Ingeniería de software 1

Introducción

Software: intrucciones, procedimientos, documentación y datos asociados que forman un sistema de computación. Es un elemento lógico el cual se desarrolla y no se gasta. Cada vez más sistemas se contruyen a partir de un producto genérico como base y luego se adapta según el dominio

A partir de nuevas tecnologías y necesidades de los clientes nace la necesidad de la ingeniería de software

Ingeniería de software: disciplica que comprende aspectos de la produción de software desde las etapas iniciales hasta que se comience a ejecutar.

Es una disciplica porque usa teorías, métodos y herramientas. Y no solo comprende los procesos técnicos si no también la gestión de proyectos y uso de herramientas de apoyo

> Según **IEEE**, la IS es el uso de métodos sistemáticos, disciplinados y cuantificales para el desarrollo, operación y mantenimiento de software (la ingeniería de SW se ocupa de todo el ciclo de vida del producto, desde la etapa de planificación y análisis de requerimientos hasta la estrategia para determinar cuándo y cómo debe ser retirado de servicio

Ingenieros de software

Los ingenieros deben conocer las tecnologías y productos (sistemas operativa, lenguajes de programación, bases de datos, etc.), conocer las técnicas de administración de proyectos (planificación, análisis de riesgo, control de calidad, seguimiento de proyectos, control de subcontratistas, etc.)

Además tienen cierta **responsabilidad profesional y ética** ya que se desarrolla en un marco económico, social y legal. Por lo tanto, no deben usar su capacida y habilidades de forma deshonesta

- Confidencialidad: respetar la confidencialidad de empleados y clientes
- Competencia: no falsificar el nivel de competencia y aceptar resposabilidades fuera de su capacidad
- Derechos de la propiedad intelectual: conocer las leyes de patentes y copyright
- Uso inapropiado de las computadoras: no usar habilidades técnicas para usar inapropiadamente las computadora

Comunicación

Es la base para obtener las necesidades del cliente y donde más errores hay

> Requerimiento: característica del sistema o descripción de algo que el sistema es capaz de hacer con el objetivo de satisfacer el propósito del sistema

Fuentes de requerimientos

- Documentación

- Stakeholders: cualquier persona/ grupo que se verá afectado por el sistema, directa o indirectamente. Pueden ser: usuarios finales, ingenierós, etc.
- Especificación de sistemas similares

Puntos de vista

- Interactuadores: personas o sistemas que interactúan directamente con el sistema
- Indirecto: stakeholders que no usan el sistema ellos mismos pero influyen en los requerimientos de alguna manera
- Dominio: características y restricciones del dominio que influyen en los requerimientos del sistema

Elicitación de requerimientos

Proceso de adquirir todo el conocimiento relevante para producir un modelo de los requerimientos de un dominio de problema con el objetivo de conocer el dominio del problema y así poder comunicarse con el cliente y entender sus necesidades, conocer el sistema actual e identificar las necesidades

En el proceso de elicitación de requrimientos surgen **problemas de comunicación** los cuales son

Dificultad para expresar claramente las necesidades	No ser conscientes de sus propias necesidades	No entender cómo la tecnología puede ayudar	Diferente cultura y vocabulario	Intereses distintos en el sistema a desarrollar

Medios de comunicación inadecuados

Conducta humana

- Conflictos y ambigüedades en los roles de los participantes

Limitaciones cognitivas

- No conocer dominio del problema
- Hacer suposiciones
- Hacer simplificaciones excesivamente

- Técnicos
  - Complejidad del dominio del problema
  - Complejidad de los requisitos
  - Múltiples fuentes de requisitos
  - Fuentes de información poco claras

Impacto de los errores en la etapa de requerimientos

El software resultante puede no satisfacer a los usuarios, las interpretaciones múltiples de los requerimientos puede causar desacuerdos entre los clientes y desarrolladores, pérdida de tiempo construyendo el sistema erróneo

Clasificación de requerimientos

- Funcionales: describen lo que el sistema debe hacer o no. Es una interacción entre el sistema y su ambiente. Son independientes de la implementación
- No funcionales: describen una restricción sobre el sistema que limita nuestras elecciones en la construcción de una solución al problema
  - Del producto: especifican comportamiento del producto

    - Organizacionales: derivados de las políticas y procedimientos existentes en la organización
    - Externos: interoperabilidad, levales, privacidad, seguridad, éticos
- Del dominio: características y restricciones del dominio de la aplicación del sistema
- Por prioridad: deben ser absolutamente satisfechos, deseables pero no indispensables
- Del usuario: declaraciones en lenguaje natural y en diagramas de los servicios que se espera que el sistema provea
- Del sistema: establecen con detalle los servicios y restricciones del sistema

> La ingeniería de requerimientos es el proceso por el cual se transforman los requerimientos declarados por los clientes a especificaciones precisas, no ambiguas, consistentes y completas del comportamiento del sistema

La ingeniería de requerimientos resulta importante porque permite gestionar las necesidades del proyecto estructuradamente, mejora la capacidad del cronograma del proyecto, aumenta la calidad del software y la comunicación entre equipos

Hay diferentes formas de recopilar información y es mediante

- **Métodos discretos:** menos perturbadores que los métodos interactivos pero resultan **insuficientes para recopilar información cuando se usan por sí solos**.
    - Muestreo de la documentación, formularios y datos existenes se recolectan datos a partir de la documentación existente y que permite conocer el historial que origina el proyecto. Los documentos pueden ser organigramas, notas internas, registros contables, solicitudes de proyectos de sistemas anteriores
    - Investigación y visitas al lugar: el analista se convierte en observador de las personas y actividades con el objetivo de aprender sobre el sistema

      Previamente a la observación se debe determinar quién y cuándo será observado, obtener permiso de las personas necesarias, tomar nota de lo visto y no interrumpir a las personas en su trabajo

      Ventajas:

        Datos confiables

        El analista ve exactamente lo que se hace

        Análisis de disposiciones físicas, tránsito, ilumincación, etc

        Es económica

      Desventajas:

        La gente se siente incómoca al ser observada

        Algunas actividades pueden ser hechas en horarios incómodos

        Tareas sujetas a interrupciones
    - observación del ambiente de trabajo

- **Métodos interactivos:** tienen como base hablar con las personas de la organización y escuchar para comprender.
  - Cuestionarios: documento para recolectar información y opiniones de los encuestados

    Resultan ser de repuesta rápida, económicas y anónimos pero se otiene un número bajo de respuertas, no responde todas las preguntas y no se pueden aclarar respuestas incompletas

    Tengo preguntas abiertas o cerradas

    Abiertas: permiten posibles opciones de respuesta

    Cerradas: cierran las opciones de respuestas disponibles

    A partir de los cuestionarios obtengo información sobre actitud, creencias, comportamiento y características de personas o casos

    Debo usar cuestionarios cuando las personas están dispersas geográficamente, muchos involucrados, queremos opiniones generales o queremos reconocer problemas generales
  - Entrevistas: el analista de sistemas recolecta información a través de la interación cara a cara. Es una conversación con un propósito específico basado en preguntas y respuestas el cual nos permitirá conocer opiniones y sentimientos

    Con las entrevistas la persona se siente incluído en el proyecto, obtener una retroalimentación, adaptar las preguntas según la persona y puedo obtener información no verbal. En su contra, son costosas, conllevan tiempo y RRHH y no se pueden aplicar a la distancia

    Las entrevistas pueden ser estructuradas o no estructuradas

    Estructuradas (cerradas): el encuestador tiene un conjunto específico de preguntas para hacérselas al entrevistado y se dirige al usuario para un requerimiento puntual

    No estructuradas (abiertas): el encuestador lleva a un tema en general, no hay una preparación previa y se inicia con preguntas que no dependen del contexto

    Las preguntas pueden ser abiertas, cerradas o por sondeo

    - Las *abiertas* permiten al encuestado responder de cualquier manera. Revelan una nueva línea de preguntas haciendo la entrevista más interesante y habrá una espontaneidad. En contra tienen que pueden dar detalles irrelevantes, puedo perder el control de la entrevisa y parece que el entrevistador no tiene objetivos claros
    - Las *cerradas* tienen respuestas directas, cortas o de selección específica. Sus pro es que ahorran tiempo, hay un control más fácil de la entrevista y consigo datos relevantes pero podría aburrir al entrevistado y no se dan muchos detalles

    Las entrevistas pueden ser piramidades, embudo o diamante

    Preparación previa a la entrevista según Kendall

    Deberé leer atecedentes teniendo en cuenta el vocabulario en común para poder entender al entrevistado, establecer objetivos de la entrevista, seleccionar los entrevistados (minimizando el número de entrevistados, quienes deben conocer el objetivo de la entrevista y las preguntas), planificar la entrevista (establecer fecha, hora, lugar y duración) y finalmente seleccioar el tipo de preguntas y la estructura

    Según Whitten

    1) *Seleccionar al entrevistado* teniendo en cuenta el requerimiento a analizar, citarlo, establecer la duración de la entrevista

2) *Preparar la entrevista* informando al entrevistado del tema a tratar, definir el guión de entrevista evitando preguntas con intención, amenazantes o críticas, usando lenguaje claro y conciso

3) *Conducir la entrevista* respetando el horario y tiempo, iniciarla respetuosamente, mencionar el objetivo, escuchar con atención, finalizar la entrevista agradeciendo

El entrevistado debe ser cortés, mantener el control, observar gestos, ser paciente y terminar a tiempo. Y debe evitar suponer que una respuesta no lleva a ningún lado, usar jerga, hablar en vez de escuchar, usar grabadores

- Planeación conjunta de requerimientos: conducción de reuniones de grupo con el objetivo de analizar problemas y definir requerimientos. Se caracteriza por requerir extenso entrenamiento, reduce el tiempo de exploración de requisitos, gran participación de los integrantes

  Su ventaja es que se ahorra tiempo, los usuarios se involucran y hay desarrollo cretivo pero es difícil coordinar por los horarios laborales

- Brainstorming: genera ideas para alentar a los participantes para que ofrezcan tantas ideas como sea posible en un corto tiempo. Se promueve el desarrollo de ideas creativas

> Cuantas más ideas se sugieren, mejores resultados se conseguirán. La producción de ideas en grupos puede ser más efectiva que la individual. Las ideas de una persona pueden hacer que aparezcan otras por “contagio A veces las mejores ideas aparecen tarde. Es mejor elegir sobre una variedad de soluciones.

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

Especificación de requerimientos

Tiene como objetivo permitir que los desarrolladores expliquen cómo han entendido lo que el cliente espera del sistema, indica a los diseñadores las funcionalidades y características e indica al equipo de prueba las demostraciones que deberán ser llevadas a cambo para convencer que el sistema hace lo que se espera

La especificación debe ser correcta, no ambigua, completa, verificable, modificable, organizada y concisa

La especificación está formada por

- Documento de definición de requerimientos: listado de todas las cosas que el cliente espera que haga el sistema
- Documento de especificación de requerimientos: definición en términos técnicos

Validación de requerimientos

Proceso de certificar la corrección del modelo de requerimientos tratando se mostrar que los requerimientos definidos son los que estipula el sistema

Boehm indica que cuanto más tarde se detecten los errores, más costarán corregirlos, hacer la validación en la etapa de especificación de requerimientos ayudará a evitar correcciones costosas.

La validación de requerimientos valida que no haya contradicciones, complete todos los requerimientos, puedan implementarse y sea posible diseñar un conjunto de pruebas

Las técnicas de validación pueden ser manuales o automatizadas, las cuales algunas son

- Revisiones de requerimientos, las cuales pueden clasificarse en informales o formales. En las informales los desarrolladores deben tratar los requeriientos con tantos stakeholders como sea posible, en cambio en las formales el equipo de desarrollo conduce al cliente explicándole las implicaciones de cada requerimiento
- Construcción de prototipos
- Generación de casos de prueba

Técnicas de especificación de requerimientos

/* agregar contenido de las explicaciones prácticas */

Estáticas

Describe el sistema a través de las entidades u objetos, sus atributos y sus relaciones con los otros. No describe cómo las relaciones cambian con el tiempo

> ***Historias de usuario:*** representación de un requisito de software escrito en una o dos frases usando el lenguaje común del usuario
>
> Son una forma rápida de administrar los requisitos de los usuarios sin tener que elabrar gran cantidad de documentos formales y sin requerir de mucho tiempo, pueden discutirse con los clientes y según estimaciones de tiempo deben requerir de entre 10 horas y un par de semanas. En caso de que una historia requiera más de dos semanas, sería necesario dividirla en varias historias
>
> El esquema de las HU es *Como (rol) quiero (algo) para poder (beneficio)*
>
> Las HU se caracterizan por ser independientes unas de otras, negociables, valoradas por los usuarios, estimables, pequeñas y verificables
>
> Los **criterios de aceptación** es el criterio por el cual se define si una historia de usuario fue desarrollada según lo esperado por el Product Manager y si se puede hacer. Los criterios de aceptación son los detalles de una historia de usuario, y ayudan a completar el patrón de valor expresado por la misma
>
> En resuen, representan requisitos del modelo de negocio, llevan poco mantenimiento, dan una relación cercana con el cliente y permite dividir el proyecto en pequeñas entregas. Hay que tener en cuenta que si no hay criterios de aceptación, la HU puede quedar abierta a distintas interpretaciones, se requiere contacto permanente con el cliente y puede ser difícil para grandes proyectos

> ***Casos de uso:*** proeso de modelado de las funcionalidades en término de los eventos que interactúan entre el usuario y el sistema. Se caracteriza por capturar requerimientos funcionales, descompone al

sistema en piezas manejables, define una línea base para definir los planes de prueba y proporciona una herramienta para seguir los requisitos

Los CU se componen por

- Diagrama de CU: ilustra las interacciones entre el sistema y los actores. El diagrama se compone por
  - CU: representa una funcionalidad del sistema y describe la secuencia de actividades
  - Actores: representa un papel desempeñado por un usuario que interactúa
  - Relaciones: asociaciones, extends, uses, depends, herencia
    - Asociaciones: relación entre un actor y un CU
    - Extends: un CU extiende la funcionalidad de otro CU
    - Uses: reduce la redundancia entre dos o más CU al combinar los pasos comúnes de los CU
    - Depends: relación entre CU que indica que un CU no puede realizarse hasta que se haya realizado otro CU
    - Herencia: relación entre actores donde un actor hereda funcioanlidades de uno o más actores
- Escenarios: descripción de la interacción entre el actor y el sistema para realizar la funcionalidad

En el proceso de modelado de los CU debo identificar los actores, identificar los CU para los requerimientos, construír el diagrama y realizar los escenarios

Dinámicas

Se considera un sistema en función de los cambios que ocurren a lo largo del tiempo, considera que el sistema está en un estado particular hasta que un estímulo lo obliga a cambiar su estado

Diagrama de transición de estados (DTE): describen al sistema como un conjunto de estados donde el sistema reacciona a ciertos eventos posibles

Para la construcción de un DTE debo identificar los estados, desde el estado inicial identificar los cambios de estado con flechas, analizar las condiciones y las acciones para pasar de un estado a otro y finalmente verificar la consistencia

Redes de Petri: usadas para especificar sistemas de tiempo real en los que son necesarios representar aspectos de concurrencia lo cual permite la ejecución simultánea de componentes. Las tareas concurrentes deben estar sincronizadas para permitir la comunicación entre ellas

Tablas de decisión: herramienta que permite presentar de forma concisa las reglas lógicas que hay que usar para decidir acciones a ejecutar según las condiciones de un problema.

Análisis estructurado

Actividad de construcción de modelos el cual debe lograr describir lo que requiere el cliente, estableer una base para la creación de un diseño de software y definir un conjunto de requisitos que se pueda

validad una vez que se construye el software. Está compuesto por

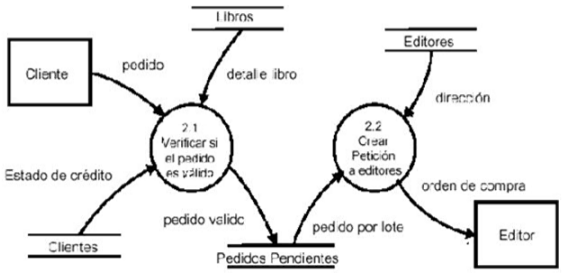
- Diagrama de flujo de datos (DFD): herramienta para visualizar un sistema como una red de procesos funcionales. Herramienta usada por sistemas en los cuales las funciones del sistema son de gran importancia y son más complejas que los datos que éste maneja

  Entidad externa: representada por un rectángulo y es un elemento del sistema que produce información para ser transformada por el software

  Proceso: aplicado a los datos los cuales los modifica

  Flecha: representa uno o más elementos de datos

- Diccionario de datos: listado organizado de todos los datos pertinentes al sistema



Propuesta de Yourdon

- Modelo escencial: debe indicarse lo que el sistema debe hacer para satirfacer los requerimientos del usuario con una mínima explicación de cómo lo hace. Evitar el detalle de cualquier restricción o aspecto derivado de la implementación
  - Modelo ambiental: define las interfaces entre el sistema y el ambiente donde el mismo se ejecuta. Es lo primero y más importante en el modelo de requerimientos, luego de ser construido hay que chequearlo con los usuarios clave y con el grupo de análisis. El modelo ambiental se compone por la declaración de propósitos, el diagrama de contexto y la lista de acontecimientos
  - Modelo de comportamiento: representación del comportamiento final que el sistema debe tener para manejar con éxito el ambiente, dentro de las especificaciones requeridas por el usuario (DFD, DER, DD, DTE).

    El modelo preliminar de comportamientos contiene un diagrama preliminar de flujo de datos del sistema (DFD), diagrama preliminar de entidad-relación (DER), primer versión del diccionario de datos (DD) y diagrama de estados (DTE)

Modelos de proceso

Proceso de software es un conjunto de actividades y resultados asociados que producen un producto de software

Modelo de proceso de software presenta una visión de ese proceso. Se caracteriza por establecer todas las actividades, usar recursos, cada actividad tiene E/S definidas, las actividades se organizan en

secuencia

- Ciclo de vida: proceso que implica la construcción de un producto
- Ciclo de vida del software: describe la vida del producto de software desde su concepción hasta su implementación, entrega, uso y matenimiento
- Modelo de proceso de software: representación abstracta de un proceso de software

Modelo de proceso, paradigma de software y ciclo de vida del software son términos equivalentes

Los modelos pueden ser

- Prescriptivos: prescriben un conjunto de elementos del proceso como actividades del marco de trabajo, acciones de la IS, tareas, calidad y mecanismos de control
- Descriptivos: descripción en la forma en que se realizan en la realidad

Ambos modelos deberían ser iguales

Los modelos posibles son

- En cascada: las etapas se representan cayendo en cascada y cada una se debe completar antes que comience la siguiente. Dentro de sus contras está que no hay resultados concretos hasta que se haya terminado todo, eliminar fallas es complicado
- En V: demuestra cómo se relacionan las actividades de prueba con las de análisis y diseño /*buscar */
- De prototipos: 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
    - Evolutivo: el objetivo es obtener el sistema a entregar permitiendo que todo el sistema o alguna de sus partes se construyan rápidamente para comprender o aclarar aspector y asegurar que los involucrados tenganuna comprensión unificada de lo que se necesita y espera
    - Descartables: no tiene funcionalidad
- Por fases (incremental/ iterativo)
    - Incremental: el sistema es particionado en subsistemas de acuerdo con su funcionalidad y en cada entrega se agrega un subsistema
    - Iterativo: entrega un sistema completo desde el principio y aumenta la funcionalidad en nuevas versiones
- Espiral (Boehm): Las actividades de este modelo consistes en la confirmación en un espiral, en la que cada bucle representa un conjunto de actividades. Las actividades no están fijadas a ninguna prioridad, sino que las siguientes se eligen en función del análisis de riesgo, comenzando por el bucle interior

Metodologías ágiles

Es un enfoque iterativo y evolutivo de desarrollo para hacer frente a proyectos medianos y grandes. Tienen como objetivo produceir software de alta calidad con costo y tiempo apropiado, con valores que

permitan al equipo desarrollar software rápidamente y respondiendo a los cambios que puedan llegar a surgir y ofrecen una alternativa a los procesos de desarrollo tradicionales

Valores

- Individios e interacciones más que procesos y harramientas
- Software operante más que documentaciones completas
- Colaboración con el cliente más que negociaciones contractuales
- Respuesta al cambio más que apegarse a una rigurosa planificación

Principios

- Satisfacer al cliente con entregas de software valuables
- Los cambios de requerimientos son bienvenidos
- Entregas frecuentes de software
- Los usuarios y desarrolladores deben trabajar juntos durante todo el proyecto
- Simplicidad
- Mejorar arquitectura, requerimientos y diseños

Desventajas

- Aunque es atractivo involucrar al cliente en el proceso de desarrollo, los clientes están sujetos a otras presioes
- Priorizar los cambios podría ser difícil, sobre todo cuando hay muchos participantes
- Mantener la simplicidad requiere trabajo adicional
- Muchas organizaciones 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

Las principales metodologías ágiles son *XP (extreme programming)*, *Scum, DSDM (dynamic systems development method), Crystal Methods (cockburn's crystal family methodologies), ASD (adaptative software development), FDD (feature-driven development)*

eXtreme Programming

Metodología basada en la sencillex, comunicación, retroalimentación, valentía y respeto. Lleva a todo el equipo reunido para ver dónde están y ajustar las práticas a su situación particular.

Se caracteriza por usar

- HU
- Roles
  - Programador: hace decisiones técnicas, contruye el sistema, en XP los devs diseñan, programan y testean
  - Jefe de proyecto: organiza y guía las reuniones y asegura las condiciones adecuadas para el proyecto
  - Cliente: determina qué construir y cuándo
  - Entrenador: responsable del proceso y pasa a segundo plano cuando el equipo mejora

    - Encargado de pruebas: ayuda al cliente con pruebas funcionales y se asegura de que las pruebas funcionales se superen
    - Rastreador: observa sin molestar y conserva datos históricos
- Proceso
    1. Exploración: son planeadas las HU, se comienza a familiarizar con las herramientas y tecnologías a utilizarse y se contruye un prototipo. Esta fase dura pocas semanas o meses, según el tamaño del dominio
    2. Planificación: se establece la prioridad de las HU, se estima el esfuerzo y se acuerda el cronograma de entrega
    3. Iteración: el plan de entrega no tiene una duración de más de 3 semanas y el cliente es quien define qué HU se implementan en cada iteración
    4. Producción: se hacen pruebas y revciones
    5. Mantenimiento
    6. Muerte: el cliente no tiene más HU para incluir, se genera la documentación final del sistema y no habrá más cambios en la arquitectura, su muerte también puede ocurrir cuando no haya más beneficios o el cliente no puede mantenerlo
- Prácticas
    - Testing: los devs hacen pruebas unitarias que deben funionar sin problema
    - Refactoring: actividad de reestructuración de código para remover duplicación, más legibilidad y simplicidad
    - Programación de a pares: el código es escrito por dos programadores en una máquina
    - Propiedad colectiva del código: cualquiera puede cambiar el código, con el objetivo de contribuir nuevas ideas
    - Integración continua: cada pieza de código se integra en el sistema una vez que esté lista. Así el sistema puede integrarse y construirse en un mismo día
    - Cliente en el lugar de desarrollo: debe estar presente y disponible para el equipo
    - Estándares de codificación: los programadores escriben el código según reglas

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 sume valor
- Construir la calidad con el producto: inyectar la calidad directamente en el código inicial
- Crear conocimiento
- Diferir las decisiones: tomar decisiones en el momento adecuado ya que uno tiene más información a medida que va pasando el tiempo
- Entregar rápido para tener mayor ventaja competitiva

- Respetar a las personas
- Optimiza todo

Roles

- Product owner: conoce y marca prioridades del proyecto
- Scrum master: asegura el seguimiento de la metodología guiando reuniones y ayuda al equipo ante cualquier problema
- Scrum team: implementa fuincionalidad del product owner
- Usuarios: beneficiarios finales del producto, pueden aportar ideas o sugerencias

Artefactos

- Product backlog: lista que tiene la funcionalidad deseada en el producto y se ordena por prioridad
- Sprint backlog: funcionalidad que el equipo se compremete a entregar en un *sprint* determinado
- Burndown chart: acumula el trabajo hecho día a día

Scrum resulta ser iterativo e incremental donde se busca atacar todos los problemas que surgan en el desarrollo

Se recomienda usar esta metodología cuando en un proyecto hay caos constante o requerimientos dinámicos

Desarrollo basado en modelos (MBD)

Intermedio entre las metofologías tradicionales y las ágiles. Apunta a que la construción de un sistema de software debe ser precedida por la construcción de un modeo.

Un modelo del sistema consiste en una conceptualizacion del dominio del problema y actúa como una especificación precisa de los requerimientos que el sistema de software debe satisfacer

Desarrollo dirigido por modelos (MDD)

dirigido enfatiza que asigna a los modelos un rol central y activo y es tan importante como el código fuente. MDD promueve enfatizar en

- Mayor nivel de abstracción en la especificación tanto del problema a resolver como de la solución correspondiente
- Aumentar la confianza en la automatización asistida por computadora
- Uso de entándares industriales como medio para facilitar las comunicaciones, interacción entre diferentes aplicaciones y productos

Los modelos pasan de ser entidades complementarias a productivas a partir de las cuales se deriva la implementación

    - PIM (platform independent model): modelo de un sistema que no contiene información sobre la plataforma que se usa para implementarlo
    - PSM (platform specific model): modelo de sistema que incluye información acerca de la tecnología que se usará

Tranformación: Colección de reglas las cuales son no ambiguas de las formas en que un modelo puede ser usado para crear otro modelo. El patrón MDD se usa sucesivas veces para producir unna sucesión de transformaciones

Los beneficios son: incremento de productividad, adaptación a cambios de tecnología y de requisitos, consistencia, re-uso, de larga duración

Calidad

Calidad: conjunto de propiedades inherentes a una cosa que permite caracterizarla y valorarla con respecto a las restantes de su especie

Calidad de los sistemas de información: la importania de los sistemas de información en la actualidad hace necesario que las empresas de tecnología hagan hincapié en los estándares de calidad

Componentes

- Calidad de la empresa
  - Calidad de los procesos de negocio soportados por SI
    - Calidad de la infraestrutura: como calidad de redes y sistemas de software
    - Calidad del software: aplicaciones de software construidas o mantenidas
    - Calidad de la información: relacionada a la calidad de los datos
    - Calidad de datos que ingresan en el sistema de información
    - Del servicio: procesos de atención al cliente
    - De la gestión: presupuesto, planificación y programación

---

Calidad de software: se divide entre la calidad del producto obtenido y la calidad del proceso de desarrollo

Calidad del producto y proceso

- Producto: la estandarización del producto define las propiedades que debe satisfacer el producto de software resultante
- Proceso: la estandarización del proceso defie la manera de desarrollar el producto de software

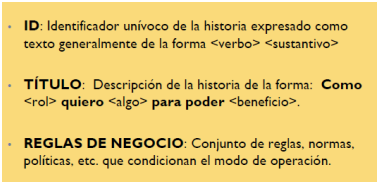
Sin un buen proceso de desarrollo es casi imposile obtener un buen producto

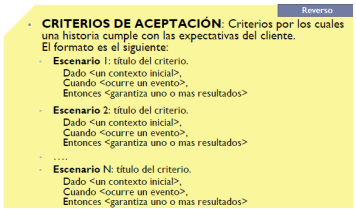
Calidad de producto de software	Se evalúa la calidad mediante	ISO/IEC 25000	Está compuesto por distintos modelos. Define características que pueden estar presentes o no en el producto. La norma nos permite evaluar si están presentes o no, y de qué manera evaluarlas. EJ: Seguridad, Compatibilidad, Seguridad. 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.

Explicaciones prácticas

Historias de usuario

Descripción corta y simple de un requerimiento de un sistema, que se redacta en lenguaje común del usuario y desde su perspectiva. Usado en XP y SCRUM





Casos de uso

Proceso de modelado del problema en término de los eventos que interactúan entre los usuarios y el sistema.

En el proceso de modelado debo identificar actores y CU, contruir el diagrama y realizar los escenarios

DTE (Diagrama de transición de estados)

Diagramas que permiten modelar el comportamiento del sistema en función del tiempo. Sirve para sistemas de tiempo real, modelado de procesos o sistemas de control

Elementos

- Estado: identifica un periodo de tiempo de un objeto/entidad en el cual el sistema está esperando alguna operación o realizando alguna acción
- Transición: relacionan estados. Tienen una única dirección y está conformado por 3 partes
  - Condición: impide que el sistema cambie de estado al darse un evento
  - Evento: suceso que provoca que el sistema cambie de estado
  - Acción: una o más tareas instantáneas que hace el sistema durante la transición de un estado al otro

Convenciones

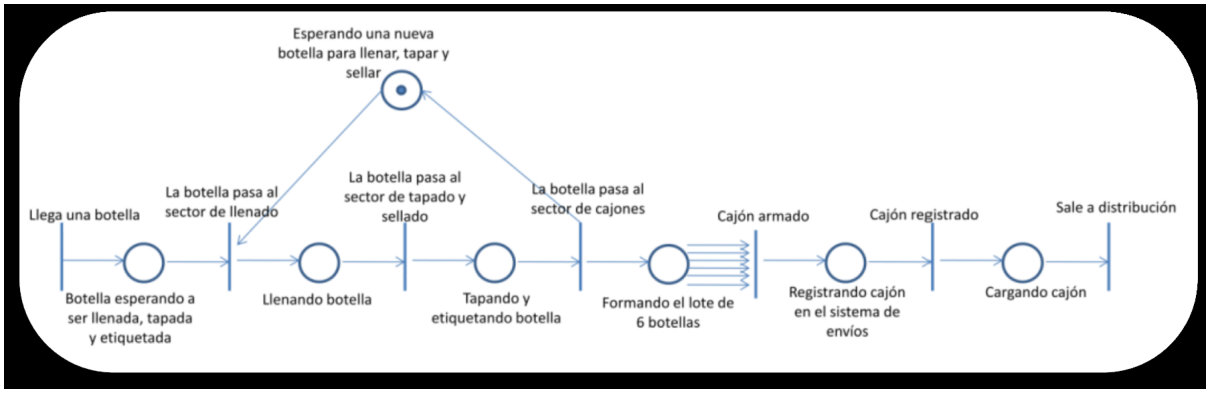
- Nombre de los estados: verbos en gerundio (ando/ endo)
- Eventos: manifiestan la ocurrencia de un estímulo que conlleva la salida del estado y tiene forma impersonal
- Acción: verbo en infinitivo con sustantivo en función del sistema

A la hora de construit el modelo es importante tener en cuenta que el modelado se realiza desde el punto de vista del sistema y NO desde el del usuario. Para contruirlo se recomienta

1. Identificar los estados del sistema
2. Desde el estado inicial identificar los cambios que lo llevan de un estado a otro mediante transiciones
3. Analizar los eventos, condiciones y las acciones para pasar de un estado a otro
4. Verificar consistencia: definir todos los estados, si alcanzo todos los estados o puedo salir de ellos y en cada estado el sistema responde a todas las condiciones posibles

Redes de Petri

Permiten modelar sistemas dinámicos y concurrentes mediante una representación gráfica de eventos discretos. La red es un grafo dirigido con 3 componentes: sitios, transiciones y arcos

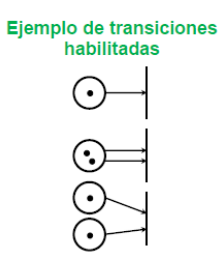


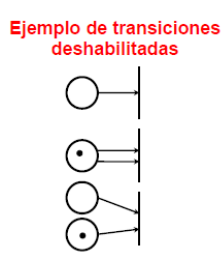
Componentes

- Sitio: modela un estado o condición
- Transiciones: modela un evento o una acción
- Arco: relaciona un sitio con una transición o una transición con un sitio
- Token: habilita/ deshabilita

Funcionamiento

- Transicón habilitada: se considera habilitada cuando al manos hay un token por cada arco que llega a la transición
- Propagación de tokens: cuando una transición se encuentra habilitada, en un instante de tiempo se absorberá tantos token como arcos llegan y producirá tantos tokens como arcos salen en el instante





Convenciones

- Convención de inicio: para indicar que se pueden generar una cantidad ilimitada de tokens se usa una transición sin entradas
- No bloquear la red: toda transicón debe tener oportunidad de ser habilitada alguna vez
- Nombres obligatorios y expresados en el diagrama: todos los estados y transiciones deben tener nombres distintos. Además las transiciones pueden llamarse según la etapa
- Convención de fin: una transición sin lugares de salida elimina tokens de la red de Petri

Tablas de decisión

Es una representación de lógicas de decisión compleja. Describe al sistema como un conjuto de posibles condiciones satisfechas por el sistema a un tiempo dado y son reglas para reaccionar ante los estímulos que ocurren cuando se reúnen determinados conjuntos de condiciones y acciones a ser tomadas como resultado

Puntos a tener en cuenta

- Cantidad de reglas: 2N, donde N es la cantidad de condiciones encontradas
- Cada regla posee una combinación de valores de verdad
- No pueden quedar reglas sin acciones marcadas
- Las condiciones que se asumen como precondiciones desde el enunciado no van en la tabla. Por ejemplo, si se analizan las condiciones para un sistema de administración de clientes, “es cliente” no debe modelarse como condición, ya que se asume que todos son clientes
- Las reducciones se realizar entre 2 reglas que comparten las mismas acciones y difieren en 1 solo valor de sus condiciones
- Las reglas inconsistentes deben eliminarse