las necesidades de los clientes por un sistema que atienda cierto propósito. Al proceso de descubrir, analizar, documentar y verificar estos servicios y restricciones se le llama ingeniería de requerimientos.

En algunos casos, un requerimiento es simplemente una declaración abstracta de alto nivel de un servicio que debe proporcionar el sistema o una restricción de este. En el otro extremo, es una definición detallada y formal de una función del sistema.

Las principales fuentes de requerimientos son: stakeholders, documentación y especificaciones de sistemas similares.

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.
- Puede gastarse tiempo y dinero construyendo el sistema erróneo.

Requerimientos funcionales y no funcionales

Los **requerimientos funcionales** de un sistema describen lo que el sistema debe hacer o incluso cómo no debe comportarse. Describen una interacción entre el sistema y su ambiente, cómo debe comportarse el sistema ante determinado estímulo. Describen con detalle la funcionalidad del mismo, son independientes de la implementación de la solución y se pueden expresar de distintas formas.

Los **requerimientos no funcionales**, como su nombre sugiere, son aquellos requerimientos que no se refieren directamente a las funciones específicas que proporciona el sistema, sino a las propiedades emergentes de éste como la fiabilidad, el tiempo de respuesta y la capacidad de almacenamiento. Describen una restricción sobre el sistema que limita nuestras elecciones en la construcción de una solución al problema. Entre ellos podemos encontrar:

- **Requerimientos del producto:** Especifican el comportamiento del producto (usabilidad, eficiencia, rendimiento, espacio, fiabilidad, portabilidad).
- **Requerimientos organizacionales:** Se derivan de las políticas y procedimientos existentes en la organización del cliente y en la del desarrollador (entrega, implementación, estándares).
- **Requerimientos externos:** Interoperabilidad, legales, privacidad, seguridad, éticos.

Otros tipos de requerimientos:

- **Requerimientos del dominio:** Reflejan las características y restricciones del dominio de la aplicación del sistema. Pueden ser funcionales o no funcionales y pueden restringir a los anteriores. Como se especializan en el dominio son complicados de interpretar.
- **Requerimientos por Prioridad:** estos deben ser absolutamente satisfechos, son deseables pero no indispensables, son posibles, pero que podrían eliminarse.
- **Requerimientos del Usuario:** Son declaraciones en lenguaje natural y en diagramas de los servicios que se espera que el sistema provea y de las restricciones bajo las cuales debe operar. Pueden surgir problemas por falta de claridad, confusión de requerimientos, conjunción de requerimientos.
- **Requerimientos del Sistema:** Establecen con detalle los servicios y restricciones del sistema. Es difícil excluir toda la información de diseño (arquitectura inicial, interoperabilidad con sistemas existentes, etc.)

¿Qué es un Stakeholder?

El término stakeholder se utiliza para referirse a cualquier persona o grupo que se verá afectado por el sistema, directa o indirectamente. Aportan visiones diferentes con respecto al sistema.

Entre los stakeholders se encuentran los usuarios finales que interactúan con el sistema y todos aquellos en la organización que se pueden ver afectados por su instalación. Otros stakeholders del sistema pueden ser los ingenieros que desarrollan o dan mantenimiento a otros sistemas relacionados, los gerentes del negocio, los expertos en el dominio del sistema y los representantes de los trabajadores.

Puntos de vista:

- **Punto de vista de los interactuadores:** representan a las personas u otros sistemas que interactúan directamente con el sistema. Pueden influir en los requerimientos del sistema de algún modo.
- **Punto de vista indirecto:** representan a los stakeholders que no utilizan el sistema ellos mismos pero que influyen en los requerimientos de algún modo.
- **Punto de vista del dominio:** representan las características y restricciones del dominio que influyen en los requerimientos del sistema.

Elicitación de requerimientos

¿Qué es la elicitación de requerimientos?

Es el proceso de adquirir todo el conocimiento relevante necesario para producir un modelo de los requerimientos de un dominio de problema. Sus objetivos: conocer el dominio del problema para poder comunicarse con clientes y usuarios y entender sus necesidades, conocer el sistema actual (manual o informatizado) e identificar las necesidades, tanto explícitas como implícitas, de clientes y usuarios y sus expectativas sobre el sistema a desarrollar.

La elicitación de requisitos es una actividad principalmente de carácter social, mucho más que tecnológico. Los problemas que se plantean son por tanto de naturaleza psicológica y social, más que técnicos.

Problemas a la hora de elicitar:

- **Comunicación:**
  - Dificultad para expresar claramente las necesidades.
  - No ser conscientes de sus propias necesidades.
  - No entender cómo la tecnología puede ayudar.
  - Miedo a parecer incompetentes por ignorancia tecnológica.
  - No tomar decisiones por no poder prever las consecuencias, no entender las alternativas o no tener una visión global.
  - Cultura y vocabulario diferentes, conflictos personales o políticos.
  - Intereses distintos en el sistema a desarrollar.
  - Medios de comunicación inadecuados (diagramas que no entienden los clientes y usuarios).
- **Limitaciones cognitivas (del desarrollador)**
  - No conocer el dominio del problema.
  - Hacer suposiciones sobre el dominio del problema.
  - Hacer suposiciones sobre aspectos tecnológicos.
  - Hacer simplificaciones excesivas.

- **Conducta humana**
  - Conflictos y ambigüedades en los roles de los participantes.
  - Pasividad de clientes, usuarios o ingenieros de requisitos.
  - Temor a que el nuevo sistema lo deje sin trabajo.
- **Técnicos**
  - Complejidad del dominio del problema.
  - Complejidad de los requisitos.
  - Múltiples fuentes de requisitos.
  - Fuentes de información poco claras.

Fuentes de información:

Métodos discretos:

Los métodos discretos son menos perturbadores que otras formas de averiguar los requerimientos. Se consideran insuficientes para recopilar información cuando se utilizan por sí solos, por lo que deben utilizarse junto con uno o varios de los métodos. Utilizar diferentes métodos para acercarse a la organización es una práctica inteligente mediante la cual podrá formarse un panorama más completo de los requerimientos.

- **Muestreo de la documentación, los formularios y los datos existentes.**
  - Organigrama (identificar el propietario, usuarios claves).
  - Memos, notas internas, minutas, registros contables.
  - Solicitudes de proyectos de sistemas de información anteriores.
  - Permiten conocer el historial que origina el proyecto.
- **Investigación y visitas al lugar.**
  - Investigar el dominio.
  - Patrones de soluciones (mismo problema en otra organización).
  - Revistas especializadas.
  - Buscar problemas similares en internet.
  - Consultar otras organizaciones.
- **Observación del ambiente de trabajo.**
  - Determinar quién y cuándo será observado.
  - Obtener el permiso de la persona y explicar el porqué será observado.
  - Mantener bajo perfil.
  - Tomar nota de lo observado.
  - Revisar las notas con la persona apropiada.
  - No interrumpir a la persona en su trabajo.

Métodos interactivos:

Los métodos interactivos pueden usarse para obtener los requerimientos de los miembros de la organización de distintas formas. El analista utilizará métodos interactivos como entrevistas, muestreos e investigación de datos duros, además de los cuestionarios y los métodos discretos, como observar el comportamiento de los encargados al tomar las decisiones y sus entornos de oficina, y los métodos integrales como la creación de prototipos. El analista utilizará estos métodos para plantear y responder muchas preguntas relacionadas con la interacción humano-computadora. La base es hablar con las personas en la organización y escuchar para comprender.

- Cuestionarios.
- Entrevistas.
- Planeación conjunta de Requerimientos (JRP o JAD).
- Lluvia de Ideas - Brainstorming.

Elicitación de requerimientos - Cuestionarios

¿Qué es un cuestionario?

Un cuestionario es un documento que permite al analista recabar información y opiniones de los encuestados. En este se muestran un conjunto de preguntas con el fin de recolectar hechos de un gran número de personas, detectar un sentimiento generalizado, detectar problemas entre usuarios y cuantificar respuestas. El tipo de información obtenida es la actitud, las creencias y el comportamiento de las personas. Esta técnica es utilizada cuando las personas están dispersas geográficamente o cuando son un número significativo, también cuando queremos obtener opiniones generales e identificar problemas generales.

- **Ventajas:**
  - Respuesta rápida
  - Económicos
  - Anónimos
  - Estructurados de fácil análisis
- **Desventajas:**
  - Número bajo de respuestas
  - No responde a todas las preguntas
  - Preguntas rígidas
  - No se puede realizar el análisis corporal
  - No se pueden aclarar respuestas incompletas
  - Difíciles de preparar

Formato de un cuestionario:

Generalmente las preguntas de los cuestionarios se agrupan en un orden específico que debe ser claro para en encuestado y a su vez las preguntas no deben ser tendenciosas ni mostrar el punto de vista u opinión del encuesta dor. La redacción de cada pregunta del cuestionario debe ser clara, completa y no ambigua. La explicación/presentación del cuestionario debe ser clara e indica el objetivo y debe tener un estilo ameno y no amenazante. El diseño elegido para el cuestionario debe ser adecuado para su propósito (lugares para completar, escribir, marcar).

Tipos de cuestionario:

- **Cuestionario de formato libre:** Cuestionario diseñado para ofrecer al encuestado más laxitud en la respuesta. Se formula una pregunta, y el encuestado registra la respuesta en el espacio provisto después de la pregunta. Se diseñan para permitir a los usuarios que ejerciten más libertad o flexibilidad en sus respuestas a cada pregunta.
- **Cuestionario de formato fijo:** Cuestionario que contiene preguntas que requieren la selección de una respuesta entre respuestas disponibles predefinidas. Requieren que el usuario seleccione una respuesta de un conjunto de respuestas posibles previamente definido. Dada cualquier pregunta, el encuestado debe escoger de las respuestas disponibles.

Elicitación de requerimientos - Entrevistas

¿Qué es una entrevista?

La entrevista es una técnica de exploración mediante la cual el analista de sistemas recolecta información de las personas a través de la interacción cara a cara. Es una conversación con un propósito específico, que se basa en un formato de preguntas y respuestas en general. El tipo de información obtenida mediante esta técnica son opiniones, objeticos, sentimientos y procedimientos informales.

- **Ventajas:**
  - El entrevistado se siente incluido en el proyecto.
  - Es posible obtener una retroalimentación del encuestado.
  - Es posible adaptar las preguntas de acuerdo al entrevistado.
  - Información no verbal observando las acciones y expresiones del entrevistado.
- **Desventajas:**
  - Costosas.
  - Tiempo y recursos humanos.
  - Las entrevistas dependen en gran parte de las habilidades del entrevistador.
  - No aplicable a distancia.

Formato de una entrevista:

Las entrevistas deben contar con un guion donde estén todas las preguntas previamente redactadas. Estas deben permiten obtener: opiniones, objetivos, procedimientos informales y sentimientos. El guion debe informar a quién estará dirigida la entrevista y el tema, así también el lugar, fecha y horario de la misma. Todas las preguntas deberán tener un tiempo asignado y también deberá haber un espacio para la presentación de la misma. Además debe mostrarse el tiempo asignado para cada parte de la entrevista y finalmente un espacio para notas y comentarios generales. (*El guion puede verse en el libro de Whitten, Análisis de Sistemas Diseño y Métodos.*)

Tipo de entrevistas:

- **Estructuradas (Cerradas)**
  - El encuestador tiene un conjunto específico de preguntas para hacérselas al entrevistado
  - Se dirige al usuario sobre un requerimiento puntual
  - No permite adquirir un amplio conocimiento del dominio
- **No estructuradas (Abiertas)**
  - El encuestador lleva a un tema en general
  - Sin preparación de preguntas específicas
  - Iniciar con preguntas que no dependen del contexto, para conocer el problema, la gente involucrada, etc.

Tipo de preguntas:

- **Abiertas:** Permite al encuestado responder de cualquier manera, permitiendo espontaneidad. También revelan una nueva línea de preguntas y hacen más interesante la entrevista. Como punto en contra pueden dar muchos detalles irrelevantes y se puede perder el control de la entrevista. Así también puede parecer que el entrevistador no tiene los objetivos claros.
- **Cerradas:** Las respuestas son directas, cortas o de selección específica. Ahorran tiempo, se mantiene más fácil el control de la entrevista y se consiguen datos relevantes. Como desventaja se tiene que pueden aburrir al encuestado y no se obtienen detalles de las respuestas.
- **Sondeo:** Permite obtener más detalle sobre un tema puntual.

Organización de una entrevista:

- **Piramidal - inductivo:**
  Empiezan con preguntas cerradas, de fácil y rápida resolución y termina con preguntas abiertas.
- **Embudo - deductivo:**
  Empiezan con preguntas abiertas y finalizan con preguntas cerradas o de seguimiento.
- **Diamante - Combinada:**
  Empiezan con preguntas cerradas, se promedia la entrevista con preguntas abiertas y se termina nuevamente con preguntas cerradas.

Preparación previa (Kendall):

1. **Leer los antecedentes:** poner atención en el lenguaje. Buscar un vocabulario en común. Imprescindible para poder entender entrevistado.
2. **Establecer los objetivos de la entrevista:** usando los antecedentes. Los directivos suelen proporcionar una visión general, mientras que los futuros usuarios, una más detallada.
3. **Seleccionar los entrevistados:** se debe minimizar el número de entrevistas Los entrevistados deben conocer con antelación el objetivo de la entrevista y las preguntas que se le van a hacer.
4. **Planificación de la entrevista y preparación del entrevistado:** establecer fecha, hora, lugar y duración de cada entrevista de acuerdo con el entrevistado.
5. **Selección del tipo de preguntas a usar y su estructura.**

Reglas para una entrevista:

Haga

- Vístase adecuadamente.
- Sea cortés.
- Escuche cuidadosamente.
- Mantenga el control de la entrevista.
- Explore.
- Observe los gestos y la comunicación no oral.
- Sea paciente.
- Mantenga al entrevistado en calma.
- Mantenga su autocontrol.
- Termine a tiempo.

Evite

- Suponer que una respuesta esté terminada o que no lleva a ningún lado.
- Revelar pistas orales y no orales.
- Usar la jerga.
- Revelar sus sesgos personales.
- Hablar en lugar de escuchar.
- Suponer cualquier cosa acerca del tema o el entrevistado.
- Uso de la grabadora: una señal de habilidades deficientes para escuchar.

Elicitación de requerimientos - Planeación Conjunta de Requerimientos (JRP)

¿Qué es la Planeación Conjunta de Requerimientos?

La Planeación conjunta de requerimientos (joint requirements planning, JRP) es el proceso mediante el cual se conducen reuniones de grupo altamente estructurados con el propósito de analizar problemas y definir requerimientos. Las sesiones de planeación conjunta de requerimientos incluyen una amplia variedad de participantes y de papeles. Se espera que cada participante asista y participe activamente en la sesión completa de JRP.

- **Ventajas**
  - Ahorro de tiempo
  - Usuarios involucrados
  - Desarrollos creativos
- **Desventajas**
  - Es difícil organizar los horarios de los involucrados
  - Es complejo encontrar un grupo de participantes integrados y organizados

Participantes:

- **Patrocinador:** Miembro de la dirección con autoridad sobre los departamentos que participan, es el responsable del proyecto, toma las decisiones finales.
- **Facilitador:** Dirige las sesiones y tiene amplias habilidades de comunicación y negociación.
- **Usuarios y Gerentes:** Los usuarios comunican los requerimientos y los gerentes los aprueban.
- **Secretarios:** Llevan el registro de la sesión y van publicando los resultados realizados.
- **Equipos de TI:** Escuchan y toman nota de los requerimientos.

Cómo planear las sesiones de JRP:

- **Selección de una ubicación para las sesiones de JRP:** Siempre que sea posible, las sesiones de JRP deberán conducirse lejos del lugar de trabajo de la compañía. Al celebrarse la sesión de JRP en una ubicación extra muros, los asistentes pueden concentrarse en los aspectos y actividades relacionados con la sesión de JRP y evitar interrupciones y distracciones que ocurrirían en su lugar de trabajo habitual. También, si la sesión de JRP incluye mucha gente, pueden ser necesarias varias salas más pequeñas para que se reúnan grupos separados de personas y que se adentren en la discusión de aspectos específicos. Las reuniones deberán tener todas las herramientas (como proyectores, pizarrones, internet, etc.) y comodidades para que los participantes se sientan a gusto.
- **Selección de los participantes:** los participantes seleccionados incluyen al facilitador de JRP, el (los) secretario(s), y los representantes de la comunidad de usuarios. Los usuarios deberán ser personas clave que dominen el área de negocios. Así, el analista debe asegurarse de que la gerencia está comprometida con el proyecto de JRP y que quieren no solamente permitir sino también requieren que estas personas clave participen. Generalmente todas las personas de TI asignadas al equipo de proyecto participan en la sesión de JRP. También pueden seleccionarse otros especialistas de TI para abordar aspectos técnicos específicos alusivos al proyecto.
- **Preparar la agenda:** El facilitador de JRP debe preparar documentación para informar brevemente a los participantes acerca del alcance y los objetivos de la sesiones. Además, deberá prepararse una agenda para cada sesión de JRP y distribuirse antes de cada sesión. La agenda determina los aspectos que se van a discutir durante la sesión y el tiempo asignado a cada tema. La agenda deberá contener tres partes: la introducción, el cuerpo y la conclusión. La introducción pretende comunicar las expectativas de la sesión, las reglas de campo, e influir o motivar a los asistentes para participar. El cuerpo pretende detallar los temas o aspectos que van a abordarse en la sesión de JRP. Finalmente, la conclusión representa el tiempo apartado para resumir la sesión del día y para recordar a los asistentes de los aspectos no solucionados (para desarrollarse posteriormente).

Elicitación de requerimientos - Brainstorming

¿Qué es el Brainstorming?

La lluvia de ideas (Brainstorming) es una técnica para generar ideas al alentar a los participantes para que ofrezcan tantas ideas como sea posible en un corto tiempo sin ningún análisis hasta que se hayan agotado las ideas. Se promueve el desarrollo de ideas creativas para obtener soluciones. Se realizan reuniones del equipo involucrado en la resolución del problema, conducidas por un director. Es una técnica efectiva para identificar posibles soluciones. Resulta especialmente efectiva cuando se utiliza una estrategia o marco de referencia organizados, como los componentes de IS u otras características de IS. La lluvia de ideas debe abarcar soluciones que representen la compra, construcción y combinaciones de compra y construcción.

Ingeniería de requerimientos

¿Qué es la ingeniería de requerimientos?

La ingeniería de requerimientos es la disciplina para desarrollar una especificación completa, consistente y no ambigua, la cual servirá como base para acuerdos comunes entre todas las partes involucradas y en donde se describen las funciones que realizará el sistema. Es el proceso por el cual se transforman los requerimientos declarados por los clientes, ya sean hablados o escritos, a especificaciones precisas, no ambiguas, consistentes y completas del comportamiento del sistema, incluyendo funciones, interfaces, rendimiento y limitaciones.

También es el proceso mediante el cual se intercambian diferentes puntos de vista para recopilar y modelar lo que el sistema va a realizar. Este proceso utiliza una combinación de métodos, herramientas y actores, cuyo producto es un modelo del cual se genera un documento de requerimientos. “Ingeniería de requerimientos” es un enfoque sistémico para recolectar, organizar y documentar los requerimientos del sistema; es también el proceso que establece y mantiene acuerdos sobre los cambios de requerimientos, entre los clientes y el equipo del proyecto.

La meta del proceso de ingeniería de requerimientos es crear y mantener un documento de requerimientos del sistema. El proceso general corresponde a cuatro subprocesos de alto nivel de la ingeniería de requerimientos:

1. **Estudio de viabilidad.**
2. **Obtención y análisis de requerimientos.**
3. **Especificación de requerimientos.**
4. **Validación de requerimientos.**
5. **Gestión de requerimientos.**

Importancia

- Permite gestionar las necesidades del proyecto en forma estructurada.
- Mejora la capacidad de predecir cronogramas de proyectos.
- Disminuye los costos y retrasos del proyecto.
- Mejora la calidad del software.
- Mejora la comunicación entre equipos.
- Evita rechazos de usuarios finales.

IR - Estudio de Viabilidad

¿Qué es un Estudio de viabilidad?

A partir de una descripción resumida del sistema se elabora un informe que recomienda la conveniencia o no de realizar el proceso de desarrollo.

Nos debemos hacer estas preguntas para saber si es factible o no iniciar el proyecto:

- ¿El sistema contribuye a los objetivos generales de la organización?
- ¿El sistema se puede implementar con la tecnología actual?
- ¿El sistema se puede implementar con las restricciones de costo y tiempo?
- ¿El sistema puede integrarse a otros que existen en la organización?

Una vez que se ha recopilado toda la información necesaria para contestar las preguntas anteriores se debería hablar con las fuentes de información para responder nuevas preguntas y luego se redacta el informe, donde debería hacerse una recomendación sobre si debe continuar o no el desarrollo

IR - Especificación de Requerimientos

¿Qué es la Especificación de Requerimientos?:

Representa una comprensión entre el cliente y el desarrollador de lo que el cliente necesita o desea, y por lo general es escrito en forma conjunta por el cliente y el desarrollador. Por la otra parte, la especificación de requerimientos reitera la definición en los términos técnicos apropiados para el desarrollo del diseño de un sistema; es la contrapartida técnica al documento de definición de requerimientos y es escrito por analistas de requerimientos.

Este paso utiliza los requerimientos ya obtenidos, que suelen ser ambiguos y poco claros, y se los traduce de manera más clara y precisa para que no haya diferentes interpretaciones entre el cliente y los integrantes del equipo de desarrollo.

Cada requerimiento obtenido debe cumplir estas **propiedades:**

- **Necesario:** Su omisión provoca una deficiencia.
- **Conciso:** Fácil de leer y entender.
- **Completo:** No necesita ampliarse.
- **Consistente:** No contradictorio con otro.
- **No ambiguo:** Tiene una sola implementación.
- **Verificable:** Puede testearse a través de inspecciones, pruebas, etc.

Dentro del documento de la especificación de requerimientos deben incluir sus **objetivos:**

- Permitir que los desarrolladores expliquen cómo han entendido lo que el cliente pretende del sistema.
- Indicar a los diseñadores qué funcionalidad y características va a tener el sistema resultante.
- Indicar al equipo de pruebas qué demostraciones llevar a cabo para convencer al cliente de que el sistema que se le entrega es lo que había pedido.

Documentación necesaria:

- Documento de definición de requerimientos: todas las cosas que el cliente espera que haga el sistema.
- Documento de especificación de requerimientos: Definición en términos técnicos
- Documento de especificación de requerimientos de Software IEEE Std. 830-1998 (SRS): colección de buenas prácticas para escribir especificaciones de requerimientos de software (SRS).

Aspectos básicos de una especificación de requerimientos:

- Funcionalidad: ¿Qué debe hacer el software?
- Interfaces Externas: ¿Cómo interactuará el software con el medio externo (gente, hardware, otro software)?
- Rendimiento: Velocidad, disponibilidad, tiempo de respuesta, etc.
- Atributos: Portabilidad, seguridad, mantenibilidad, eficiencia
- Restricciones de Diseño: Estándares requeridos, lenguaje, límite de recursos, etc.

Técnicas de Especificación de Requerimientos

- **Estáticas:** Se describe el sistema a través de las entidades u objetos, sus atributos y sus relaciones con otros. No describe como las relaciones cambian con el tiempo. Cuando el tiempo no es un factor mayor en la operación del sistema, es una descripción útil y adecuada. En este grupo tenemos Referencia indirecta, Relaciones de recurrencia, Definición axiomática, Expresiones regulares, Abstracciones de datos, entre otras.
- **Dinámicas:** Se considera un sistema en función de los cambios que ocurren a lo largo del tiempo. Se considera que el sistema está en un estado particular hasta que un estímulo lo obliga a cambiar su estado. Como ejemplo tenemos a las 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.

IR - Especificación de Requerimientos - Historias de Usuario

¿Qué es una Historia de usuario?

Una historia de usuario es una representación de un requisito de software escrito en una o dos frases utilizando el lenguaje común del usuario. Al momento de implementar las historias, los desarrolladores deben tener la posibilidad de discutirlas con los clientes. Permiten responder rápidamente a los requisitos cambiantes.

Debe ser limitada, ésta debería poder escribirse sobre una nota adhesiva pequeña. 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. Generalmente se espera que la estimación de tiempo de cada historia de usuario se sitúe entre unas 10 horas y un par de semanas, estimaciones mayores a dos semanas son indicativo de que la historia es muy compleja y debe ser dividida en varias historias.

Deben responder a tres preguntas:

- ¿Quién se beneficia? – Como (rol)
- ¿Qué se quiere? – quiero (algo)
- ¿Cuál es el beneficio? – para poder (beneficio)

Ventajas:

- Al ser muy corta, ésta representa requisitos del modelo de negocio que pueden implementarse rápidamente (días o semanas).
- Necesitan poco mantenimiento.
- Mantienen una relación cercana con el cliente.
- 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:

- 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.

Épicas:

Se denomina Épica a un conjunto de Historias de usuario que se agrupan por algún denominador común.

IR - Especificación de Requerimientos - Casos de Uso

¿Qué son los Casos de Uso?

Un caso es uso es un escenario de negocios o evento respecto del cual el sistema debe proporcionar una respuesta definida. Los casos de uso evolucionaron a partir del análisis orientado a objetos; pero su utilización se ha vuelto común en muchos otros métodos de análisis y diseño de sistemas.

Un modelo de casos de uso muestra una vista del sistema desde la perspectiva del usuario, por lo cual describe qué hace el sistema sin describir cómo lo hace. Proceso de modelado de las "funcionalidades" del sistema en término de los eventos que interactúan entre los usuarios y el sistema. El uso de CU facilita y alienta la participación de los usuarios.

Componentes:

- **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.

Ventajas:

- 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.

IR - Especificación de Requerimientos - Diagrama de Transición de Estados (DTE)

¿Qué es un diagrama de transición de estados?

Es una herramienta que se usa para ilustrar la secuencia y variación de pantallas que pueden ocurrir durante una sesión de usuario. Se usa para esquematizar la secuencia y las variaciones de pantallas que pueden generarse cuando el usuario del sistema se sienta ante la terminal.

Un estado identifica un periodo de tiempo de un objeto/entidad en el cual está esperando alguna operación. Una transición es el paso de un estado a otro. Se describe como el sistema responde a eventos internos o externos, muestra los estados del mismo y los eventos que provocan las transiciones de un estado a otro. No muestra el flujo de datos dentro de sistema. Un DTE supone que, en cualquier momento, el sistema está en uno de varios estados posibles y cuando recibe un estímulo dispara una transición a un estado diferente.

IR - Especificación de Requerimientos - Redes de Petri

¿Qué es una Red de Petri?

Una Red de Petri es una representación matemática o gráfica de un sistema a eventos discretos en el cual se puede describir la topología de un sistema distribuido, paralelo o concurrente. 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.

Las tareas concurrentes deben estar sincronizadas para permitir la comunicación entre ellas (pueden operar a distintas velocidades, deben prevenir la modificación de datos compartidos o condiciones de bloqueo). Pueden realizarse varias tareas en paralelo, pero son ejecutados en un orden impredecible. Éstas no son secuenciales.

IR - Especificación de Requerimientos - Tablas de decisión

¿Qué es una tabla de decisión?

Es una herramienta que permite presentar el sistema en función de posibles condiciones en un tiempo dado, reglas para reaccionar ante los estímulos que ocurren cuando se reúnen determinados conjuntos de condiciones y las acciones a ser tomadas como un resultado.

Es una herramienta que permite 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.

Describe el sistema como un conjunto de: Posibles **condiciones** satisfechas por el sistema en un momento dado, **reglas** para reaccionar ante los estímulos que ocurren cuando se reúnen determinados conjuntos de condiciones y **acciones** a ser tomadas como un resultado.

Para construir tablas de decisión, el analista necesita determinar el tamaño máximo de la tabla; eliminar cualquier situación imposible, inconsistencia o redundancia, y simplificar la tabla lo más que pueda.

Es esencial que verifique la integridad y precisión de sus tablas de decisión. Pueden ocurrir cuatro problemas principales al desarrollar tablas de decisión: que estén incompletas, que existan situaciones imposibles, contradicciones y redundancia.

¿Qué es el análisis estructurado?

El análisis estructurado es una actividad de construcción de modelos. Mediante una notación creamos modelos que representan el contenido y flujo de la información (datos y control). Es más amplio que una técnica de especificación de requerimientos.

El análisis estructurado no es método sencillo aplicado siempre de la misma forma por todos los que lo usan. Más bien, es una amalgama que ha evolucionado durante los últimos 30 años.

La técnica de análisis estructurado permite lograr una representación gráfica que permite lograr una comprensión más profunda del sistema a construir y comunicar a los usuarios lo comprendido. La notación no especifica aspectos físicos de implementación. Hace énfasis en el procesamiento o la transformación de datos conforme estos pasan por distintos procesos.

Diagrama de Flujo de Datos (DFD)

Es una herramienta que permite visualizar un sistema como una red de procesos funcionales, conectados entre sí por "conductos" y almacenamientos de datos. Representa la transformación de entradas a salidas y es también llamado diagrama de burbujas. Es una herramienta comúnmente utilizada por sistemas operacionales en los cuales las funciones del sistema son de gran importancia y son más complejas que los datos que éste maneja.

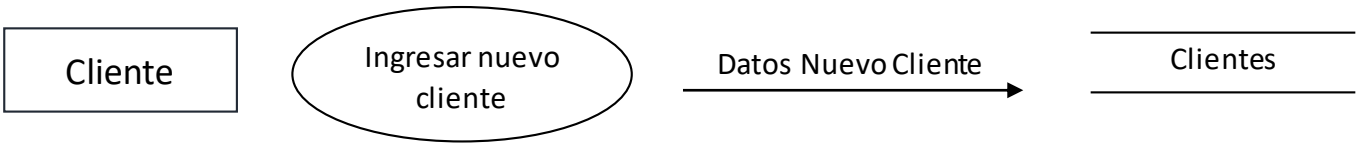
Elementos del DFD:

Se utiliza un rectángulo para representar una ***entidad externa,*** esto es, un elemento del sistema (por ejemplo, un elemento hardware, una persona, otro programa) 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).

Un rectángulo abierto (lado izquierdo y derecho) que representa un **almacén de datos**



Nivelación de un DFD:

Se debe visualizar desde una perspectiva jerárquica de arriba hacia abajo. Cada proceso se puede a su vez ampliar para crear un diagrama hijo más detallado. Las entradas y salidas del proceso padre permanecen, sin embargo, pueden aparecer nuevos almacenes de datos y nuevos flujos.

Diagrama de contexto:

Se muestra un panorama global que muestre las entradas básicas y las salidas. Es el nivel más alto en un DFD y contiene un solo proceso que representa a todo el sistema

Nivel 0:

Es la ampliación del Diagrama de contexto. Las entradas y salidas del Diagrama de contexto permanecen, sin embargo, se amplía para incluir hasta 9 procesos (como máximo) y mostrar los almacenes de datos y nuevos flujos.

Desarrollo de DFD:

1. Redactar la lista de actividades de la organización para determinar:
    - Entidades externas
    - Flujos de datos
    - Procesos
    - Almacenes de datos
2. Crear un diagrama de contexto que muestre las entidades externas y los flujos de datos desde y hacia el sistema.
3. Dibujar el Diagrama 0 (siguiente nivel), con procesos generales y los almacenes correspondientes
4. Dibujar un diagrama hijo por cada uno de los procesos del Diagrama 0

IR - Validación de requerimientos

¿Qué es la Validación de requerimientos?

Es el proceso de certificar la corrección del modelo de requerimientos contra las intenciones del usuario. Trata de mostrar que los requerimientos definidos son los que estipula el sistema. Se describe el ambiente en el que debe operar el sistema. Es importante, porque los errores en los requerimientos pueden conducir a grandes costos si se descubren más tarde. La validación sólo se puede hacer con la activa participación del usuario ya que se trata de mostrar que los requerimientos realmente definen el sistema que el cliente desea.

Comprenden:

- Verificaciones de validez (para todos los usuarios)
- Verificaciones de consistencia (sin contradicciones)
- Verificaciones de completitud (todos los requerimientos)
- Verificaciones de realismo (se pueden implementar)
- Verificabilidad (se puede diseñar conjunto de pruebas)

Técnicas de validación:

- Pueden ser manuales o automatizadas.
- Revisiones de requerimientos (formales o informales):
    - Informales: Los desarrolladores deben tratar los requerimientos con tantos stakeholders como sea posible.
    - Formal: El equipo de desarrollo debe conducir al cliente, explicándole las implicaciones de cada requerimiento

  Antes de una revisión formal, es conveniente realizar una revisión informal.
- Construcción de prototipos.
- Generación de casos de prueba.

Modelo de proceso de software

¿Qué es un proceso de software?

Es el conjunto de métodos, técnicas y prácticas que guían a los ingenieros de software en el desarrollo y evolución del software. El proceso de software es también llamado ciclo de vida del software, porque describe la vida de un producto de software desde su concepción hasta su implementación, entrega, utilización y mantenimiento. Un proceso de desarrollo de software debe describirse de manera flexible que permita a los que diseñan y construyen el software utilizar las herramientas y técnicas preferidas; el proceso en sí ayuda a mantener la consistencia y calidad en los productos que son producidos por muchas personas diferentes. El conjunto de procedimientos debe estar organizado de tal modo que los productos se construyan para satisfacer un conjunto de metas o estándares.

Actividades fundamentales en el proceso de software:

1. **Especificación del software:** Tienen que definirse tanto la funcionalidad del software como las restricciones de su operación.
2. **Diseño e implementación del software:** Debe desarrollarse el software para cumplir con las especificaciones.
3. **Validación del software:** Hay que validar el software para asegurarse de que cumple lo que el cliente quiere.
4. **Evolución del software:** El software tiene que evolucionar para satisfacer las necesidades cambiantes del cliente.

¿Qué es un modelo de proceso de software?

Es una representación abstracta 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. Existen varios modelos de proceso de software y cada uno representa una descripción, desde una perspectiva particular, de la manera en que el desarrollo del software se hace en la realidad. Los modelos de proceso también nos prescriben la manera en que sebe avanzar el desarrollo del software.

Características de un modelo de proceso de software:

- Establece todas las actividades.
- Utiliza recursos, está sujeto a restricciones y genera productos intermedios y finales.
- Puede estar compuesto por subprocesos.
- Cada actividad tiene entradas y salidas definidas.
- Las actividades se organizan en una secuencia.
- Existen principios que orientan sobre las metas de cada actividad.
- Las restricciones pueden aplicarse a una actividad, recurso o producto.

Términos equivalentes:

El **ciclo de vida del software** describe la vida del producto de software desde su concepción hasta su implementación, entrega, utilización y mantenimiento. El **modelo de proceso de software** es una representación abstracta de un proceso del software. Estos términos son equivalentes y refieren a lo mismo. También se los puede encontrar como **paradigma de software**.

Tipos de modelo de software:

- **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.
- **Modelos descriptivos:**
  Descripción en la forma en que se realizan en la realidad.

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:

- 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.

Fue el primer modelo que surgió, actualmente no hay control de etapas y cada instancia del desarrollo de software puede iterar entre las demás de manera indistinta.

Modelo en cascada con prototipo:

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.

Hay de dos tipos:

- **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.

Proyectos candidatos:

- Usuarios que no examinarán los modelos abstractos
- Usuarios que no determinarán sus requerimientos inicialmente
- Sistemas con énfasis en los formatos de E/S más que en los detalles algorítmicos
- Sistemas en los que haya que explorar aspectos técnicos
- Si el usuario tiene dificultad al tratar con los modelos gráficos para modelar los requerimientos y el comportamiento
- Si se enfatiza el aspecto de la interfaz humana

Para asegurar el éxito:

- Debe ser un sistema con el que se pueda experimentar
- Debe ser comparativamente barato (< 10%)
- Debe desarrollarse rápidamente
- Énfasis en la interfaz de usuario
- Equipo de desarrollo reducido
- Herramientas y lenguajes adecuados

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 (Boehm):

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.

Metodologías ágiles

¿Qué es una metodología ágil?

"Es un enfoque iterativo e incremental (evolutivo) de desarrollo de software". El desarrollo iterativo es una estrategia de reproceso en la que el tiempo se separa para revisar y mejorar partes del sistema. Es una estrategia programada y en etapas, en la que las diferentes partes del sistema se desarrollan en diferentes momentos o a diferentes velocidades, y se integran a medida que se completan. Básicamente divide el sistema en partes y los separa para revisar y mejorar y volver a ensamblar y revisar.

Una Metodología Ágil es aquella en la que "se da prioridad a las tareas que dan resultados directos y que reducen la burocracia tanto como sea posible", adaptándose además rápidamente al cambio de los proyectos.

Objetivos:

- 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.

Valores:

- **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.

Es importante comprender que aun cuando se deben valorar los conceptos que se encuentran del lado derecho, debemos valorar aún más aquellos que están a la izquierda. Una buena manera de interpretar el manifiesto, es asumir que éste define preferencias, no alternativas.

Principios:

1. Nuestra mayor prioridad es satisfacer al cliente a través de fáciles y continuas entregas de software valuable.
2. Los cambios de requerimientos son bienvenidos, aún tardíos, en el desarrollo. Los procesos Ágiles capturan los cambios para que el cliente obtenga ventajas competitivas.
3. Entregas frecuentes de software, desde un par de semanas a un par de meses, con el menor intervalo de tiempo posible entre una entrega y la siguiente.
4. Usuarios y desarrolladores deben trabajar juntos durante todo el proyecto.
5. Construir proyectos alrededor de motivaciones individuales.
6. Darles el ambiente y el soporte que ellos necesitan y confiar el trabajo dado. El diálogo cara a cara es el método más eficiente y efectivo de intercambiar información entre el equipo de desarrolladores.
7. El software que funciona es la medida clave de progreso.
8. Los procesos ágiles promueven un desarrollo sostenible. Los stakeholders, desarrolladores y usuarios deberían ser capaces de mantener un paso constante indefinidamente.
9. Atención continua a la excelencia técnica y buen diseño incrementa la agilidad.
10. Simplicidad (el arte de maximizar la cantidad de trabajo no dado) es esencial.
11. Las mejores arquitecturas, requerimientos y diseños surgen de la propia organización de los equipos.
12. A intervalos regulares, el equipo reflexiona sobre cómo volverse más efectivo, entonces afina y ajusta su comportamiento en consecuencia.

Diferencias:

Metodología ágil	**Metodología no ágil**
Pocos artefactos	Más artefactos
Pocos roles	Más roles
No existe un contrato tradicional o al menos es bastante flexible	Existe un contrato prefijado
El cliente es parte del equipo de desarrollo (además in-situ)	El cliente interactúa con el equipo de desarrollo mediante reuniones
Grupos pequeños (< 10 integrantes) y trabajando en el mismo sitio	Grupos grandes
Menos énfasis en la arquitectura	La arquitectura es esencial

Artefactos: todos los componentes que hay que desarrollar (documentación, entrega, etc.).

Desventajas:

En la práctica, los principios que subyacen a los métodos ágiles son a veces difíciles de cumplir:

- Aunque es atractiva la idea de involucrar al cliente en el proceso de desarrollo, los representantes del cliente están sujetos a otras presiones, y no intervienen por completo en el desarrollo del software.
- Priorizar los cambios podría ser difícil, sobre todo en sistemas donde existen muchos participantes. Cada uno por lo general ofrece diversas prioridades a diferentes cambios.
- Mantener la simplicidad requiere trabajo adicional. Bajo la presión de fechas de entrega, es posible que los miembros del equipo carezcan de tiempo para realizar las simplificaciones deseables al sistema.
- Muchas organizaciones, especialmente las grandes compañías, pasan años cambiando su cultura, de tal modo que los procesos se definan y continúen. Para ellas, resulta difícil moverse hacia un modelo de trabajo donde los procesos sean informales y estén definidos por equipos de desarrollo.
- Por lo general, el documento de requerimientos del software forma parte del contrato entre el cliente y el proveedor. Como en los métodos ágiles se minimiza la documentación, suele ser complejo reglamentarlo.
- La mayoría de los libros que describen los métodos ágiles y las experiencias con éstos hablan del uso de dichos métodos para el desarrollo de nuevos sistemas. Sin embargo, una enorme cantidad de esfuerzo en ingeniería de software se usa en el mantenimiento y la evolución de los sistemas de software existentes. Al no existir documentación se complejizaría.

Metodologías ágiles - eXtreme Programming

¿Qué es 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:

- Desarrollo iterativo e incremental: pequeñas mejoras, unas tras otras.
- Pruebas unitarias continuas, frecuentemente repetidas y automatizadas, incluyendo pruebas de regresión.
- Programación en parejas.
- Frecuente integración del equipo de programación con el cliente o usuario.
- Corrección de todos los errores antes de añadir nueva funcionalidad.
- Refactorización del código.
- Propiedad del código compartida.
- Simplicidad del código.

Características esenciales:

- Historias de usuario
- Roles
- Proceso
- Prácticas

Roles:

- **Programador (Programmer)**
  - Responsable de decisiones técnicas
  - Responsable de construir el sistema
  - Sin distinción entre analistas, diseñadores o codificadores
  - En XP, los programadores diseñan, programan y realizan las pruebas
- **Jefe de Proyecto (Manager)**
  - Organiza y guía las reuniones
  - Asegura condiciones adecuadas para el proyecto
- **Cliente (Customer)**
  - Es parte del equipo
  - Determina qué construir y cuándo
  - Establece las pruebas funcionales
- **Entrenador (Coach)**
  - Responsable del proceso
  - Tiende a estar en un segundo plano a medida que el equipo madura
- **Encargado de Pruebas (Tester)**
  - Ayuda al cliente con las pruebas funcionales
  - Se asegura de que las pruebas funcionales se superan
- **Rastreador (Tracker)**
  - Metric Man
  - Observa sin molestar
  - Conserva datos históricos

Procesos (ciclo de vida):

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.

Prácticas

- **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.

Metodologías ágiles - Scrum

¿Qué es 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:

- **Eliminar el desperdicio:** no generar artefactos, ni perder el tiempo haciendo cosas que no le suman valor al cliente.
- **Construir la calidad con el producto:** la idea es inyectar la calidad directamente en el código desde el inicio.
- **Crear conocimiento:** En la práctica no se puede tener el conocimiento antes de empezar el desarrollo.
- **Diferir las decisiones:** tomar las decisiones en el momento adecuado, esperar hasta ese momento, ya que uno tiene más información a medida que va pasando el tiempo. Si se puede esperar, mejor.
- **Entregar rápido:** Debe ser una de las ventajas competitivas más importantes.
- **Respetar a las personas:** la gente trabaja mejor cuando se encuentra en un ambiente que la motive y se sienta respetada.
- **Optimizar el todo:** optimizar todo el proceso, ya que el proceso es una unidad, y para lograr tener éxito y avanzar, hay que tratarlo como tal.

Roles:

- El **Product Owner (Propietario)** conoce y marca las prioridades del proyecto o producto.
- El **Scrum Master (Jefe)** es la persona que asegura el seguimiento de la metodología guiando las reuniones y ayudando al equipo ante cualquier problema que pueda aparecer. Su responsabilidad es entre otras, la de hacer de paraguas ante las presiones externas.
- El **Scrum Team (Equipo)** son las personas responsables de implementar la funcionalidad o funcionalidades elegidas por el Product Owner.
- Los **Usuarios o Cliente**, son los beneficiarios finales del producto, y son quienes viendo los progresos, pueden aportar ideas, sugerencias o necesidades.

Artefactos:

- **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:

- 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.

¿Cuándo usar Scrum?

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.

Desarrollo de Software Basado en Modelos (MBD)

¿Qué es el 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)

¿Qué es el 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.

Model Driven Development (MDD) promueve enfatizar los siguientes puntos claves:

- 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.

Beneficios de MDD:

- Incremento en la productividad (modelos y transformaciones).
- Adaptación a los cambios tecnológicos.
- Adaptación a los cambios de requisitos.
- Consistencia (automatización).
- Re-uso (de modelos y transformaciones).
- Mejoras en la comunicación con los usuarios y la comunicación entre los desarrolladores (los modelos permanecen actualizados).
- Captura de la experiencia (cambio de experto).
- Los modelos son productos de larga duración (resisten cambios).
- Posibilidad de demorar decisiones tecnológicas.

Calidad

¿Qué es Calidad?

Propiedad o conjunto de propiedades inherentes a algo, que permiten juzgar su valor. Esta es totalmente subjetiva porque depende del juicio de quien la evalúa.

Las principales normas internacionales definen la calidad como:

- "El grado en el que un conjunto de características inherentes cumple con los requisitos" (ISO 9000)
- **"Conjunto de propiedades o características de un producto o servicio que le confieren aptitud para satisfacer unas necesidades expresadas o implícitas" (ISO 8402)**

Criterios erróneos comunes sobre la calidad:

- Un producto de calidad es un producto de lujo.
- La calidad es intangible y por lo tanto no mensurable.
- Los problemas son originados por los trabajadores de producción.
- La calidad se origina en el Depto. de calidad.

Calidad de los Sistemas de Información:

La importancia de los sistemas de información (SI) en la actualidad hace necesario que las empresas de tecnología hagan mucho hincapié en los estándares (o normas) de calidad. Stylianou y Kumar plantean que se debe apreciar la calidad desde un todo, donde cada parte que la componen debe tener su análisis de calidad.

Componentes:

- Calidad de la empresa:
  - Calidad de los procesos de negocio soportados por SI.
  - Calidad del SI:
    - Calidad de la infraestructura: incluye, por ejemplo, la calidad de las redes, y sistemas de software
    - Calidad del software: de las aplicaciones de software construidas, o mantenidas, o con el apoyo de IS.
    - Calidad de la información: está relacionada con la calidad de los datos.
    - Calidad de los datos: Que ingresan en el sistema de información.
    - Calidad del servicio: incluye los procesos de atención al cliente
    - Calidad de la gestión: incluye el presupuesto , planificación y programación

Calidad de Software

Esta calidad se divide entre la calidad del producto obtenido y la calidad del proceso de desarrollo. Estos son dependientes el uno del otro, no puedo tener uno sin tener el otro.

Calidad del Producto y 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.

Normas y modelos de calidad:

Calidad de producto de software	Se evalúa la calidad mediante	ISO/IEC 25000	Está compuesto por distintos modelos. Define características que pueden estar presentes o no en el producto. La norma nos permite evaluar si están presentes o no, y de qué manera evaluarlas. EJ: Seguridad, Compatibilidad, Seguridad. Etc.
Calidad de proceso de desarrollo	Se evalúa la calidad mediante	ISO/IEC 12207	ISO/IEC 12207 establece un modelo de procesos para el ciclo de vida del software. Define cómo debería ser el modelo de proceso para ser completo y con calidad. Actividades, tareas etc.
		ISO/IEC 15504	Es una norma internacional para establecer y mejorar la capacidad y madurez de los procesos Define que se debe tener en cuenta para evaluar el modelo de proceso y concluir si es completo y con calidad.
		ISO/IEC 90003	Proporciona una guía sobre cómo aplicar la ISO 9001 en procesos de software
		CMMI	Proporciona un marco estructurado para evaluar los procesos actuales de la organización, establecer prioridades de mejora, e implementar esas mejoras. Se utiliza para organizaciones desarrolladoras de software de medianas a grandes dimensiones
Calidad de Procesos/Servicios/Productos en general	se evalúa mediante	ISO 9001	La Norma ISO 9001 determina los requisitos para establecer un Sistema de Gestión de la Calidad, de producto y/o servicio. Forma parte de la familia ISO 9000, que 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.