[Introducción al Análisis de Sistemas 4](#_Toc466047079)

[Necesidad del Análisis y Diseño. 4](#_Toc466047080)

[Rol del Analista de Sistemas. 4](#_Toc466047081)

[Conceptos 4](#_Toc466047082)

[Teoría general de sistemas 4](#_Toc466047083)

[Formas de determinar un sistema 4](#_Toc466047084)

[Tipos de Sistemas 5](#_Toc466047085)

[Catalogación de Sistemas. 5](#_Toc466047086)

[Descripción de Sistemas 5](#_Toc466047087)

[Concepto de seguridad informática). 5](#_Toc466047088)

[Ciclos de Vida 6](#_Toc466047089)

[Etapas. 6](#_Toc466047090)

[Tipos de ciclos de vida. 6](#_Toc466047091)

[Concepto de proyecto 8](#_Toc466047092)

[Etapas de un Proyecto 8](#_Toc466047093)

[Introducción a la Ingeniería de Requisitos 8](#_Toc466047094)

[Proceso de adquisición de conocimientos. 8](#_Toc466047095)

[Ingeniería de Requisitos (IR) 9](#_Toc466047096)

[Proceso de la Ingeniería de Requisitos 9](#_Toc466047097)

[Elicitación 9](#_Toc466047098)

[Modelización 9](#_Toc466047099)

[Análisis. 9](#_Toc466047100)

[Gestión. 9](#_Toc466047101)

[Técnicas para la extracción de conocimientos (Fuentes NO humanas) (Gomez 1997) 10](#_Toc466047102)

[Técnicas para la educción de conocimientos (Fuentes Humanas) (Gomez 1997) 10](#_Toc466047103)

[Entrevista 10](#_Toc466047104)

[Cuestionario - Encuesta 10](#_Toc466047105)

[Joint Aplication Development. 10](#_Toc466047106)

[Brainstorming. 10](#_Toc466047107)

[Ciclo de Educción 11](#_Toc466047108)

[Tipo de Preguntas para Educcion (Entrevista, Cuestionatrio, Encuesta) 11](#_Toc466047109)

[Preguntas Cerradas 11](#_Toc466047110)

[Preguntas Abiertas 11](#_Toc466047111)

[Diferencias entre Joint Aplication Development y Brainstorming 11](#_Toc466047112)

[Joint Aplication Development 11](#_Toc466047113)

[Brainstorming 11](#_Toc466047114)

[Modelos de Dominio 11](#_Toc466047115)

[Concepto de modelo de dominio, Clases conceptuales e Identificación 11](#_Toc466047116)

[Requisitos Funcionales y no funcionales. 12](#_Toc466047117)

[Tipos de Requisitos No funcionales 12](#_Toc466047118)

[Propiedades de Requisitos (Hadad, 2008) 12](#_Toc466047119)

[Especificación de requisitos (IEEE 830-1998). 12](#_Toc466047120)

[Metodología Estructurada 13](#_Toc466047121)

[Modelo Ambiental 13](#_Toc466047122)

[Declaración de propósito (DP) 13](#_Toc466047123)

[Lista de acontecimientos (LC) 13](#_Toc466047124)

[Diagrama de Contexto (DC) 13](#_Toc466047125)

[Modelo de Comportamiento 13](#_Toc466047126)

[Diagrama de Flujo de Datos (DFD) 13](#_Toc466047127)

[Diagrama de Entidad Relación (DER) 13](#_Toc466047128)

[Diccionario de Datos (DD) 13](#_Toc466047129)

[Especificaciones de Procesos (EP) 13](#_Toc466047130)

[Diagrama de Transición de Estados (DTD) 13](#_Toc466047131)

[Permite especificar la varianza en el tiempo de una entidad, una relación, etc. 13](#_Toc466047132)

[Metodología orientada a objetos 14](#_Toc466047133)

[Clase 14](#_Toc466047134)

[Relaciones entre clases 14](#_Toc466047135)

[Pilares de Programación Orientada a Objetos 14](#_Toc466047136)

[Metodologías Agiles 14](#_Toc466047137)

[Practicas Agiles 15](#_Toc466047138)

[Proceso Ágil 15](#_Toc466047139)

[Comparativa con entre Métodos Agiles SCRUM, XP, DSDM 15](#_Toc466047140)

[Características XP, SCRUM, DSDM 15](#_Toc466047141)

[Roles XP, SCRUM, DSDM 15](#_Toc466047142)

[Ciclo de Vida XP, SCRUM, DSDM 15](#_Toc466047143)

[Scrum 16](#_Toc466047144)

[Historias de usuario (XP) 17](#_Toc466047145)

[Porque Usarlas 17](#_Toc466047146)

[Características 17](#_Toc466047147)

[Tarjeta de Historia de Usuario 17](#_Toc466047148)

[Casos de uso. 18](#_Toc466047149)

[Actor 18](#_Toc466047150)

[Características 18](#_Toc466047151)

[Metodología de Casos de Uso (MCU) 18](#_Toc466047152)

[Interfaz Tentativa 18](#_Toc466047153)

[Especificaciones 19](#_Toc466047154)

[Comparación entre Casos de uso e Historias de usuario (XP) 19](#_Toc466047155)

[UML Lenguaje Unificado de Modelado 20](#_Toc466047156)

[UML. Componentes del lenguaje. Distintos diagramas que se utilizan en las diferentes etapas de la construcción de un sistema software. Vista estática y dinámica. 20](#_Toc466047157)

[Cuales son utilizados en la etapa de Análisis OO 20](#_Toc466047158)

[Proceso Unificado 21](#_Toc466047159)

[Caracteristicas 21](#_Toc466047160)

[Ciclo de Vida 21](#_Toc466047161)

[Características: centrado en la arquitectura, desarrollo iterativo e incremental y dirigido por Casos de uso. 21](#_Toc466047162)

[Cuáles son las fases y disciplinas que componen el proceso unificado. 21](#_Toc466047163)

[Que actividades se realizan en cada una de ellas. 21](#_Toc466047164)

[Cuáles son los roles y los artefactos producidos 21](#_Toc466047165)

[Modelo de análisis 22](#_Toc466047166)

[Diagrama de clases de análisis 22](#_Toc466047167)

[Diagrama de comunicación de análisis 22](#_Toc466047168)

[Descripción de las clases con los atributos 22](#_Toc466047169)

[Propios y vinculantes. 22](#_Toc466047170)

[Otros diagramas de UML 22](#_Toc466047171)

[Diagrama de actividades. Diagrama de estados 22](#_Toc466047172)

[Preguntas Cerradas 22](#_Toc466047173)

[Preguntas Abiertas 22](#_Toc466047174)

Introducción al Análisis de Sistemas

Necesidad del Análisis y Diseño.

Analisis y Diseño de Sistemas

* Sirven para analizar sistemáticamente la entrada o el flujo de datos, procesar o transformar datos, el almacenamiento de datos y la salida de información en el contexto de una empresa en particular
* Son procesos sistemáticamente emprendidos con el propósito de mejorar un negocio con ayuda de sistemas de información computarizados

Rol del Analista de Sistemas.

|  |  |
| --- | --- |
| **Características**   * Debe ser autodiciplnado y automotivado para administrar y coordinar diversos recursos e los proyectos que afronte * Es en esencia una persona curiosa y un solucionador * Dada a la comunicación * Tiene gran empatía para ponerse del lado del cliente | **Consultor**: Agente externo a la organización   * + Los consultores externos tienen una perspectiva fresca de la cual carecen los demás miembros de una organización. * - Alguien externo nunca conocerá la verdadera cultura organizacional. |
| **Experto en Soporte**: Agente interno experto en el tema   * Es a quien se le consulta o es al que se lo envía para generar las Historias de Usuario * Tiene un conocimiento de Hardware/Software y experiencia de negocio fundamental * Es un recurso de los que están a cargo de Proyecto | **Agente de Cambio**: Agente interno experto que por naturaleza motiva al cambio   * El de mayor responsabilidad * Desempeña actividades en el ciclo de vida del software * Su presencia en el negocio es la de iniciar los cambios, catalizarlos y cooperar con ellos * Es el que entiende con profundidad el negocio para saber cómo realizar el cambio * Encargado de diseñar el plan para el cambio * Es el encargado de enseñar, controlar, motivar el cambio. |

Conceptos

**Sistema**: Conjunto de elementos interrelacionados entre si con un fin en común de modo tal que el todo frente a su finalidad es mucho mas que la suma de sus partes.

**Sistema de información**: Conjunto de elementos que interactúan entre si con el fin de apoyar las actividades de una empresa o negocio

**Sistema informático**: Conjunto de elementos que interactúan entre sí con el fin de apoyar una actividad donde su procedimiento es automatico.

**Dominio**: Área de aplicación sobre la que deseamos construir nuestro sistema

**Tensión Interna**: Son las preocupaciones dentro del espacio del problema que el cliente solicita resolver.

**Tensión Externa**: Son las preocupaciones fuera del espacio del problema que debe resolver el analista.

**Función**: Objetivo o fin que el sistema percibe y es razón de ser desde el punto de vista de la función principal del problema.

**Límite**: Frontera de un sistema, determina el “desde donde” y el “hasta donde” es el alcance de las funciones del sustema

**Alcance**: Es la descripción clara de las funcionalidades que abarca el sistema y de aquellas que escluye si son relevantes a destacar.

Teoría general de sistemas

**Objetivo**: Realizar formulaciones conceptuales de fácil entendimiento, impulsar terminologías para tener un lenguaje común y definir un conjunto de leyes y reglamentos para pensar de manera sistémica

**Comportamiento sistémico**: Cada elemento del sistema debe tener un objetivo dentro de la continuidad el mismo.

**Premisas**: Todo sistema existe dentro de otro sistema, todos son abiertos de algún modo por su interrelación con el entorno y de misma manera todos sufren perdidas

**Metas**: Integrar las ciencias naturales con las sociales, generando principios unificadores y de integración

**Análisis de Sistemas**: Es la acción de resolver un problema y/o oportunidad de lograr un negocio a veces.

**Entropía**: Mide el nivel de desorden de un sistema a través del desgaste producido por el paso del tiempo o la utilidad que este ofrece

**Homeostasis**: Mide el nivel de adaptación al cambio de un sistema.

Formas de determinar un sistema

|  |  |  |
| --- | --- | --- |
| **Etapas de un Sistema Genérico** | **Kendall y Kendall** | **Pressman** |
| Análisis | Determinación de problemas, oportunidades y objetivos | Comunicación |
| Determinación de los requerimientos humanos de información | Planificación |
| Análisis de las necesidades del sistema |
| Diseño | Diseño del sistema recomendado | Modelado |
| Desarrollo y documentación del software | Construcción |
| Implementación | Prueba y mantenimiento del sistema | Construcción |
| Implementación y evolución del sistema |

Tipos de Sistemas

Catalogación de Sistemas.

|  |  |
| --- | --- |
| * Los **ESS (Sistema de apoyo a ejecutivos)** se encuentran en el nivel estratégico de la administración.   Ejemplo: Planeamiento Operativo, de Presupuesto, de planificación de personal   * Los **GDSS** y **CSCWS** se auxilian la toma de decisiones semiestructuradas o no estructuradas a nivel de grupo. * Estos tres últimos soportan el nivel estratégico, o más alto, apoyando la toma de decisiones semiestructuradas o las no estructuradas. * Los **MIS (Sistema de información Gerencial)** y **DSS (Sistema de apoyo a decisiones)** Sistemas de alto nivel   Ejemplo Análisis de regiones de ventas, Análisis de costos, Análisis de precios, Análisis de rentabilidad   * Los **OAS (Sistema de automatización de la oficina)** y **KWS (Sistemas de trabajo del conocimiento)** apoyan el trabajo al nivel del conocimiento.   Ejemplo: Procesamiento de Texto, Imágenes, Agendas   * Los sistemas expertos aplican el conocimiento de los encargados de la toma de decisiones para problemas estructurados específicos. * Los **TPS (Sistema de procesamiento de Transacciones)** funcionan al nivel operativo de una organización.   Ejemplo: Ventas, Marketing, Producción, Finanzas, Contabilidad, Recursos humanos |  |

Descripción de Sistemas

|  |  |
| --- | --- |
| Sistema de Procesamiento de Transacciones **TPS**   * Sistemas de información computarizados. * Creados para procesar grandes cantidades de datos relacionadas con transacciones rutinarias de negocios, como las nóminas y los inventarios. * Los usuarios aún tienen que capturar datos. * Es importante que estos sistemas funcionen sin ningún tipo de interrupción. * Más rápidos que los no-computarizados. | Sistema de Apoyo a la Toma de Decisiones **DSS**   * Depende de una base de datos para abastecerse de datos. * Pone énfasis en el apoyo a la toma de decisiones en todas sus fases. * La decisión definitiva es responsabilidad exclusiva del encargado. * Se ajustan más al gusto de la persona o grupo que los utiliza. |
| Sistemas de Información Gerencial **MIS**   * Sistemas de información computarizados. * Incluyen al TPS. * Su propósito es contribuir a la correcta interacción entre los usuarios y las computadoras. * Dan apoyo a un espectro de tareas organizacionales como el análisis y la toma de decisiones. * Los usuarios comparten una base de datos en común. * Producen información que se emplea en la toma de decisiones. | Sistema de Automatización de la Oficina **OAS**   * Apoyan a los trabajadores de datos. * Por lo general no generan nuevos conocimientos. * Analizan la información con el propósito de transformar los datos o manipularlos de alguna manera antes de compartirlos/distribuirlos formalmente con el resto de la organización. * Componentes más comunes: Procesamiento de texto, hojas de cálculo, comunicaciones mediante correo de voz, email, videoconferencia.   Sistema de Trabajo del Conocimiento **KWS**   * Sirven de apoyo a los trabajadores profesionales en sus esfuerzos de creación de nuevo conocimiento. |
| Sistemas Expertos e Inteligencia Artificial **SE, IA**   * Utilizan las técnicas de razonamiento de la Inteligencia Artificial. * Captura y utiliza el conocimiento de un experto para solucionar un problema especifico en una organización. * Selecciona la mejor solución para un problema o una clase especifica de problemas. * Componentes básicos: base de conocimientos, motor de inferencia (Ejemplo: SQL) y la interfaz del usuario. * Los ingenieros de conocimiento capturan la pericia de los expertos, construyen un sistema de cómputo que contiene este conocimiento experto y lo implementan. | Sistemas de Apoyo a la Toma de Decisiones en Grupo **GDSS**   * Se utilizan en salones especiales equipados con diversas configuraciones. * Faculta a los miembros del grupo a interactuar con apoyo electrónico y la asistencia de un facilitador especial. * Tienen el propósito de unir a un grupo en la búsqueda de la solución a un problema con la ayuda de diversas herramientas como los sondeos, los cuestionarios, brainstorms y la creación de escenarios.   Sistemas de Trabajo Colaborativo Apoyados por Computadora **CSCWS**   * Además de lo anterior tienen software específico para colaborar en red |
| Sistema de Apoyo a Ejecutivos **ESS**   * Ayudan a los ejecutivos a organizar sus actividades relacionadas con el entorno externo mediante herramientas gráficas y de comunicaciones. * Ayudan a los usuarios a resolver problemas de toma de decisiones no estructuradas. * Amplían y apoyan las capacidades de los ejecutivos al darles la posibilidad de comprender sus entornos. | |

Concepto de seguridad informática).

* **Confidencialidad**: Solo el responsable de la información puede tener acceso a la misma
* **Integridad**: La información que se envía debe ser consistente a su llegada
* **Disponibilidad de los datos**: La información debe ser brindada de manera que quien tenga los permisos de acceso pueda acceder a la misma

Ciclos de Vida

Es el proceso que se sigue para construir, entregar y hacer evolucionar el software, desde la concepción de una idea hasta la entrega y el retiro del sistema. Representa todas las actividades y artefactos (productos intermedios) necesarios para desarrollar una aplicación.

* Determinar el orden de las fases del Proceso Software.
* Establecer los criterios de transición para pasar de una fase a la siguiente.

Se elige en base a la cultura de la corporación, el deseo de asumir riesgos, el área de aplicación la volatilidad de los requisitos y hasta qué punto se entienden bien dichos requisitos.

**Proceso software**: Conjunto completo de actividades de Ingeniería de Software necesarias para transformar los requerimientos del usuario en software.

Etapas.

* Todo ciclo de vida contempla mínimamente las etapas de **Requerimientos, Diseño, Implementación, Pruebas y Mantenimiento** las cuales pueden ser distribuidas, mas ponderadas, repetidas, unificadas con otras o conectadas de manera no secuenciales dependiendo de las diferentes implementaciones de ciclos de vida.

Tipos de ciclos de vida.

|  |  |
| --- | --- |
| Ciclo de Vida en Cascada (Royce 1970)  Procede a través de una secuencia ordenada de transiciones de una fase a la siguiente según un orden lineal permitiendo iterar entre fases y en cada fase. Útil para ayudar a estructurar y gestionar grandes proyectos de desarrollo La mayor iteración se produce una vez terminado el desarrollo y cuando se ha visto el software producido, se decide comenzar de nuevo y redefinir los requisitos del usuario.  **Características**   * Identifica entregables e hitos. * Ciclo de vida corto y económico. * Al ser secuencial desaprovecha RRHH * Espera requerimientos definidos completamente al inicio del proyecto. * Obliga a especificar lo que el sistema debe hacer antes de construir el sistema. * Obliga a definir cómo van a interactuar los componentes antes de construirlos. * Permite al jefe de proyecto seguir y controlar los progresos * Requiere que el proceso de desarrollo genere documentos para utilizarse en la validación y el mantenimiento del sistema * El éxito depende de que todos los estados sean desarrollados y desarrollar en orden erróneo produce producto de inferior calidad * Se congelan los requisitos al principio del proyecto. * Sirve para problemas no complejos, bien definidos y bien comprendidos. * Las etapas están organizadas de un modo lógico. * Cada etapa incluye cierto proceso de revisión, y se necesita una aceptación del producto antes de que la salida de la etapa pueda usarse. * De fácil entendimiento e implementación * Ampliamente utilizado y conocido. * Dificultad para integrar administración del riesgo.   **Se lo utiliza cuando….**   * Definición del cliente es buena. * Comprensión del grupo de desarrollo buena. * Volatilidad baja. * Experiencia alta en el dominio y en las técnicas. * Entregas parciales no necesarias. * No se tenga en cuenta el riesgo. * Sistema no complejo.   **Problemas que genera**   * Envía al cliente el primer producto solamente después de que se han consumido el 99% de los recursos para el desarrollo. * Hacer cambios es costoso y difícil. | Ciclo de Vida Incremental (Hirsch 1985)  El desarrollo incremental es el proceso de construir una implementación parcial del sistema global y posteriormente ir aumentando la funcionalidad del sistema. Reduce la posibilidad de que las necesidades del usuario cambien durante el desarrollo.  **Características**   * El Software se construye de modo sólo satisfaga unos pocos requisitos de todos los que tiene el usuario (generalmente las funcionalidades principales), y que permita la incorporación de los restantes. * Se reduce el tiempo de desarrollo inicial. * Es más fácil de mejorar el software. * El punto crítico es la primera entrega * Descompone el desarrollo de un sistema en partes (subsistemas) * A cada parte se le aplica un modelo de desarrollo. * Se realizan entregas al finalizar cada parte. * No se le agregan funcionalidades, pero si puede modificarse algo aún no hecho, pero no se pueden modificar los mini sistemas, ósea los requisitos iniciales. * Volatilidad media * Aprovecha los RRHH. * El equipo de desarrollo no necesita tener una gran experiencia en el dominio de la aplicación. * El usuario puede ir mejorando las definiciones de requisitos a lo largo del proyecto, pero no puede agregarlas. * No tiene en cuenta el hardware, al tomar porciones puede ser que no tenga en cuenta la arquitectura total y cuando lo lleve a gran escala no sirva lo que se hizo al principio.   **Se lo utiliza cuando….**   * Definición del cliente es regular. * Compresión del grupo de desarrollo es regular. * Volatilidad media. * Experiencia en el dominio y en las técnicas media. * Las entregas deben ser parciales. * Sistema complejo, se puede comprender mejor al dividirlo en sub sistemas. * Cuando no se tiene en cuenta el riesgo.   **Problemas que genera**   * Pérdidas de tiempo en la primera entrega en el prototipo funcional que a veces es inútil * La arquitectura a largo plazo puede ser invalida e inviable según los requisitos se van cumpliendo * La primera entrega esta tan alejada de la realidad que puede hacer al cliente desecharnos |

|  |  |
| --- | --- |
| Ciclo de Vida Prototipado (Brooks 1975)  Se utiliza cuando el cliente no sabe bien lo que quiere, entonces hay que armarle una maqueta o prototipo para presentarle. También cuando no se está seguro de a viabilidad, tiene como objetivo contrarrestar el problema de la congelación de requisitos mal comprendidos.  **Tipos**   * **Maqueta**: Puede ser en papel. Aporta al usuario ejemplo visual de entradas y salidas, datos estáticos. Se realiza para mejorar la definición o el entendimiento y después se elige otro ciclo de vida. Sirve para problemas simples. * **Prototipo desechable**: Tiene algo de funcionalidad para que el cliente vea su funcionamiento y luego se desecha y se elige otro ciclo de vida. Se usa para ayudar al cliente a identificar los requisitos de un nuevo sistema, se implantan sólo aquellos aspectos del sistema que se entienden mal o son desconocidos.   Pretende asegurar que el software que se está proponiendo cumple realmente las necesidades del usuario.  El usuario proporciona retroalimentación, esta se usa para modificar la especificación de requisitos de modo que refleje las verdaderas necesidades del usuario.   * **Prototipo evolutivo**: Ciclo de vida en sí mismo, fácilmente modificable y ampliable, aporta a los usuarios una representación física de las partes claves del sistema antes de la implantación, comienza con un prototipo con funcionalidad, se le muestra al cliente y de acuerdo a eso las modificaciones van evolucionando, evolucionarán hacia el sistema final.   Los desarrolladores construyen una implementación parcial que satisface los requisitos conocidos.  No se conocen desde el principio todos los requisitos.  Los requisitos de fiabilidad y rendimiento se implementan desde el principio.  Se centra en funcionalidad y contenido.  Extensión del desarrollo incremental.  Se pone en la evolución, de un modo más continuo, hacia una solución.  Cada sucesivo prototipo explora una nueva área de necesidades de usuario, la solución Software evoluciona acercándose cada vez más a las necesidades del usuario.  Diseñado para ser mucho más adaptativo.  Es largo y costoso  Hay que controlar los cambios para que el sistema no sea eterno.  Aprovecha RRHH.  El cliente ve prontamente una parte del sistema.  Permite volatilidad alta (cambio y agregado de funcionalidades) hasta el alcance del sistema. No agregar cosas que serían por si mismas otro sistema.  **Se lo utiliza cuando….**   * Definición del cliente es mala a regular. * Comprensión del grupo de desarrollo mala a regular. * Volatilidad alta. * Experiencia en el dominio y técnicas regular. * Entregas parciales tempranas. * Validación de requerimientos del cliente. (Mediante prototipo) * Sistema complejo. * No tiene en cuenta riesgos. * No deben estar definidos los requerimientos totalmente en el comienzo del ciclo de vida. Se van a hacer tantos prototipos como sean necesarios para comprender bien los requisitos.   **Problemas**   * El problema del prototipo es la elección de las funciones que se desean incorporar, y cuáles son las que hay que dejar fuera. * En la construcción de un prototipo hay incluido un ciclo de vida en cascada, porque hay que analizar, diseñar, codificar, probar e instalar el prototipo. * Cuando se acaba la fase de análisis del prototipo, se refinan los requisitos del software y se procede al comienzo del desarrollo a escala real. | Ciclo de Vida en Espiral (Boehm 1988)  Representa un enfoque dirigido por el riesgo para el análisis y estructuración del proceso software, se lleva a cabo representando ciclos de desarrollo iterativo en forma de espiral. Durante las primeras iteraciones, la versión incremental podría ser un modelo en papel o un prototipo. Durante las últimas iteraciones, se producen versiones cada vez más completas del sistema diseñado.   * Permite la reutilización * Se centra en la eliminación de errores y alternativas poco atractivas. * No establece una diferenciación entre desarrollo de software del sistema. * Tiene en cuenta los riesgos durante todo el desarrollo del proyecto. * Es largo y costoso, más que el prototipado evolutivo. * Permite observar mejor todos los elementos del proceso. * Es una mejor representación de los modelos de ciclo de vida, no un ciclo de vida en sí mismo. * Proporciona un marco estable   **Etapas**   1. Comunicación con el cliente, produce una especificación del producto. 2. Planificación: Tareas requeridas para definir recursos, el tiempo y otra información relacionada con el proyecto. Cada paso por esta área produce ajustes en el plan. 3. Análisis de riesgos: Se evalúan situaciones que pueden llevar al fracaso al proyecto. 4. Ingeniería: Tareas requeridas para construir una o más representaciones de la aplicación. Análisis y diseño de la solución. 5. Construcción y acción/adaptación: Tareas requeridas para construir, probar, instalar y proporcionar soporte al usuario (Por ejemplo: Documentación y práctica) 6. Evaluación del cliente Feedback para mejorar siguiente espira de proceso.     **Se lo utiliza cuando….**   * Sistema complejo. * Se tiene en cuenta los riesgos. * Validar las definiciones de los requerimientos con el cliente. * Entregas tempranas. * No es necesaria la definición de todas las necesidades al comienzo. * Definición del cliente regular o media. * Entendimiento del grupo de desarrollo regular o malo. * Volatilidad media o alta. * Experiencia en el dominio y las técnicas regular (como mínimo) |

Concepto de proyecto

|  |  |
| --- | --- |
| Proyecto es aquel proceso que puede definir claramente el alcance, costo y tiempo brindándome la calidad necesaria para que sea viable su realización   * Tiene un inicio y un final * Objetivo a cumplir (siempre es más extenso que la duración del proyecto) * Temporalidad (tiene un tiempo de gestión-duración) * Alcance * Costo * Tiempo de realización * Esta bajo un entorno de calidad |  |

Etapas de un Proyecto

|  |  |
| --- | --- |
| Factibilidad (Pressman)  Es un estudio entre muchos proyectos para dar, con los posibles a realizar. Este estudio debe ser rápido y nos debe brindar cuales podremos realizar y cuales descartar. Se debe definir los Objetivos de cada uno, para luego definir los recursos necesarios y por último la evaluación de los mismos. Los tipos son:   * Factibilidad Técnica   Tecnología disponible para poder llevarlo a cabo  Sera una adiciona sistemas actuales o no   * Factibilidad Económica   El tiempo de análisis del analista es posible  Costo del estudio del sistema  Costo del tiempo de los empleados en ese estudio  Costo estimado de Hardware  Costo del desarrollo del paquete Software   * Factibilidad Operacional   Si el sistema funcionara al ser instalado  Si el sistema será usado | Planificación  Consiste en etapas de la realización del proyecto:   * El nivel de entregables * Diseño de los prototipos necesarios * Módulos que garanticen el funcionamiento * Integración con otros sistemas * Se desarrollan las entrevistas para la educción de requisitos |
| Ejecución   * Se desarrolla lo planificado, según lo definido en el dominio del proyecto y las formas de cumplir con los objetivos establecidos |
| Cierre   * Se da cuando no se cumplen los plazos o requisitos funcionales * Se establecen lecciones aprendidas con cada proyecto * La experiencia en los proyectos se define en sus métricas |

Introducción a la Ingeniería de Requisitos

Proceso de adquisición de conocimientos.

|  |  |
| --- | --- |
| La **Adquisición de Conocimientos**, es el proceso de recolección de información necesaria para construir un sistema. Como tal no es un paso concreto del desarrollo de un sistema sino un proceso en paralelo con las demás actividades (Identificación, Conceptualización, Formalización, Validación, Mantenimiento). En cada etapa este proceso proporciona la información requerida. Tanto en la conceptualización como en el mantenimiento la cantidad de información adquirida aumenta en la conceptualización, baja en la formalización, pasa desapercibida en la implementación y aumenta en el mantenimiento por el feedback con el cliente/s. Los métodos de obtención de información son por vía escrita (Extracción) o vía oral con personas (Educción) y el proceso de adquisición. Este proceso comienza en un grano grueso y se convierte progresivamente en grano fino. | **Proceso**   * Primeras reuniones y evaluación de viabilidad (de expertos, directivos y usuarios). * Extracción de conocimientos (de la documentación). * Educción de conocimientos (del experto).   Interrogatorio inicial.  Investigación profunda |
| **Etapas**   * **En las primeras reuniones se buscan conocimientos generales**, las perspectivas en estas reuniones son el entorno de la tarea y sus usuarios, la profundidad es mínima. Se busca el grano grueso del problema. * **El siguiente paso es el estudio de la documentación existente.** Tiene como objetivo reducir el tiempo que deberá malgastar el experto a fin de iniciar al analista en el tema mientras se van definiendo los requisitos funcionales genéricos. Así terminado el segundo paso se tiene una visión genérica del tema a tratar * **El tercer paso es el Ciclo de Educción de Conocimientos.** La educción es el proceso de interactuar con un experto humano con el propósito de construir un sistema. Se divide en un interrogatorio inicial y la investigación profunda. En el primero se tratará una visión de alto nivel del dominio, donde el analista llegue a comprender el alcance del dominio, cual es la tarea del experto y el entorno de la tarea obligando al experto a mantener homogeneidad de la profundidad del modelo de dominio | **Ciclo de adquisición recomendado debe tener:**   1. Entendimiento general de la tarea. 2. División de las funcionalidades del sistema. 3. Análisis de la tarea (Pasos en la resolución de problemas) 4. Proceso de razonamiento del experto. 5. Datos de entrada necesarios para resolver el problema. 6. Desarrollo de un modelo conceptual.   *Para 1) y 2) es necesario el proceso inicial de Reuniones y Educcion7Extraccion* |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Viabilidad** | **Conceptualización** | **Formalización** | **Implementación** | **Mantenimiento** | |  | | | | | | **Gráfico de Etapas de la Adquisición de Conocimiento** |

Ingeniería de Requisitos **(IR)**

* **Leite (1987)**: “La IR es el proceso para intercambian diferentes puntos de vista y así modelar lo que el sistema va a realizar en un documento de requerimientos”
* **Loucopoulo (1995)**: La IR son las actividades que tratan de comprender las necesidades exactas de los usuarios para plasmarlas en el desarrollo software
* **Sommerville (2005)**: Proceso de desarrollar una especificación de software y comunicarlas a los desarrolladores.
* **Pressman (2006)**: Ayuda a los analistas o ingenieros de requisitos a entender mejor el problema en cuya solución trabajaran.
* **STARTS Guide (1987)**: Proceso en el cual se transforman los requerimientos de los clientes a especificaciones del comportamiento del sistema.

Proceso de la Ingeniería de Requisitos

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| * **Sommerville**: Comienza analizando si el sistema es útil para el negocio, describe una fase de descubrimiento de requerimientos, estos se transforman a un estándar y se analiza que los mismos definan el sistema para el cliente. * **Wiegers (2003)**: Plantea dividir el proceso en dos subprocesos principales   El desarrollo de requisitos lo divide en *elicitación*, *análisis*, *especificación* y *validación*.  La gestión de los requisitos implica establecer y mantener un *acuerdo con el cliente* sobre los *requisitos*   * **Pressman**: Se logra a través de: inicio, obtención, elaboración, negociación, especificación, validación y gestión. * **Hadad (2008): Proceso Genérico**  |  |  |  |  | | --- | --- | --- | --- | | **Elicitación** | **Modelización** | **Análisis** | **Gestión** | | *Identificación de fuentes de información* | *Representación* | *Verificación* | *Identificación de cambios* | | *Recolección de hechos* | *Organización* | *Validación* | *Análisis de cambios* | | *Comunicación* | *Almacenamiento* | *Negociación* | *Realización de cambios* | |  |

|  |  |
| --- | --- |
| Elicitación  Es la habilidad de trabajar en colaboración con los clientes para **descubrir las necesidades** actuales del producto y acordar la visión y las metas del proyecto propuesto. Es el proceso de adquirir todo el conocimiento relevante necesario para producir un modelo de los requerimientos de un problema.   * **Identificación de Fuentes de Información**.   Llamamos **fuente de información** a todo aquello que pueda ayudarnos a comprender tanto el dominio, como el problema en sí mismo.  La priorización de las fuentes de información puede ser en función del costo, cantidad, calidad y disponibilidad de información.   * **~~Recolección de Hechos~~**.   Esta actividad implica conseguir todos los requerimientos del sistema. Para determinar el dominio del sistema, definir sus funciones y encontrar sus restricciones, *se ve en segundo parcial*   * **Comunicación**.   Es la característica más relevante de la elicitación, se establece:  Dentro de la organización (Cliente - Cliente)  Entre la organización y el equipo de desarrollo (Cliente - Analista)  Dentro del equipo de desarrollo. (Analista - Desarrolladores) | Modelización  Se representan, organizan y registran los hechos recolectados. Organizamos la información, a través de un proceso de abstracción, generamos los modelos que irán representando el sistema.   * **Representación.**   Seleccionamos la técnica que utilizaremos para realizar cada modelo en particular.   * **Organización**.   Confeccionamos el modelo que representa la información recolectada.   * **Almacenamiento**.   Se deben guardar todos los documentos que le van dando forma. Debe llevarse en una forma organizada, comprensible y significativa. Por ejemplo:  Autor  Destinatario  Fecha de Entrega  Lugar de recolección  Numero de versión del documento  **Relevar** un sistema implica **aceptar lo que el cliente solicita**. El objetivo es determinar qué software es el que el cliente necesita. |
| Análisis.  Con esta actividad vemos si realmente comprendimos la información que modelizamos. Se debe acordar con el cliente en todo momento los requerimientos y requisitos del sistema.   * **Verificación**.   ¿Estamos construyendo correctamente el producto?   * **Validación**.   ¿Estamos construyendo el producto correcto?   * **Negociación**.   Tiene por objetivo lograr el consenso entre todos los involucrados y los requisitos del sistema.  Estudia y resuelve diferencias de intereses entre los involucrados.  Se evalúa y decide sobre las propuestas que elaboran los ingenieros de requisitos sobre alternativas o soluciones candidatas al problema en cuestión. | Gestión.  Se realiza durante todo el desarrollo del sistema. Se ocupa de mantener actualizados todos los modelos que corresponda, cada vez que se decide tener en cuenta un cambio. Debe documentar todas las propuestas de cambio.   * **Identificación de cambios**.   Consiste en identificar algún problema en los requisitos o un pedido de modificación o de nuevas necesidades.   * **Análisis de cambios**.   Consiste en analizar la validez de los cambios propuestos, estimar, evaluar y negociar el impacto del cambio en el alcance, el cronograma y el costo.   * **Realización de cambios**.   Consiste en implementar los cambios en los documentos de requisitos y en los modelos, así como verificar y validar estos cambios. |

Técnicas para la extracción de conocimientos (Fuentes NO humanas) (Gomez 1997)

* **Lectura** La más utilizada y efectiva ya sea de libros, manuales, documentos formales e informales, informes, investigaciones.
* **Ver Videos** Es un proceso interactivo en el cual generalmente e evalúan procesos correctos o incorrectos o los más eficientes
* **Escuchar Audios** Generalmente de entrevistas con Personas para la extracción concisa de requerimientos.

Técnicas para la educción de conocimientos (Fuentes Humanas) (Gomez 1997)

|  |  |
| --- | --- |
| Entrevista   * Técnica de elicitación más utilizada. * Buena para obtener conocimientos acerca del trabajo actual * Pueden dar ideas sobre el sistema futuro. * Posibilidad del contacto directo con los entrevistados y validación inmediata. * Se debe asumir un conocimiento tácito. * Puede haber problemas por no detectar el vocabulario del entrevistado. * Puede haber diferencias culturales entre entrevistador y entrevistado. * Las entrevistas **abiertas** generalmente se realizan a niveles altos de dirección y normalmente se refiere a cuestiones estratégicas y generales   Puede ser largo y contener detalles irrelevantes.   * Las entrevistas **cerradas** se realizan a niveles operativos donde se desea conocer detalles de volúmenes, procedimientos, modelos y toma de decisiones   Tiene un alto costo de preparación.  Se pueden perder detalles importantes. | Cuestionario - Encuesta   * Sirve cuando no se puede entrevistar personalmente. * Sirve si tenemos que educir información de muchas personas que cubren el mismo rol. * La encuesta es un *procedimiento* a una población, para obtener un resultado * El cuestionario es un *instrumento* y se aplica a personas seleccionadas. * Las respuestas estandarizadas facilitan la evaluación de los resultados por métodos estadísticos. * Las preguntas deben ser neutras o imparciales. * Facilitan la recopilación de información. * Si tienen preguntas cerradas, evitan la dispersión de la información * Hacen impersonal la aportación de respuestas. * No se pueden ir más allá del cuestionario. * Pueden provocar la obtención de datos equivocados. * Hay dificultades para controlar y revisar la información. |
| Joint Aplication Development.   * Técnica grupal, se utiliza para definir requisitos y diseñar la interfaz de usuario. * Reuniones grupales durante un periodo de 2 a 4 días, dónde participan clientes, directivos y desarrolladores. * Se basa en los siguientes principios:   Dinámica de grupo.  Uso de ayudas visuales para mejorar la comunicación.  Mantener un proceso organizado y racional.  Una filosofía de documentación What You See Is What You Get.   * Se elabora una agenda. ( lo que ves es lo que obtienes ) * Se elige un facilitador. * Todo se documenta en forma visual. * Se trabaja Top – Down. * Evita que las opiniones de los participantes se contrasten por separado. * Todo el grupo revisa la documentación generada. * Es difícil hacer coincidir las agendas de todos los participantes. | Brainstorming.   * Puede ayudar a generar una gran variedad de vistas del problema y a formularlo de diferentes formas. * Durante la elicitación, el interés está puesto en ideas para el nuevo sistema. Si la creatividad no surge en la sesión, el analista puede introducir ideas que han surgido mediante otras técnicas. * Requiere poca organización. * Es difícil reunir a todos los participantes. * Pueden salir ideas innovadoras. * Puede no producir resultados de buena calidad. |
| Storyboard (Guion grafico o grafica de historia de movimientos)   * Consiste en representar sobre papel en forma muy esquemática las diferentes interfaces al usuario. |
| Concept Mapping (Mapa Conceptual)   * Los mapas de conceptos son grafos en los que los vértices representan conceptos y las aristas representan posibles relaciones entre dichos conceptos. * Los grafos sirven para aclarar los conceptos relacionados con el sistema a desarrollar * Pueden ser muy subjetivos y pueden ser ambiguos como los ***MALDITOS MODELOS DE DOMINIO*** | Observación   * Técnica de información, no se elicitan respuestas. * No participante: Se recoge la información desde afuera, sin intervenir para nada en el grupo social. * Participante: el investigador se incluye en el grupo, hecho o fenómeno observado. * Es un registro de lo que ocurre en el mundo real. * El observador influye y es influenciado. * Imposibilidad de predecir la ocurrencia espontánea de un suceso. |
| Card Sorting (Clasificacion por Cartas)   * Técnica que se usa para categorizar contenidos. * Se basa en la observación de cómo los usuarios agrupan y asocian entre sí un número predeterminado de tarjetas etiquetadas con las diferentes categorías temáticas del sitio web. * En el **abierto** el usuario puede agrupar las categorías libremente en el número de conjuntos que crea necesarios, en el **cerrado**, los grupos o conjuntos están predefinidos y etiquetados. * Ayuda para la toma de decisiones en la etapa de diseño conceptual y para evaluar una organización concreta de categorías en etapas de evaluación de usabilidad. | Análisis de protocolos   * Consiste en analizar el trabajo del experto a través de sus relatos. * Se hace para descubrir el proceso mental subyacente que realiza en forma natural el experto. * La técnica consiste en grabar, en un protocolo, el comportamiento del experto mientras trabaja en la solución del problema. Ese protocolo se transcribe y luego se analiza. Esto se convierte en un conjunto de razonamientos que llevan a la solución del problema. Requiere de múltiples sesiones. * Posibilita educir hechos no fácilmente observables. * Mejor entendimiento de los hechos. * Es caro * Centrado en la performance del entrevistado * Lo que se dice puede ser diferente a lo que se hace |

Ciclo de Educción

|  |  |
| --- | --- |
| Preparación de la sesión (1)   * 1. **Fijar el contenido de la sesión**: Consiste en elegir una perspectiva o área del dominio y concentrar una serie de sesiones en ella hasta que el analista se encuentre satisfecho con los conocimientos obtenidos.   2. Fijar el <tamaño de grano> de los conocimientos.   3. Fijar la técnica de educción a utilizar.   4. Durante la preparación de la sesión, se debe **planificar el contenido de la misma**. | Sesión (2)   * Las sesiones de educción comienzan con 5 o 10 minutos de repaso de la sesión anterior.   Repaso del análisis de la última sesión.  Explicación al experto de los objetivos de la nueva sesión.  Educción.  Resumen y comentarios del experto. |
| Transcripción (3)   * Tras la grabación de la sesión de educción, es altamente recomendable transcribir el contenido a papel. Tiene un alto costo. | Análisis de la sesión: (3)  Lectura para obtención de una visión general.  Extracción de conocimientos concretos.  Lectura para recuperar detalles olvidados. |
| Evaluación (4): Se evalúa si se han alcanzado los objetivos, si se necesitan más sesiones sobre la misma temática, etc. | |

Tipo de Preguntas para Educcion (Entrevista, Cuestionatrio, Encuesta)

|  |  |
| --- | --- |
| Preguntas Cerradas   * Valores usuales de las respuestas son: Sí, No, Tal vez * Respuestas cortas * Toma poco tiempo responderlas * Respuestas fáciles de estandarizar * Más fácil de analizar y obtener resultados. | Preguntas Abiertas   * Usualmente la pregunta lleva: Qué, Cuál, Cómo, Dónde, Cuándo… * Respuestas más largas que permiten que se obtenga más información. * Toma más tiempo responderlas * Respuestas difíciles de estandarizar * Más difícil procesar y obtener resultados. |

*Diferencias entre Joint Aplication Development y Brainstorming*

|  |  |
| --- | --- |
| Joint Aplication Development   * Es una técnica grupal * Participan representantes del cliente, usuario y desarrolladores * Hay un facilitador, que dirige la sesión * Definir requisitos e interfaz de un sistema * La información que se genera es visual WYGIWYS (lo que ves es lo que obtienes) * Estructurado | Brainstorming   * Es una técnica grupal * Participan representantes del cliente, usuario y desarrolladores * Hay un facilitador, que dirige la sesión * Generar una gran cantidad de ideas * La información puede visual, textual, coloquial, etc. * Desestructurado. |

Modelos de Dominio

Concepto de modelo de dominio, Clases conceptuales e Identificación

|  |  |
| --- | --- |
| Es una representación descriptiva (como es / como se proyecta que sea) de una porción del mundo real (“Dominio”). Captura los tipos de clases más importantes en el contexto del sistema y los sucesos que ocurrirán o han ocurrido.   * *No es una descripción de una implementación en software ni un diagrama de clases* * Las clases de dominio pueden aparecer en formas típicas:   Objetos del negocio que representan las cosas que se manipulan en el negocio.  Objetos del mundo real y conceptos de los que el sistema debe hacer un seguimiento.  **Clases conceptuales**: Son aquellos elementos sustantivos que se pueden encontrar en el espacio del problema. Es una idea, cosa u objeto que podría considerarse en términos de sus símbolos (palabras o imágenes que lo representan), intención (definición de una clase conceptual) y extensión (conjunto de ejemplos a los que se aplica.   * Tienen solamente atributos. * Atributos de clases (Característica intrínseca, distintiva que defina una perspectiva de una clase conceptual)   **Identificación de Clases conceptuales según Larman 2003**  **A** *es una parte física* de **B**  **A** *es una parte lógica* de **B**  **A** *este contenido* físicamente en **B**  **A** *este contenido* lógicamente en **B**  **A** *es una línea de una transacción* o informe de **B**  **A** *es miembro* de **B**  **A** *esta relacionado con una* transacción **B**  **A** *es un evento* relacionado con **B**  **A** *es una transacción relacionada con* otra transacción **B** | **Reglas del Cartógrafo**. Un MD debe:   * *Utiliza los nombres existentes en el territorio.* * *Excluye las características irrelevantes.* * *Excluye cosas que no se encuentran en el dominio del problema.*   Es válido tener clases conceptuales sin atributos, o clases conceptuales con un rol puramente de comportamiento en el dominio, en lugar de un rol de información.  **Rebecca Wirfs:** Los sustantivos son clases conceptuales, los adjetivos atributos y los verbos asociaciones.   * **Grado** (Cantidad de clases conceptuales que participan en la asociación) * **Dirección** o sentido de la asociación (Símbolo / Etiqueta asignada) * **Cardinalidad** (Representa la correspondencia cardinal [“Cantidad de”] entre instancias [caso particular] de cada clase conceptual) |

Requisitos Funcionales y no funcionales.

* **Requerimiento**: Todo aquello que el cliente solicita.
* **Requisito**: Identificación precisa, no ambiguas, consistentes y completas del comportamiento del sistema, incluyendo funciones, interfaces, rendimiento y limitaciones. Lo que el equipo de desarrollo se compromete a cumplir a través de la especificación de un requerimiento.
* **Requisitos del sistema**: Definen con detalle cada una de las funcionalidades, servicios y restricciones del sistema a construir.
* Los requisitos representan el **que** del sistema. Estos comienzan con una descripción abstracta y general de una funcionalidad, servicio o restricción (alto nivel) hasta la definición detallada del mismo (bajo nivel)
* **Requisitos Funcionales (RF):** Los definen las funciones que el sistema será capaz de realizar. En ocasiones también se documenta lo que el sistema no realizará.
* **Requisitos No funcionales (RNF):** Son restricciones que afectan a los servicios o funciones del sistema. El rendimiento, interfaces de usuario, fiabilidad, mantenimiento, seguridad, portabilidad, estándares, etc. Se deben definir de forma que sean cuantificables, para que puedan ser validados por el cliente y no tengan interpretaciones ambiguas.

.

Tipos de Requisitos No funcionales

|  |  |
| --- | --- |
| * **Requisitos del producto**: Especifican que el producto entregado debe comportarse de una manera particular.   **Usabilidad**: Características relacionadas con la facilidad de uso del producto.  Ejemplo: De cada 10 usuarios que utilicen el sistema 9 lo harán correctamente luego de un uso asistido por un instructor.  **Eficiencia**: dar valores a los tiempos de respuesta a funciones específicas o medir capacidades del sistema.  De rendimiento: mide la rapidez de ejecución del sistema.  Ejemplo: *El sistema tarda como máximo 5 segundos en entregar el dinero al cliente, contados a partir del momento que haya finalizado de verificar que hay cambio suficiente en la terminal para satisfacer el pedido del cliente.*  De espacio: mide la cantidad de recursos utilizados por el sistema.  Ejemplo: *El sistema requiere 1 GB de memoria RAM y al menos 20 MB de espacio en el disco rígido*.  **Fiabilidad**: Fija la tasa de fallos para que el sistema sea aceptable, podría expresarse en término de algunos aspectos: Disponibilidad, tiempo mínimo entre fallas, tiempo mínimo de reparación, certeza, bugs máximos o ratios de defecto, bugs o ratios de defectos, backup, estabilidad.  Ejemplo: *El sistema tiene que estar disponible 100% del tiempo. Las 24 horas del día los 7 días de la semana*.  **Portabilidad**: debe expresar las necesidades de crecimiento del producto hacia otras tecnologías de desarrollo, sistemas operativos y/o plataformas de hardware.  Ejemplo: *El sistema se visualizará y funcionará correctamente en Internet Explorer 8, Google Chrome v.23 y Firefox v.17.* | * **Requerimientos Organizacionales**: derivan de políticas y procedimientos existentes en la organización del cliente y en la del desarrollador.   **De Entrega**:especifican cuándo se entregará el producto y su documentación.  Ejemplo: *El sistema debe entregarse dentro de los 180 días hábiles contados a partir de la fecha de la firma del contrato.*  **De Implementación**: hacen referencia a los lenguajes de programación o de método de diseño a utilizar.  Ejemplo: *El software se escribirá en lenguaje COBOL.*  **Estándares**: Algunos ejemplos son los estándares en los procesos que deben utilizarse.  Ejemplo: *El proceso de desarrollo debe ser conforme a la norma ISO 9003.* |
| * **Requerimientos externos**: requerimientos que se derivan de los factores externos al sistema y de su proceso de desarrollo.   **Requerimientos de interoperabilidad**: que definen la manera en que el sistema interactúa con sistemas de otras organizaciones.  Ejemplo: *La aplicación utilizara la interfaz definida en el anexo X, para conectarse al sistema XYZ de la casa matriz.*  **Requerimientos éticos**: Estos son puestos en un sistema para asegurar que será aceptado por sus usuarios y por el público en general.  Ejemplo: *El software no realiza doble registro de datos.*  **Requerimientos legislativos**: Hacen referencia a los requerimientos legales que protegen la seguridad y confidencialidad del producto y toda documentación asociada.  Ejemplo*: Se implementa lo establecido en la Ley 25.236 para la protección de los datos personales que administra el sistema*. |

Propiedades de Requisitos (Hadad, 2008)

**Correctos**: Los requisitos deben ser realmente necesarios y cumplir con el propósito del sistema.

**Conciso**: Un requisito es conciso si es fácil de leer y entender. Su redacción debe ser simple y clara para aquellos que vayan a consultarlo en un futuro.

**Consistentes**: Los requisitos no deben estar en conflicto entre ellos.

**Completos**: Los requisitos deben abarcar toda posible entrada al sistema y toda posible respuesta del sistema.

**Entendibles**: Los requisitos deben poder ser comprendidos por cualquier involucrado con un mínimo de explicación.

**No ambiguo**: Un requerimiento no es ambiguo cuando tiene una sola interpretación. El lenguaje usado en su definición, no debe causar confusiones al lector

**Realizables**: Los requisitos deben poder satisfacerse considerando los recursos disponibles y restricciones, ósea factibles de llevarse a cabo.

**Rastreables**: Los requisitos deben poder relacionarse bidireccionalmente a las fuentes que les dieron origen.

**Verificables**: los requisitos deben poder comprobarse a través del software.

**Abstractos**: los requisitos deben ser independientes de la implementación.

**Referenciales por importancia y/o estabilidad**: los requisitos deben permitir que se les asignen prioridad y grado de estabilidad que presentan en el tiempo.

Especificación de requisitos (IEEE 830-1998).

**Correcta**: Una especificación es correcta si el producto final se apega a la especificación.

**No ambigua**: Si cada requerimiento declarado tiene una sola interpretación

**Completa**: Una especificación está completa si no hay requerimientos con elementos a ser definidos.

**Consistente**: Debe ser consistente internamente (entre los propios requerimientos) y externa. (con otros documentos de especificación)

**Priorizable**: Ayuda a las partes interesadas a establecer cuáles son los requerimientos claves para el negocio.

**Verificable**: Si existe algún método finito y sin costo excesivo para probarlo.

**Modificable**: Deben ser sencillos para poder ser modificados y a su vez mantener su correctitud y completitud.

**Trazable**: La trazabilidad es la habilidad de verificar la historia, ubicación o aplicación de un ítem a través de su identificación almacenada y documentada.

Metodología Estructurada

* Permite construir modelos de sistemas a partir del análisis de sus procesos y/o, actividades que se ejecutan asociadas al sistema.
* El objetivo es organizar las tareas relacionadas a la determinación de requerimientos para obtener la comprensión completa y exacta de una situación dada.
* El modelo de Análisis y Diseño Estructurado Moderno se divide en Ambiental (Visión Externa o Caja Negra) y de Comportamiento (Visión Interna o de Caja Blanca)

Modelo Ambiental

|  |  |
| --- | --- |
| Declaración de propósito (DP)  Es una definición en 1 o 2 párrafos del objetivo o razón de ser del sistema “Captura de la esencia de lo que hace” ósea la Función del mismo | Lista de acontecimientos (LC)  Es una enumeración de los sucesos que tienen lugar en el ambiente del sistema a los que este responde y la evaluación de su respuesta, ósea el Alcance |
| Diagrama de Contexto (DC)  Es la lista de acontecimientos puesta en un diagrama. Tiene una única burbuja que es el sistema con los agentes externos o terminadores y mensajes | Receptor  Emisor |

Modelo de Comportamiento

|  |  |
| --- | --- |
| **Diagrama de Flujo de Datos (DFD)**  Son datos en movimiento. Es una vista dinámica del movimiento de datos de un sistema.   * *Procesos*: indican las funciones individuales que el sistema lleva a cabo. Es deseable que su nombre se conforme por *Verbo Infinitivo + Objeto*. Se representa por círculos. * *Flujos*: Sirven para representar la información que dicho proceso necesita como entrada o genera como salida. Se representa por medio de flechas. * *Almacenes*: (Pueden ser de entrada o de salida) Muestran colecciones de datos que el sistema debe recordar por un período de tiempo. * *Terminadores*: muestran la entidad externa con la que el sistema se comunica. Las relaciones entre terminadores **no se representan en el DFD**. * Niveles de DFD   Nivel 0- Diagrama de contexto: Es una vista general que incluye las entradas básicas, el sistema general y las salidas.  Nivel 1- Diagrama de nivel superior: Es la expansión del DFD de Contexto, incluye los subprocesos del sistema, se numeran con enteros.  Nivel 2 en adelante – Diagrama de detalle: Se realizan refinamientos sucesivos, compuesto por procesos, flujos, almacenes y terminadores.   * Reglas   Los almacenes y entidades se conectan a través de las burbujas.  Un flujo de entrada no puede ser de salida al mismo tiempo.  No puede haber burbujas sueltas.  Los flujos no se bifurcan  No puede haber burbujas sin flujos de entrada.  No puede haber burbujas sin flujo de salida.  Cada burbuja debe recibir por lo menos dos flujos de entrada.  La salida de una burbuja siempre es el mismo flujo de datos.  Los flujos no son recursivos.  Evitar idas y vueltas al mismo proceso  Almacén  Almacén | **Diagrama de Entidad Relación (DER)**   * Es un modelo de red que describe con alto nivel de abstracción la **distribución de datos** almacenados en un sistema. * Enfatiza las **relaciones** entre los almacenes de datos de los DFD que de otra forma se hubieran visto solo en la especificación de procesos. * **Se conforma por:**   *Tipo de objeto*  Se identifican unívocamente  Cada uno juega un papel necesario en el sistema que se construye  Cada uno puede describirse por uno o más datos  *Relaciones*  Se representan mediante un rombo  Cada instancia de la relación representa una asociación entre cero o más ocurrencias de un objeto y cero o más ocurrencias del otro.  *Indicadores asociativos de tipo de objeto*  Representa algo que funciona como objeto y relación.  *Indicadores de subtipo/supertipo*: tipos de objetos de una o más subcategorías, se conectan al supertipo por medio de una relación sin nombre. |
| **Diccionario de Datos (DD)**  Proporciona un listado organizado de todos los datos del sistema con definiciones precisas y rigurosas, describe el significado de los flujos y paquetes de datos de los almacenes que se muestran en el DFD.   |  |  | | --- | --- | | * **=** SE COMPONE DE * **+** Y * **( )** Optativo * **{ }** Iteración * **[ ]** Selecciona una de varias alternativas * **|** Separa alternativas en la construcción * **\*\*** Comentarios * **@** Campo clave | * Nombre del cliente = Titulo + Nombre + Apellido * Título = [Sr. | Srta. | Sra. | Dr. | Prof.] * Nombre= {Carácter válido} * Apellido = {Carácter válido} * Carácter válido= [A - Z| a -z|´|-] | |
| Especificaciones de Procesos (EP)  Es una forma de especificar lo que hay en una burbuja (seudocódigo), ósea lo que debería suceder dentro del proceso   * La forma más utilizada para realizarlas es el lenguaje estructurado. * Debe expresarse de una manera que pueda ser verificada, evitando ambigüedades. * Debe especificarse en una forma que pueda ser entendido claramente por los involucrados. |
| Diagrama de Transición de Estados (DTD)  Permite especificar la varianza en el tiempo de una entidad, una relación, etc. |

Metodología orientada a objetos

* Se basa en el POO donde los objetos tienen responsabilidades y se los modelan para resolver pequeños problemas que en suma son la solución.
* El objeto representa casi cualquier información compuesta que un sistema. Es una entidad tangible que exhibe algún comportamiento bien definido.
* Los objetos pueden ser una entidad externa al sistema, una cosa, un suceso, un evento, Un rol, una unidad organizacional, un lugar o una estructura
* Los objetos tienen un **comportamiento** según las responsabilidades que deben atender, un **estado** según los atributos, y una **identidad** que lo hace único
* El POO se especializa en resolver problemas complejos y se lo utiliza para cambios frecuentes de código.
* Clase: Es un modelo de témplate que permite generar diferentes instancias de objetos en el sistema

|  |  |
| --- | --- |
| Clase   * Una clase es una abstracción de las cosas que forman parte del problema y de la solución. * Representa un conjunto de objetos. * Comparten los mismos atributos y responsabilidades. * Un atributo es una propiedad de una clase identificada con un nombre, es compartida por todos los objetos de esa clase. * Crear un objeto/instancia consiste en colocarle los valores a los distintos atributos de la clase. * Un objeto siempre es instancia de una clase. * Los atributos de una clase y el nombre de la misma dependen del contexto. | Relaciones entre clases   * **Herencia**: Es una relación entre clases por la cual una o más clases heredan los atributos y el comportamiento de una clase y le agregan atributos y comportamiento propio hasta.   Es también llamada relación “Es un”  Se puede obtener por especialización o por generalización.  **Especialización**: De la superclase se distinguen las clases hijas.  **Generalización**: De las clases hijas se distingue una superclase.  *Ventajas*:  Reutilización de código ya testeado.  Se pueden agregar subclases sin que se altere el sistema.  Se pueden eliminar subclases sin que se altere el sistema.  *Desventajas*:  Genera alto acoplamiento entre clases (Lo que está en uno se traslada a otro y no es modificable) y baja cohesión (Los métodos solo hacen lo mínimo necesario)   * **Asociación Simple**: Relación semántica entre clases.   Se puede dar el caso de tener que utilizar dos asociaciones con distinto sentido de lectura según qué necesitemos modelizar.  Se pueden tener relaciones entre instancias de la misma clase.   * **Agregación** (Relación Parte de Tiene un): Se tiene una clase que está formada por atributos que son objetos de otra clase. Estas clases son independientes entre sí (El ciclo de vida de un objeto no afecta al ciclo de vida de otro) * **Composición**: Es una relación entre entidades fuertes y débiles desde el punto de vista de objetos. Su diferencia con agregación es que si deja de existir el objeto fuerte afecta directamente al débil. |
| Pilares de Programación Orientada a Objetos   * **Encapsulamiento**: Los objetos de una clase tienen atributos que pueden ser privado y solo los ve esa clase. * **Ocultamiento** **de información**: La comunicación entre objetos es mediante mensajes y los otros objetos del sistema solo pueden ver lo permitido. * **Reutilización**: La clase utiliza el código de la superclase. * **Polimorfismo**: La capacidad de poder enviar el mismo mensaje a objetos de distintas clases |

Metodologías Agiles

Conjunto de metodologías denominadas livianas o ligeras que utilizan practicas basadas en resultados, la gente y la interacción.

|  |  |
| --- | --- |
| Principios de las metodologías agiles.   * Los **individuos y las interacciones** son más importantes que los *procesos* y las *herramientas* * El **software funcionando** sobre la *documentación extensiva* * **Colaboración con el cliente** sobre la *negociación contractual*. * **Respuesta al cambio** sobre *seguir un plan* | Principios del manifiesto ágil   1. La mayor prioridad es satisfacer al cliente con entregas tempranas continuas y de valor 2. Aceptamos cambios en requisitos así se aprovecha la competitividad del cliente 3. Entregamos software funcional entre 2 semanas a 2 meses 4. Los responsables del negocio y los desarrolladores trabajan juntos durante todo el proyecto 5. Los proyectos se desarrollan con gente motivada en entornos para tal fin 6. La forma de comunicación más clara es el cara a cara 7. El software funcional es la medida del progreso 8. Se apoya el desarrollo sostenible a un ritmo constante e indefinido 9. La atención continua a la excelencia técnica y el buen diseño mejora la agilidad 10. La simplicidad o el arte de maximizar la cantidad de trabajo no realizado es esencial 11. Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados 12. A intervalos regulares se debe reflexionar sobre la efectividad para autorregularse y perfeccionarse |
| Técnicas Agiles   * Programación Extrema (Kent Beck) * Scrum (Jeff Sutherland) * Crystal Clear (Alistair Cockburn) * Desarrollo basado en funcionalidades (Jeff De Luca) * Desarrollo Ágil de Software (Jim Highsmith y Sam Bayer) * Proceso Unificado (Scott Ambler) * Método de Desarrollo de Sistemas Dinámicos (Consorcio DSDM) * Desarrollo de Software Lean (Mary Poppendieck y Tom Poppendieck) |

|  |  |
| --- | --- |
| Practicas Agiles   * En entregas pequeñas el equipo comprime el tiempo de entregas del producto. Así se resuelve primero las características más importantes y luego progresivamente las demás * En una semana de 40hs se trabajará intensamente las 40hs. Trabajar tiempo extra es dañino para la salud, el proyecto y los desarrolladores. Así se motiva el trabajo intenso y eficiente y luego de estas jornadas un descanso que evite el estrés. * Alojar al cliente en el sitio implica que durante el desarrollo se encuentre un experto del cliente asesorando al equipo y dando su punto de vista para asignar la prioridad y equilibrar las necesidades a largo plazo de la empresa cliente que solicita el software * La programación en pareja es una importante práctica, así los programadores aprenden las costumbres del otro, el más experimentado codifica mientras que el otro planifica y los roles cambian permitiendo ahorro de tiempo y pensamiento torpe, suscitando la creatividad | Proceso Ágil   1. Escuchar Historias de Usuarios por medio del cliente 2. Dibujar un modelo del flujo de trabajo lógico para apreciar las decisiones de negocios en la historia de usuario 3. Crear Historia de usuarios con base al modelo lógico 4. Desarrollar prototipos de visualización con interfaces graficas potenciales 5. Usar retroalimentación de los prototipos y los diagramas de flujo de trabajo lógico para desarrollar el sistema físico de datos del mismo |

Comparativa con entre Métodos Agiles SCRUM, XP, DSDM

Características **XP, SCRUM, DSDM**

|  |  |  |
| --- | --- | --- |
| * Trabajar estrechamente con el cliente * Mini entregas, en tiempos reducidos (entre 2 y 4 semanas) * Pair Programming (programación de a pares). * 40 horas semanales de trabajo. * Proceso guiado por las pruebas (test driven development) * Propiedad colectiva del código * Todos los integrantes del grupo de desarrollo y el cliente juntos en el mismo lugar físico. * Refactorización del código. * Integración frecuente. | * Equipos auto dirigidos y auto organizados. * Una vez seleccionadas las tareas para una iteración, no hay adición externa de tareas a la misma. * Encuentros diarios con preguntas, previamente * definidas. * Realizar una demo a los stakeholders externos al final de cada iteración. * Plan adaptativo dirigido por el cliente. * Todos los integrantes del grupo de desarrollo juntos en el mismo lugar físico | * Entregas frecuentes. * Los cambios en los requerimientos son bienvenidos. * La implicación activa de los usuarios es imprescindible. * Desarrollo iterativo e incremental. * Todos los cambios durante el desarrollo pueden ser revertidos * La agilidad para el propósito del negocio es un * criterio esencial para la aceptación de los productos. * El testeo es integrado a través de todo el ciclo de * vida. * La colaboración y la cooperación de todos los stakeholders es esencial |

Roles **XP, SCRUM, DSDM**

|  |  |  |
| --- | --- | --- |
| * Cliente * Desarrollador (programador y tester) * Management (coach, tracker) * Consultor técnico | * Cliente * Desarrollador * Management * Chicken | * Desarrolladores seniors * Coordinador técnico * El usuario embajador * El sponsor ejecutivo * El visionario. |

Ciclo de Vida **XP, SCRUM, DSDM**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Concepción del proyecto | Especificación de los requerimientos | Diseño | Codificación y Desarrollo | Testeo | Producción |
| **XP** | * Fase de exploración | * Fase de exploración * Fase de Planeamiento | * Fase de iteraciones para la 1° entrega | * Fase de iteraciones para la 1° entrega | * Fase de iteraciones para la 1° entrega * Fase de producción | * Fase de mantenimiento |
| **SCRUM** | * Fase de Planeamiento | * Fase de Organización | * Fase de Desarrollo | * Fase de Desarrollo | * Fase de Desarrollo | * Fase de reléase |
| **DSDM** | * Fase de estudio de factibilidad | * Fase de estudio de negocio * Fase de iteración con modelo funcional | * Fase de iteración de diseño y construcción | * Fase de iteración de diseño y construcción | * Fase de iteración de diseño y construcción | * Fase de implementación |

Scrum

|  |  |  |
| --- | --- | --- |
| Características   * Optima para equipos de trabajo de hasta 8 personas * La documentación en Scrum no es indispensable * El cliente va viendo mediante entregas parciales la evolución del producto cada 2/4 semanas * Es indispensable que el cliente se involucre en el proyecto desde el comienzo * Se hacen reuniones diarias de todos los equipos de desarrollo * Diariamente se realiza test de código para integrarlo al existente | Partes   * Product Back-Log (Funciones) * Sprint BackLog (Tareas) * Sprint Planing Meeting (Planeamiento) * Daily Meeting (Tarea diaria) * Sprint Review Meeting (Revisión) * Sprint Restrospective Meeting (Lección aprendida) | Roles   * Product Owner (Responsable) * Scrum Master (Arbitro) * Scrum Team (Equipo Desarrollo) * StakeHolder (Usuarios o clientes) * Manager (Administrador) |
| Product Back-Log  Conjunto de funcionalidades expresadas por el cliente-usuario mediante HU jerarquizadas por relevancia y para el sistema……………… | Sprint BackLog  Conjunto de funcionalidades como HS que se toman de **Product Backlog** para ser abordadas y resueltas por el Scrum Team a lo largo de 1 Sprint | |
| Sprint Planing Meeting  Encuentro de 8 horas máximo donde el **Scrum Team** determinará que se ahora, que se incluirá en el **Sprint Backlog**. Se define tiempos y prioridades y se identifican y comu8nican cuanto se hará | Daily Meeting  Encuentro de no más de 15 minutos puntualmente en el mismo horario y mismo lugar. Todos serán bienvenidos, pero solo pueden participar los involucrados del Scrum Team para responder preguntas como   * ¿Qué tareas realice? ¿Qué inconvenientes tuve? ¿Qué tareas realizare hoy? | |
| Sprint Review Meeting  Encuentro de hasta 4 horas donde se revisa que se completó y que ……. Del **Sprint Planing** y su **Sprint Back-Log**  También presentar el producto software a liberar en esta interacción a los StakeHolders. Lo que está incompleto no se muestra | Sprint Restrospective Meeting  Encuentro de 4 horas máximo donde se realiza la identificación y el análisis de las lecciones aprendidas, aquellas conclusiones acerca del proceso del equipo de trabajo que sea importante adquirir como conocimiento para el proceso de mejora continua | |
| Product Owner  Responsable de proyecto, escribe HU, las prioriza y las coloca ene l **Product Backlog**. Debe conocer profundamente el negocio y debe integrarse al **Scrum Team** | Scrum Master  Su trabajo es eliminar los obstáculos, incidentes que impidan el al equipo alcanzar los objetivos (**Sprint Backlog**), oficia de árbitro. No es el líder de equipo ya que el Team se auto organiza. | |
| Scrum Team  Son los responsables de llevar adelante todo el sprint del proyecto, desarrollando y entregando la correspondiente versión del mismo. Este equipo está dispuesto a auto gestionarse con responsabilidad y posee conocimiento de nivel medio o superior en diversas habilidades transversales de desarrollo de software ya sean de analista, diseñador, programador, evaluador, documentador, etc. | StakeHolder  Hace posible el proyecto y seria el eventual receptor del beneficio que el producto software brindara. Participan del **Sprint Meeting** | |
| Managers: Administradores.  Son los responsables de establecer y mantener los entornos de desarrollo del proyecto. Con ellos lidiara el Scrum Master | | |
| SystemsPlus-Scrum-Image-.jpg (1024×688) | | |

Historias de usuario (XP)

Las metodologías agiles con Scrum utilizan las historias de usuario como el instrumento principal para modificar los

requerimientos del usuario, son descripciones cortas y simples, escritos desde la perspectiva que de la persona que

necesita una nueva capacidad, por lo general el usuario, área de negocio o bien.

|  |  |
| --- | --- |
| Porque Usarlas   * Potencian la participación del equipo en la toma de decisiones. * Se crean y evolucionan a medida que el proyecto avanza. * Son peticiones concretas y pequeñas. * Contienen solo información imprescindible (+ es -). * Apoya la colaboración y conversación entre los miembros del equipo fundamental para este tipo de desarrollo | * ¿Qué son las Historias de usuario?   Una manera simple de describir una tarea concisa que aporta valor al usuario o al negocio que tienen que tomar  una decisión. No se detalla más hasta el momento en que las historias de usuario se van a desarrollar.  Las historias de usuario pueden ser creadas durante las conversaciones con los usuarios interesados (**StakeHolder**)  sobre nuevas funcionalidades o mejoras al producto. |
| * **Tarjeta**: Descripción escrita en lenguaje de negocio, que sirve como identificación y recordatorio del requerimiento y ayuda a la planificación mediante la priorización (Aprender la jerga del negocio es importante). * **Conversación**: El dialogo que ocurre entre los miembros del equipo y el **Product Owner** para aclarar los detalles y dudas sobre las historias de usuario. * **Confirmación**: Que pruebas y/o validaciones se llevan a cabo para poder decir que una historia de usuario se ha completado con éxito | |
| Características   1. **Independiente**: Deben ser atómicos en su definición es decir que debe intentar que no dependa de otras historias de usuarios para poder completarlo. 2. **Negociable**: Son entidades vivas, debes son ambiguas en su enunciado para poder ser debatibles dejando sus conclusiones a los criterios de aceptación. 3. **Valorados**: Deben ser valorados por el cliente. Para poder saber cuánto aporta al valor de la aplicación y junto con la estimación convertirse en un criterio de prioridad. 4. **Estimables**: Aunque sea simple en poco como leer una bola de cristal deben ser estimados. Tener en su alcance suficientemente definido como para poder suponer una medida de trabajo en la que pueda ser completado. 5. **Pequeñas**: Para poder realizar una estimación concreta valorar y no perder la visión de la historia de usuario se recomiendan que sean mayores a 2 días y menores a 2 semanas. 6. **Verificables** (Gran avance de las historias de usuario): Junto con el cliente se acuerdan unos CRITERIOS DE ACEPTACION que verifican si se han cumplido con las funcionalidades descriptas esperadas. (Desmenuzable que se pudiera discutir si es correcta con el cliente) | |

Tarjeta de Historia de Usuario

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Enumeración de HU | | | | | Criterio de Aceptación de HU | | | |
| ID | Rol | Características Funcionalidades | Razón Resultado | Numero de Escenario | Título de  Criterio de Aceptación | Contexto | Evento | Resultado Comportamiento |
| 01 | Como cliente | Necesito ver un listado de categorías de productos y poder seleccionar una categoría | Con la finalidad de realizar búsquedas de productos por categorías | Escenario 1 | Categoría con al menos 1 producto | En caso de una categoría tenga al menos un producto asociado | Cuando se despliegue el listado de categorías a seleccionar | A continuación del nombre de las categorías se mostrará entre paréntesis el número de productos asociados |
| Escenario 2 | Categoría sin producto | En caso de que una categoría no tenga productos asociados | A continuación del nombre de categoría se mostrará “SIN PRODUCTOS ASOCIADOS” |
| Escenario 3 | Ordenamiento de categoría | N/A | El sistema mostrando las categorías en orden alfabética |

Casos de uso.

Un caso de uso es una descripción detallada en un lenguaje de usuario de los pasos a seguir para cumplir en forma completa un requerimiento funcional. Utiliza sus propios términos para definir **que se hace,** ósea para cumplir el requisito funcional y no el **~~cómo se hace~~.** Están escritos desde el punto de vistan del cliente.

|  |  |
| --- | --- |
| Actor   * Es “algo o alguien” que usa le sistema * Son siempre externos al sistema * Tiene un rol particular * No es un usuario del sistema ya que un usuario asume un rol y un grupo de usuarios puede asumir más de un rol mientras que el actor tiene individualmente un solo rol * Pueden ser personas físicas, sistemas software o sistemas hardware * Si un actor realiza las mismas tareas que otro y agrega propias se lo llama generalización del actor principal y generalmente es un actor secundario * Los actores se asocian al sistema mediante actividades |  |
| Características   * Están expresados desde el punto de vista del actor y no del sistema, el punto de vista del actor representa una vista externa del sistema   Ej.: Registrar Venta: Iniciado por Actor Vendedor desde su punto de vista es un *inicio de venta* y desde el sistema es una *recepción de información de pedido de compra*. **Es un error común confundir quien interactúa.**   * Se documentan con texto informal porque está dirigido al cliente. * Describe tanto lo que hace el sistema como lo que hace el actor que interactúa * Son iniciados por un único actor a la vez cada CU. * Están acotados a una única funcionalidad completa del sistema.   Ej.: Registrar Venta implica acceder a esa funcionalidad, pero si se quiere **Ingresar Artículos Vendidos** para Registrar Venta como es ***parte de***, no se debe indicar esa sub-funcionalidad ya que es parte de la cronología de Registrar Venta   * Debe dejarle un beneficio al Actor que lo inicio al CU que lo inicio | |

|  |
| --- |
|  |

Metodología de Casos de Uso (MCU)

|  |
| --- |
| 1. Encontrar los actores que cumplan roles distintos y las generalizaciones de estos. Los nombres y su descripción 2. Encontrar CU para cada actor y definir el nombre de cada uno. 3. Identificar nuevos CU desde los a través de estos métodos:   Variación significativa en CU existente: Un CU puede ser muy específico al variar dependiendo del caso.  Ej.: Registrar Venta: Obtiene Registrar Venta Minorista y Mayorista  CU opuesto:  Ej.: Registrar Venta: Obtiene Anular Venta  CU que preceden: CU que están antes de otro. Puede no pertenecer al sistema o justamente ser uno nuevo.  CU que suceden***:*** ser uno   1. Hacer Diagrama de CU 2. Describir cada CU en trazo grueso: Visto desde el Actor, describirlos y así obtener los requerimientos del sistema y verificar que así sea 3. Priorizar CU   Imprescindibles: Si no se implementan el sistema no funcionara  Importantes: Su implementación puede ser negociable  Deseables: Aquellos que se implementan si hay tiempo   1. Describir CU en trazo fino: Visto desde actor describir el Flujo normal y el alternativo paso a paso según la Interfaz tentativa. 2. Ver de nuevo las relaciones entre los CU, seguramente algo nuevo surgió 3. Realizar interfaz tentativa final |

Interfaz Tentativa

|  |  |  |  |
| --- | --- | --- | --- |
| Tipo | Nombre | El sistema | El usuario |
|  | Caja de Texto | - | Ingresa… |
|  | Lista desplegable | Carga/Verifica | Selecciona |
|  | Etiqueta | Muestra | - |
|  | Botón de Acción | - | Oprime - Pulsa |
|  | Contenedor | Muestra | - |

Especificaciones

|  |
| --- |
| * **Nombre del caso de uso**: Comienza con un verbo en infinitivo y está expresado desde el punto de vista del actor. * **Actor principal**: Es el que inicia el caso de uso y el que interactúa con el sistema. * **Actor secundario**: Participa una vez iniciado el CU. Nunca inician el CU. * **Descripción general o trazo grueso**: Es una breve descripción, a grandes rasgos, de lo que hace el caso de uso, donde se debe especificar quién es el actor involucrado, la funcionalidad que se va a llevar a cabo y cuál es el beneficio que deja el caso de uso para el actor. * **Precondición**: como precondición se debe indicar el nombre del/los CU que tienen que haberse ejecutado antes del caso de uso que se está describiendo.   Ej.: Registrar Venta la precondición debe ser *Registrar Productos*.   * **Postcondición**: como postcondición se indica cual es el resultado de interés o beneficio que deja el caso de uso para al actor.   Ej. Registrar Venta la postcondición debe ser *Venta Registrada*   * **Descripción detallada o Trazo fino**: es la descripción de la interacción entre el actor y el sistema paso a paso y convenientemente con una interfaz tentativa.   **Flujo normal**: el flujo normal se describe en base a la interfaz y sigue el curso normal de los acontecimientos, es el caso en el que *“todo bien”.* Dentro del flujo normal puede ocurrir que el actor tenga que realizar varias interacciones. La frase típica es:  “para cada x seleccionado se repiten los pasos y a z”  “se repite el paso x hasta que ocurra y”  **Flujo alternativo**: en el flujo alternativo se describen las alternativas al flujo normal, ante determinada acción del actor se espera que el sistema responda de una manera y este lo hace de otra, en estos casos además de describir la alternativa hay que indicar a que punto del flujo normal se vuelve luego o si se finaliza el caso de uso. Ej.: Registrar Venta: El actor en forma errónea ingresa un *código de producto inexistente*, en este caso el sistema deberá informar de dicha situación y regresar al punto del flujo normal donde se permita al actor ingresar nuevamente el *código de producto.* Pueden ser:  Representan un error o una excepción.  No tienen sentido en sí mismas fuera del contexto del caso de uso en el que ocurren.   * **Relaciones entre CU:** Se hacen al terminar el trazo fino e incorporan todas las dependencias de los CU entre si y por falla de los flujos   **Extensión**: Representa algo que no siempre ocurre. Es una relación entre 2 CU donde el que extiende agrega funcionalidad. Generalmente por excepción, pero no siempre es de esta manera.  Ej.: Autorizar Venta extiende Consultar Historial de Pagos  **Inclusión**: Representa algo que siempre ocurre dentro de **2 o más CU sí o sí**. Todos estos CU son independientes por sí mismo excepto por la tarea repetida. |

Comparación entre Casos de uso e Historias de usuario (XP)

|  |  |  |
| --- | --- | --- |
|  | **Casos de Uso** | **Historias de Usuario** |
| **¿Qué es?** | Análisis funcional con descripción de requisitos | En lenguaje coloquial, recordatorio de lo acordado bajo mínimos criterios de aceptación |
| **¿Qué hace?** | Pruebas por seguimiento de requisitos | Incorpora Criterios de Aceptación |
| **Objetivo** | Modelar la interacción Actor-Sistema | Redactar una descripción breve de una funcionalidad desde la vista usuario |
| **Estructura** | Texto detallado sobre planilla predefinida a completar con conceptos técnicos (Objetivo, actor, beneficio, etc.) | Corta y consistente. 1 o 2 frases escritas en el lenguaje del usuario |
| **Planificación** | No contemplada | Contemplan la planificación de prioridades (por el cliente) y estimación de esfuerzo (por el equipo de desarrollo) |
| **Agilidad** | Poco ágil. Requieren tiempo para el análisis y redacción según una plantilla | Muy ágil. Se redactan al toque perro |
| **Comprensión** | De difícil comprensión, incluso para gente entrenada | Fáciles de leer y comprender |
| **Mantenimiento** | Suelen pertenecer a documentos de cientos de paginas | Fáciles de mantener por su baja complejidad e independencia |
| **Comunicación** | Modelo textual, con diagramas de apoyo | Basado en comunicación verbal y orientado a la colaboración y discusión para clarificar detalles |
| **Soporte** | Escrito en documentos para ser archivados como referencia | Escrito en tarjetas para ser usadas |
| **Duración** | Larga vida. Pueden ser abordados luego de varias iteraciones | Deben ser implementadas y aprobadas en una única interacción |
| **Autores** | Definida por observador que hace de intérprete del CU (el analista) | Definida por usuarios (login por ej.) y clientes (paga por ej.) |
| **Pruebas** | La definición se realiza separadamente y con posterioridad | Contiene Criterios de Aceptación al reverso de la tarjeta de la HU |
| **Contexto** | Proporcionan una visión general del sistema y su integración | Proporciona una visión menos obvia y menos integral. Para clarificar esto se usan las Pruebas y la retroalimentación de manera importante. |
| **Metodologías** | Asociadas al Proceso Unificado | Asociada a metodologías agiles, especialmente XP |

UML Lenguaje Unificado de Modelado

UML es un lenguaje unificado que permite modelar software en todas las etapas de desarrollo y en especial en análisis y diseño

|  |  |
| --- | --- |
| Permite   * **Visualizar**: Ayuda a representar lo que se desea construir a través de un gráfico. Así permite mejorar comunicación entre desarrolladores y entre el cliente y el desarrollador. * **Especificar**: Permite definir requisitos, funcionalidades, arquitectura, etc. de un sistema evitando ambigüedades * **Construir**: UML no es un lenguaje de programación, pero si permite modelar diagramas que serán adaptados a un código representativo * **Documentar**: Los modelos que usa UML pueden ser usados como documentación del Sistema * **Modelo-Proceso**: UML no es un proceso, permite crear modelos, pero no como estos se transformarán en procesos que realicen las tareas asignadas * **Tres amigos**: Bosh, Jacobson, Rumbaugh | Preguntas Abiertas   * Usualmente la pregunta lleva: Qué, Cuál, Cómo, Dónde, Cuándo… * Respuestas más largas que permiten que se obtenga más información. * Toma más tiempo responderlas * Respuestas difíciles de estandarizar * Más difícil procesar y obtener resultados. |
| Elementos   * **Elementos Estructurales**   **Clases**: Es un caso general de un conjunto de instancias de un objeto  Objeto:  Interfaz: Entidad que reune  Clase de Uso  Clase Activa  Colaboración  Componente  Artefacto  Nodos   * Elementos de Comportamiento   Interacción  Estado  Actividad  Elementos de Agrupación  Paquete   * Elementos de Anotación   Notas |  |
| Relaciones   * Dependencia * Asociación * Casos Especiales de Asociación   Agregación simple  Composición   * Generalización * Realización |  |
| Diagramas   * Diagramas Estructurales   Diagrama de clases  Diagrama de componentes  Diagrama de estructura compuesta  Diagrama de objetos  Diagrama de paquetes  Diagrama de artefactos   * Diagrama de Comportamiento   Diagrama de casos de uso  Diagrama de estados  Diagrama de actividades   * Diagrama de Interacción   Diagrama de secuencia  Diagrama de comunicación  Diagrama de Tiempos  Diagrama de visión global de interacciones |  |

Proceso Unificado

|  |
| --- |
| Es un proceso para desarrollar gran parte de sistemas ponderando:   * Trabajo coordinado * Integrar fases de desarrollo * Proporcionar un elemento común de comunicación * Desarrollar una guía de actividades de los equipos * Especificar lo que se debe producir en cada etapa |
| Caracteristicas  **Dirigido por casos de uso**   * Se usa un conjunto de casos de uso o escenarios para definir la funcionalidad del sistema. Los casos de uso limitan la especificación de los requisitos del sistema y además guían el proceso de desarrollo.   **Centrado en la arquitectura**   * La arquitectura y los casos de uso deben desarrollarse en forma paralela, los casos definirían la FUNCION y la arquitectura la FORMA. La arquitectura incluye los aspectos estáticos y dinámicos más importantes del sistema.   **Iterativo e Incremental**   * Se divide el trabajo en iteraciones o mini proyectos. En los primeros se toma muy en cuenta el riesgo, por lo cual se reduce el riesgo del proyecto. En cada iteración se especifican los casos de uso. Esto produce un crecimiento del producto y facilita ubicar cambios sin tener mucho costo, además los trabajadores ven un avance en sus trabajos, lo cual produce un mayor rendimiento. |
| Ciclo de Vida  Cada iteración realiza los procesos estándar de ciclo de vida y además finaliza con un hito (artefactos o documentos) los cuales permiten verificar lo hecho hasta el momento y tomar decisiones antes de pasar a la próxima etapa.   * ***Inicio:*** Se buscan los objetivos principales y la arquitectura que se utilizara.   Artefacto: Requerimientos planteados como casos de uso, boceto inicial de la arquitectura, descripción de proyecto, versión de plan de proyecto y modelo de negocio.  Hitos: Alcance y objetivo.   * ***Elaboración:*** Se detallan los casos de uso, la arquitectura y se planean las próximas iteraciones.   Artefacto: Prototipo de arquitectura, casos de prueba y la mayoría de los casos de uso.  Hitos: Arquitectura.   * ***Construcción:*** Se desarrolla el producto sobre los casos de uso definidos para esta iteración.   Artefacto: Sistema software, Casos de prueba y manuales.  Hito: Versión beta con funcionalidades.   * ***Transición:*** Se prueba el producto, se capacita a los usuarios y se buscan errores. Una vez corregido, se cambia la versión beta por la final.   Artefacto: Igual que en construcción.  Hitos: Lanzamiento del producto u obtención de la versión final. |

Características: centrado en la arquitectura, desarrollo iterativo e incremental y dirigido por Casos de uso.

Cuáles son las fases y disciplinas que componen el proceso unificado.

Que actividades se realizan en cada una de ellas.

Cuáles son los roles y los artefactos producidos

Modelo de análisis

Diagrama de clases de análisis

Diagrama de comunicación de análisis

Descripción de las clases con los atributos

Propios y vinculantes.

Otros diagramas de UML

Diagrama de actividades. Diagrama de estados