Modelos del desarrollo de software

En los años 50 no existían metodologías de desarrollo, el desarrollo estaba
A cargo de los propios programadores. No se sabía la fecha exacta en que
Concluiría un proyecto de software, no había forma de controlar las actividades
Que se estaban desarrollando. Tampoco se contaba con documentación
Estandarizada. El nacimiento de técnicas estructuradas es lo que da origen
Al desarrollo de aplicaciones a través de métodos de ingeniería. La informática
Aporta herramientas y procedimientos que se apoyan en la ingeniería de
Software con el fin de mejorar la calidad de los productos, aumentar la
Productividad y trabajo de los ingenieros desarrolladores de software, facilitar
El control del proceso de desarrollo de software y suministrar a los
Desarrolladores las bases para construir software de alta calidad en una
Forma eficiente.
El objetivo principal que busca la ingeniería de software es convertir el
Desarrollo de software en un proceso formal, con resultados predecibles,
Que permitan obtener un producto final de alta calidad y satisfaga las
Necesidaddes y expectativas del cliente.

Modelos del desarrollo de software

Una metodología define una representación que permite facilitar la manipulación
De modelos y la comunicación e intercambio de información entre todas las
Partes involucradas en la construcción de un sistema.
Goncalves (2005) plantea que la experiencia ha demostrado que los proyectos
Exitosos son aquellos que son administrados siguiendo una serie de procesos
Que permiten organizar y luego controlar el proyecto, considerando válido
Destacar que aquellos procesos que no sigan estos lineamientos corren un
Alto riesgo de fracasar.
Metodología de desarrollo de software: es un enfoque estructurado
Para el desarrollo de software que incluye modelos de sistemas, notaciones,
Reglas, sugerencias de diseño y guías de procesos.

Los modelos pueden pensarse como marcos de trabajo del proceso y que
Pueden ser adaptados para crear procesos más específicos.

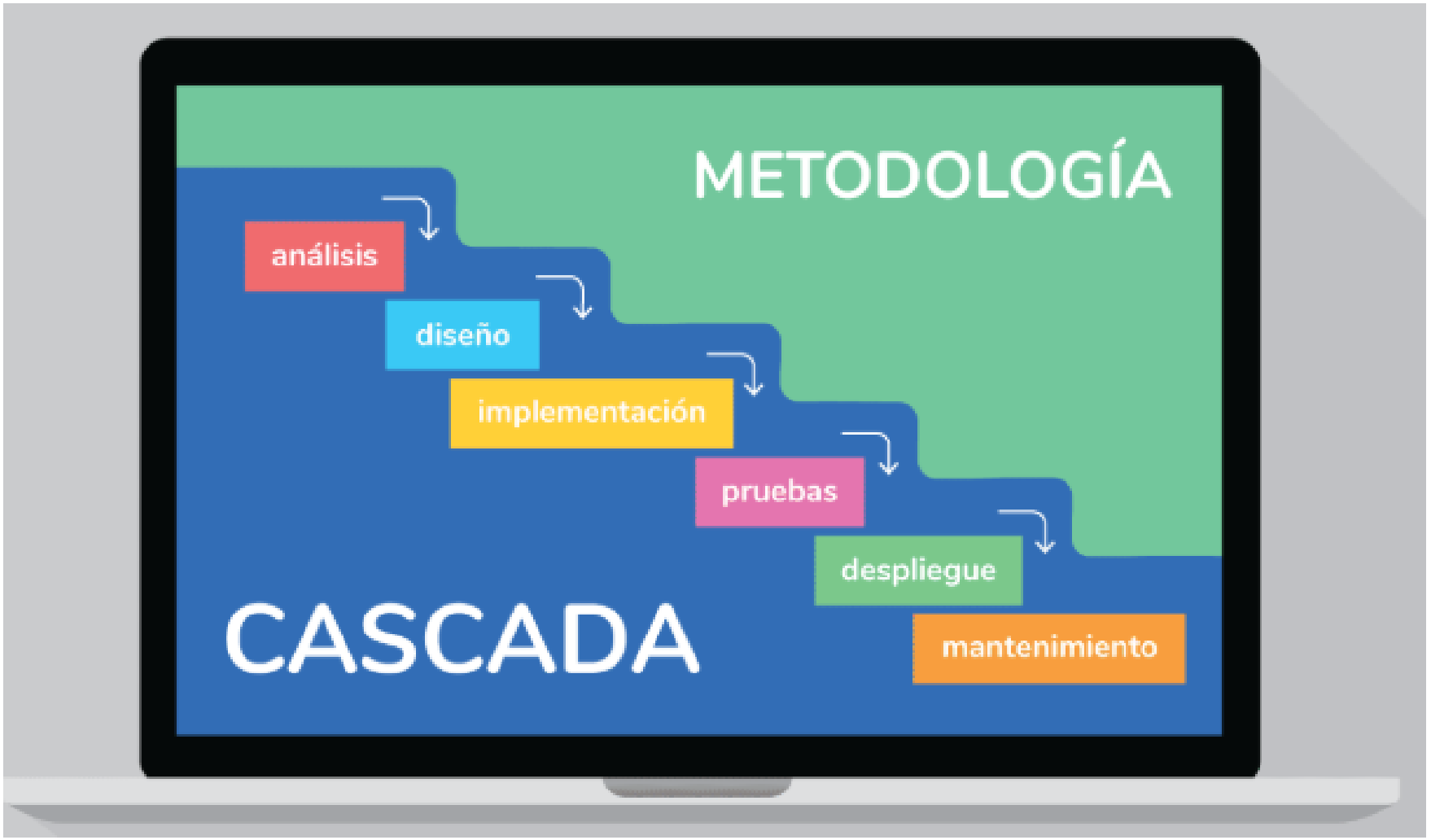
Modelo de cascada

El desarrollo en cascada es un procedimiento lineal que se caracteriza por Dividir los procesos de desarrollo en sucesivas fases de proyecto. Al contrario Que en los modelos iterativos, cada una de estas fases se ejecuta tan solo Una vez. Los resultados de cada una de las fases sirven como hipótesis De partida para la siguiente.

Royce propone un modelo compuesto por siete fases:

1. Requisitos de sistema
2. Requisitos de software
3. Análisis
4. Diseño
5. Implementación
6. Prueba
7. Servicio

Modelo de cascada



Modelo de cascada

1) Análisis y definición de requerimientos: Los servicios, restricciones y metas
Del sistema se definen a partir de las consultas con los usuarios.
2) Diseño del sistema y del software: Se establece una arquitectura completa
Del sistema, el diseño del software identifica y describe los elementos
Abstractos que son fundamentales para el software y sus relaciones.
3) Implementación y pruebas unitarias: Durante esta etapa, el diseño del
Software se lleva a cabo como un conjunto de unidades de programas, la
Prueba unitaria implica verificar que ésta cumpla su función específica.
4) Integración y prueba del sistema: Los programas o las unidades individuale
De programas se integran y se prueban como un sistema completo, para así
Asegurar que se cumplan los requerimientos.
5) Funcionamiento y mantenimiento: En esta fase el sistema se instala y se
Pone en funcionamiento práctico. El mantenimiento implica corregir errores
No descubiertos en las etapas anteriores así como la implementación de
Nuevos requerimientos.

Modelo de Prototipos

El modelo de prototipos permite que todo el sistema o algunas de sus partes,
Se construyan rápidamente para comprender con facilidad y aclarar ciertos
Aspectos en los que se aseguren que el desarrollador, el usuario y el cliente
Estén de acuerdo en lo que se necesita, así como también la solución que se
Propone para dicha necesidad y de esta forma minimizar el riesgo y la
Incertidumbre en el desarrollo.
Este modelo se utiliza para dar al usuario una vista preliminar de parte
Del software. Se basa en prueba y error, ya que si al usuario no le gusta
Una parte del prototipo, significa que la prueba falló por lo cual se debe corregir
Hasta que el usuario quede satisfecho.
El prototipo debe ser construido en poco tiempo, usando los programas
Adecuados y no se debe utilizar mucho dinero.
El construir el prototipo nos asegura que nuestro software sea de mejor
Calidad, además de que su interfaz gráfica sea del agrado del usuario.

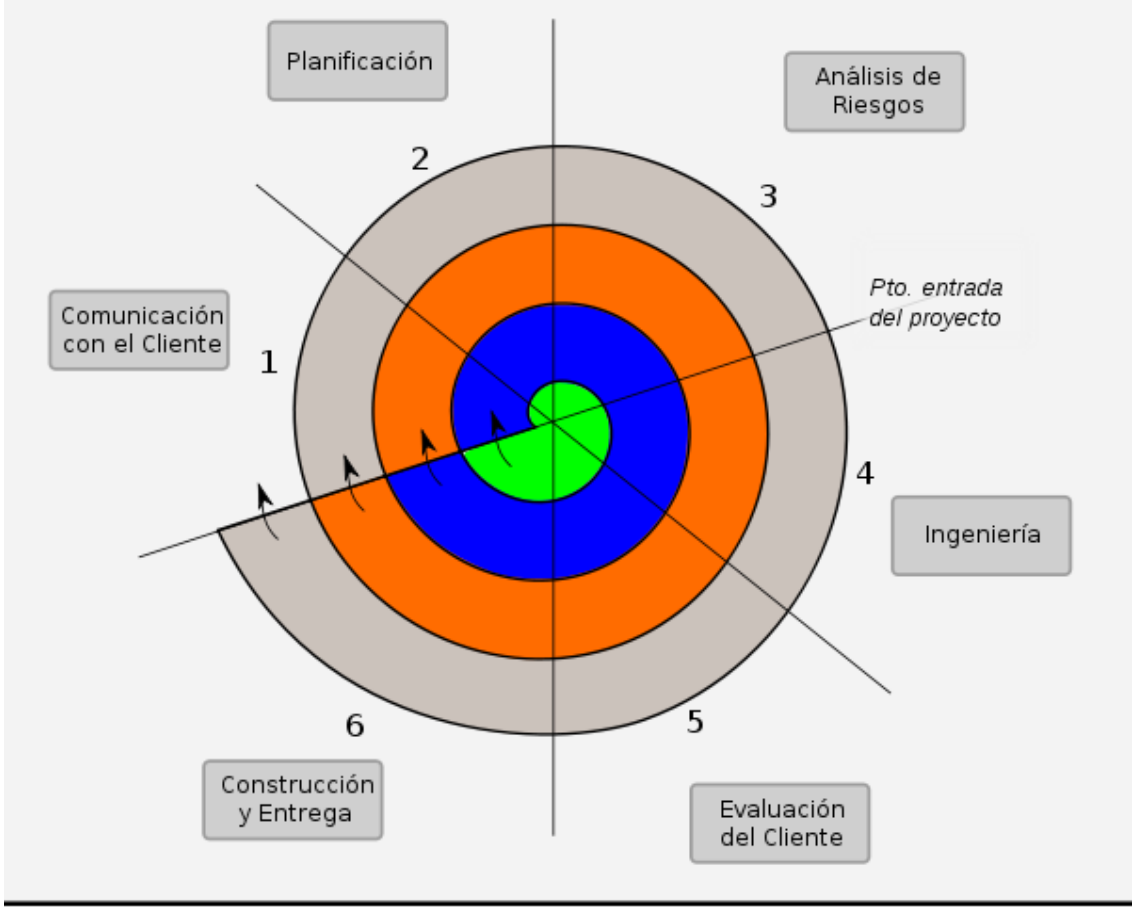
Modelo de Prototipos

Hay dos clases de prototipos:
- Desechable: Nos sirve para eliminar dudas sobre lo que realmente quiere El cliente, además para desarrollar la interfaz que más le convenga al Cliente.
- Evolucionario: Es un modelo parcialmente construido que puede pasar de Ser prototipo a ser software pero no tiene una buena documentación y calidad.

Modelo de Espiral

Éste modelo describe el ciclo de vida de un software por medio de espirales Que se repiten hasta que se puede entregar el producto terminado.
El desarrollo en espiral también se conoce como desarrollo o modelo Incremental. El producto se trabaja continuamente y las mejoras a menudo Tienen lugar en pasos muy pequeños. Cuando se aplica este modelo, el Software se desarrolla en una serie de entregas evolutivas. Cada una de las Actividades del marco de trabajo representan un segmento de la ruta en Espiral.
Este modelo se basa en la idea de desarrollar una implementación inicial, Exponiéndola a los comentarios del usuario y refinándola a través de las Diferentes versiones que se generan hasta que se desarrolle un sistema Adecuado.

Modelo de Espiral



Modelo ágil

Es un marco de trabajo conceptual de la ingeniería de software que promueve
Iteraciones en el desarrollo a lo largo de todo el ciclo de vida del proyecto.
Existen muchos métodos de desarrollo ágil, la mayoría minimiza riesgos
Desarrollando software en cortos lapsos de tiempo.
Se caracterizan por enfatizar la comunicación frente a la documentación,
Por el desarrollo evolutivo y por su flexibilidad.

Elementos clave

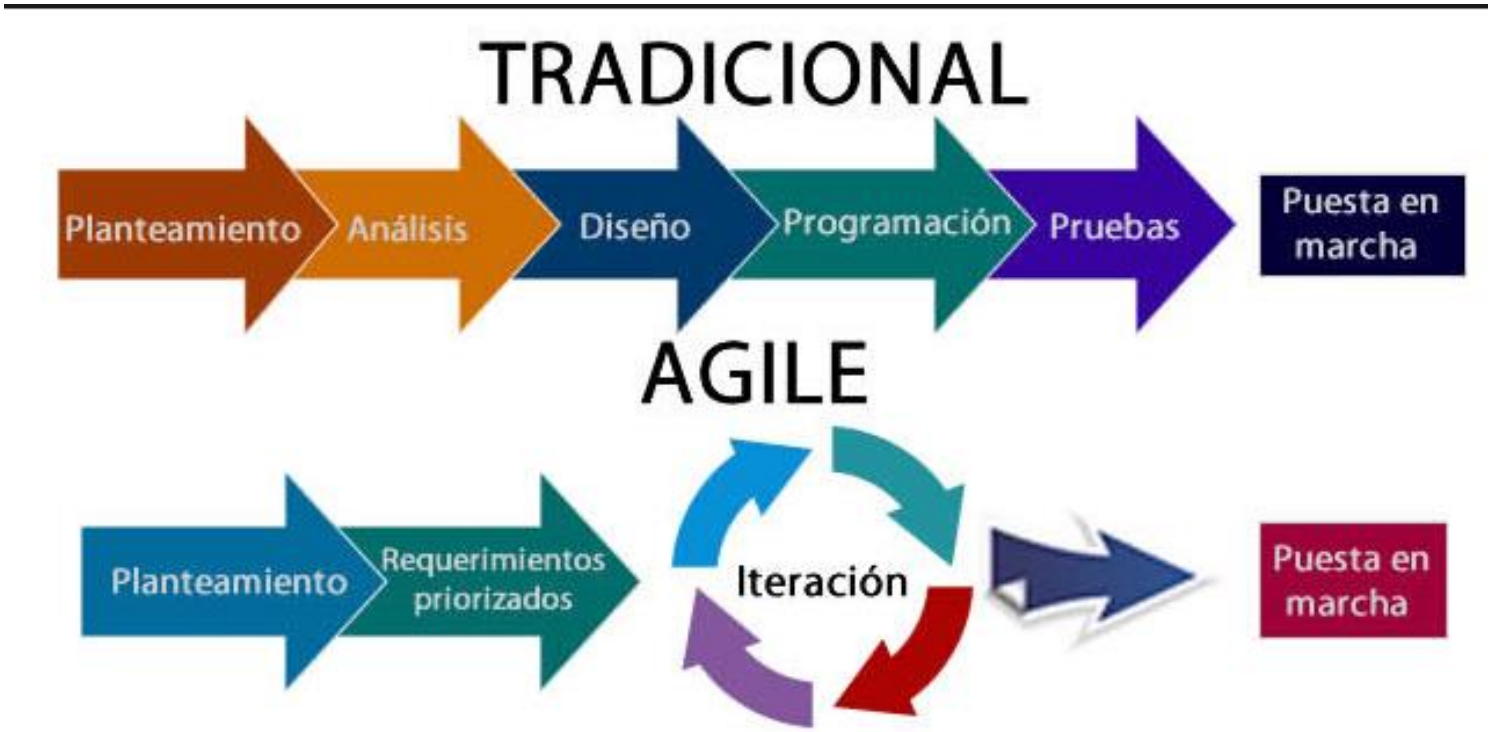
- **Individuos:** Toda la importancia hay que dársela a las personas, que deben
Permanecer en un primer plano.
- **Software funcionando:** Esto se debe a que había llegado un punto en el que l
Documentación de un trabajo había alcanzado tanta importancia como el objeto
De trabajo en sí mismo, el producto. Cuando realmente la mayor atención
Debe estar puesta siempre en lo que queremos construir y lo demás debería
Ser secundario.

Modelo ágil

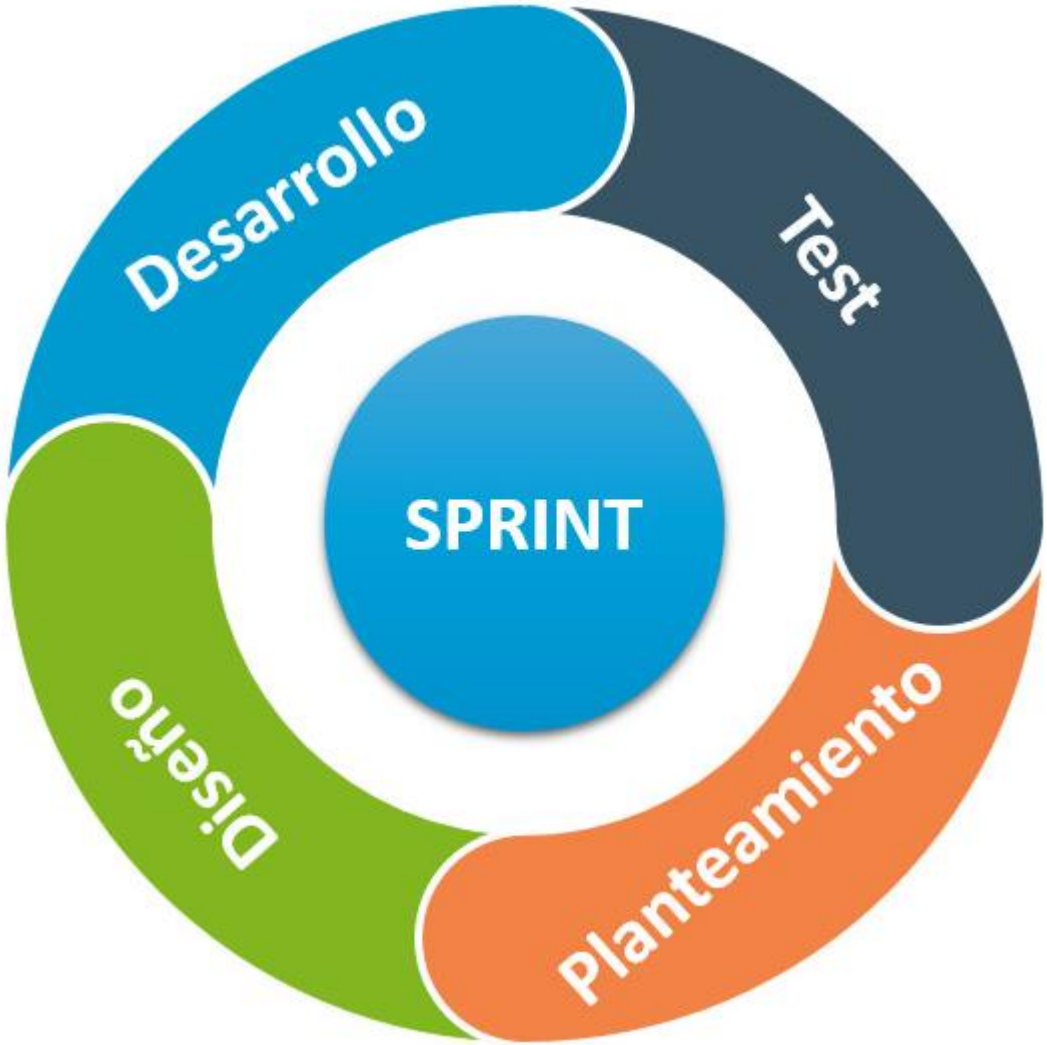
- **Colaboración del cliente:** A la hora de sacar un proyecto adelante, la forma Más productiva siempre será estableciendo un marco de colaboración y Confianza con quién nos lo encarga. Tanto cliente como el desarrollador Comparten objetivos e interéses similares.
- **Respuesta al cambio:** Se trata de apreciar la incertidumbre como un Componente básico del trabajo, de tal manera que la adaptabilidad y la Flexibilidad se convierten en virtudes y no en defectos de la manera De trabajar del equipo.

Éstos modelos deben cumplir con lo siguiente:
- Ser adaptables de forma incremental
- Tener abundante retroalimentación del cliente
- Basarse en la entrega continua de incrementos

Modelo ágil


TRADICIONAL
Planteamiento
Análisis
Diseño
Programación
Pruebas
Puesta en marcha
AGILE
Planteamiento
Requerimientos priorizados
Iteración
Puesta en marcha

Modelo ágil



Modelo ágil

METODOLOGÍA ÁGILE	METODOLOGÍAS TRADICIONALES
– Flexibilidad ante los cambios del proyecto de forma moderada a rápida – Los clientes hacen parte del equipo de desarrollo – Grupos pequeños (promedio 10 participantes in situ) en el mismo lugar. – Menor dependencia de la arquitectura de software – Continuo Feedback acortando el tiempo de entrega – Diversidad de roles – Basadas en heurísticas a partir de prácticas de producción de código – Procesos menos controlados, pocas políticas y normas – Capacidad de respuesta ante los cambios	– Rigidez ante los cambios, de manera lentos o moderada – Los clientes interactúan con el equipo de desarrollo mediante reuniones – Grupos de gran tamaño y varias veces distribuidos en diferentes sitios – Dependencia de la arquitectura de software mediante modelos – Poco Feedback lo que extiende el tiempo de entrega – Mínimos roles – Basadas en normas de estándares de desarrollo – Procesos muy controlados por políticas y normas – Seguimiento estricto del plan inicial de desarrollo

Modelo ágil

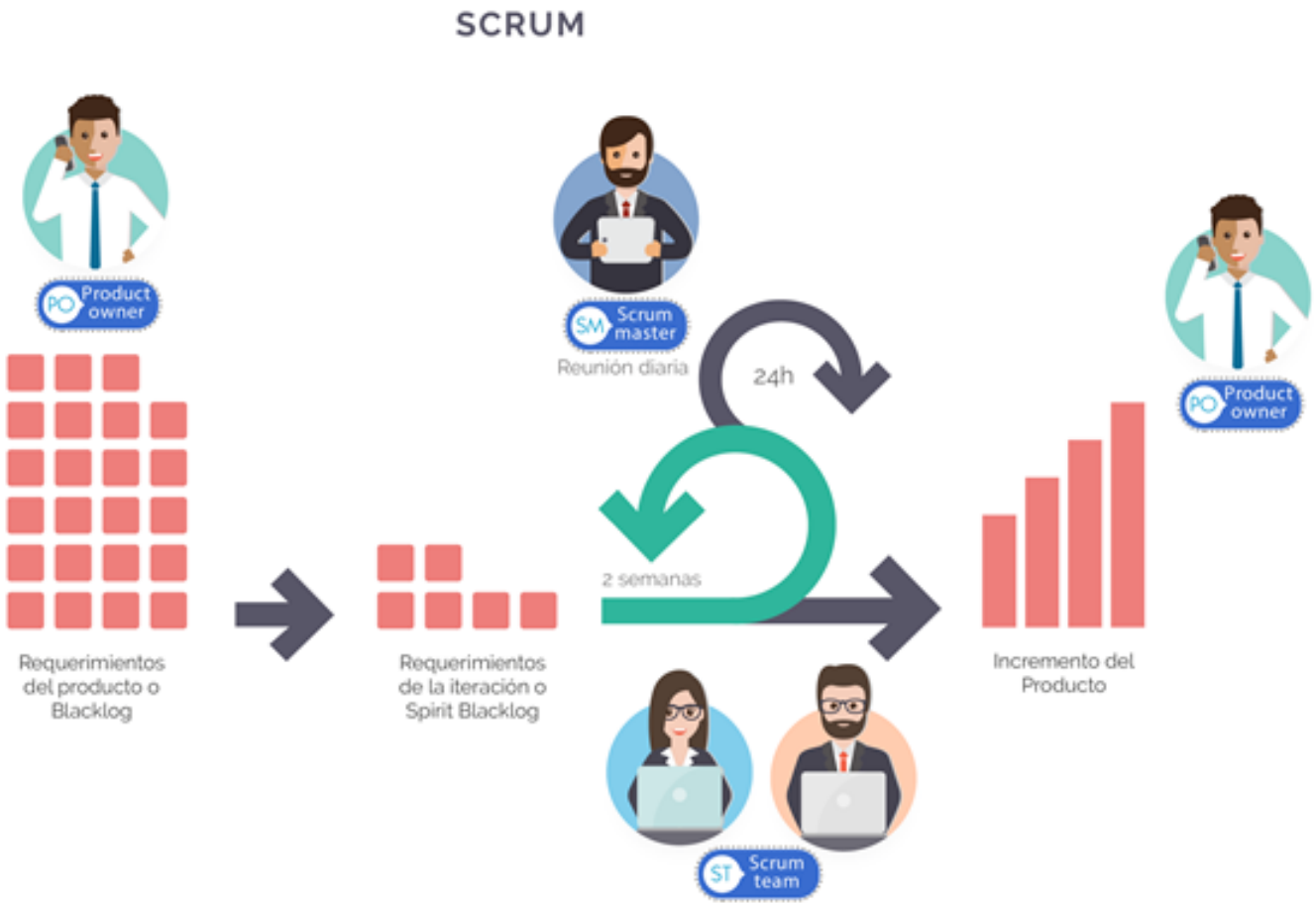
ALGUNOS TIPOS DE METODOLOGÍAS ÁGILES

1. METODOLOGIA AGILE SCRUM
2. PROGRAMACIÓN EXTREMA – XP
3. METODOLOGIA AGILE KANBAN

Modelo ágil - SCRUM

En Scrum se realizan entregas parciales y regulares del producto final,
Priorizadas por el beneficio que aportan al receptor del proyecto. Por ello,
Scrum está especialmente indicado para proyectos en entornos complejos,
Donde se necesita obtener resultados pronto, donde los requisitos son
Cambiantes o poco definidos, donde la innovación, la competitividad,
La flexibilidad y la productividad son fundamentales.
Scrum también se utiliza para resolver situaciones en que no se está
Entregando al cliente lo que necesita, cuando las entregas se alargan
Demasiado, los costos se disparan y la calidad no es aceptable, cuando
Se necesita capacidad de reacción ante la competencia, cuando la moral
De los equipos es baja y la rotación alta, cuando es necesaario identificar
Y solucionar ineficiencias sistemáticamente o cuando se quiere trabajar
Utilizando un proceso especializado en el desarrollo de un producto.

Modelo ágil

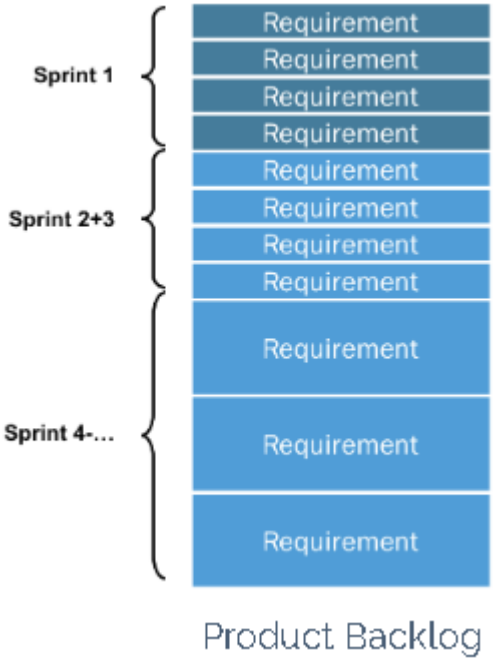

SCRUM
Product owner
Scrum master
Reunión diaria
24h
2 semanas
Product owner
Requerimientos del producto o Blacklog
Requerimientos de la iteración o Spirit Blacklog
Incremento del Producto
Scrum team

Modelo ágil - SCRUM

Product Backlog

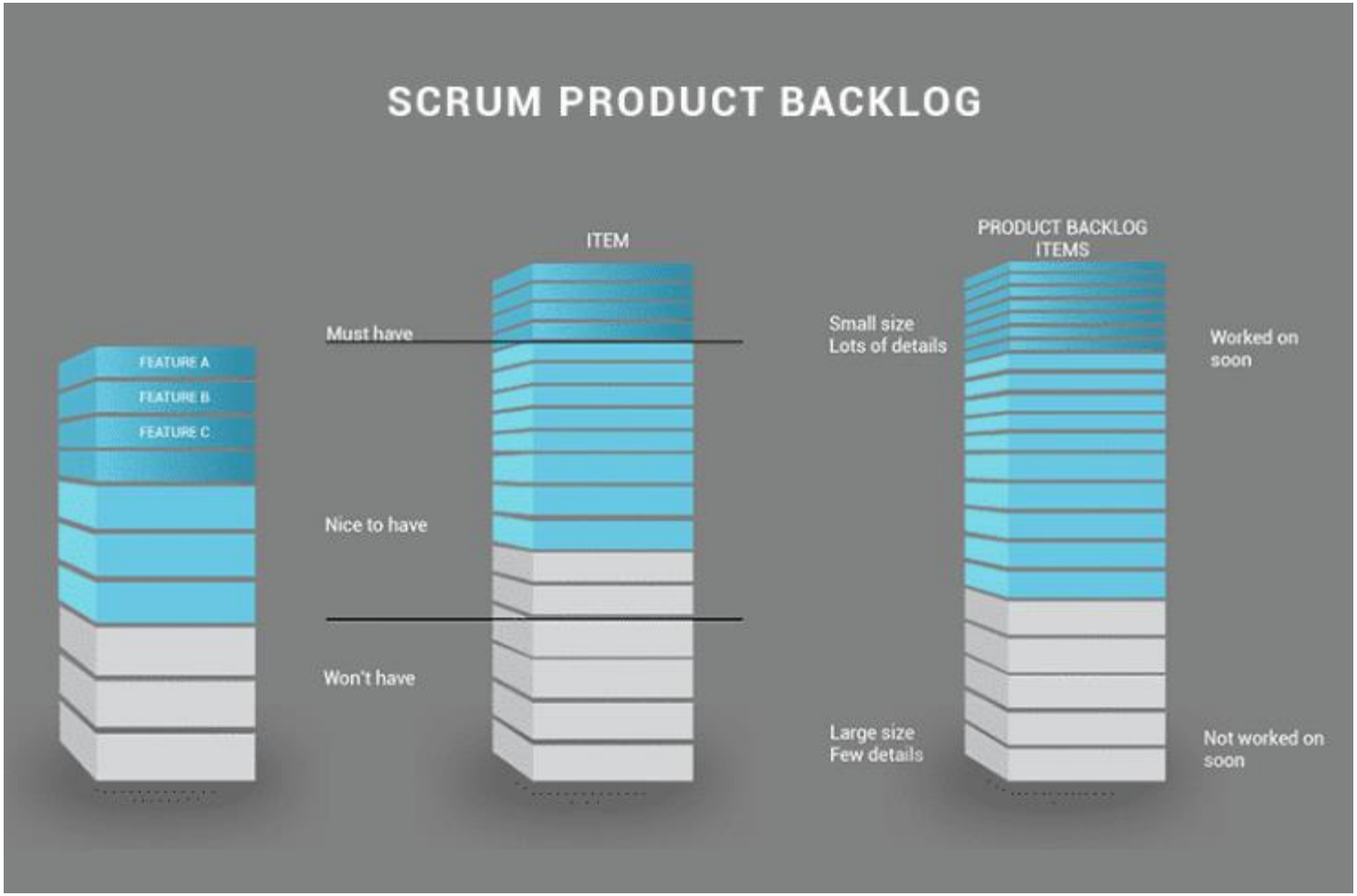
El Product Backlog es una lista ordenada de todo lo que se conoce que se Necesitará en el producto final. El Product Owner es el responsible del Product Backlog, incluyendo su contenido, disponibilidad y ordenamiento.

El Product Backlog es dinámico, constantemente tiene cambios acorde A lo que el producto va necesitando.



Product Backlog

Modelo ágil


SCRUM PRODUCT BACKLOG
ITEM
PRODUCT BACKLOG ITEMS
FEATURE A
FEATURE B
FEATURE C
Must have
Nice to have
Won't have
Small size
Lots of details
Large size
Few details
Worked on soon
Not worked on soon

Modelo ágil – Programación Extrema (XP)

Este concepto de programación extrema, conocido como XP, es un proceso
Ágil cuyo objetivo es mejorar la calidad del software y la capacidad de
Respuesta a las necesidades cambiantes de los clientes.
Se diferencia de las metodologías tradicionales principalmente en que pone
Más énfasis en la adaptabilidad que en la previsibilidad.
Con esto se pretende ser capaz de adaptarse a los cambios de requerimientos
En cualquier punto de la vida del proyecto para ser más realista que intentar
Definir todos los requisitos al comienzo del proyecto e invertir esfuerzos
Después en controlar los cambios en los requerimientos.

Simplicidad
Es la base de la programación extrema. Se simplifica el diseño para agilizar
El desarrollo y facilitar el mantenimiento. Un diseño complejo del código
Junto a sucesivas modificaciones por parte de diferentes desarrolladores
Hacen que la complejidad aumente exponencialmente.

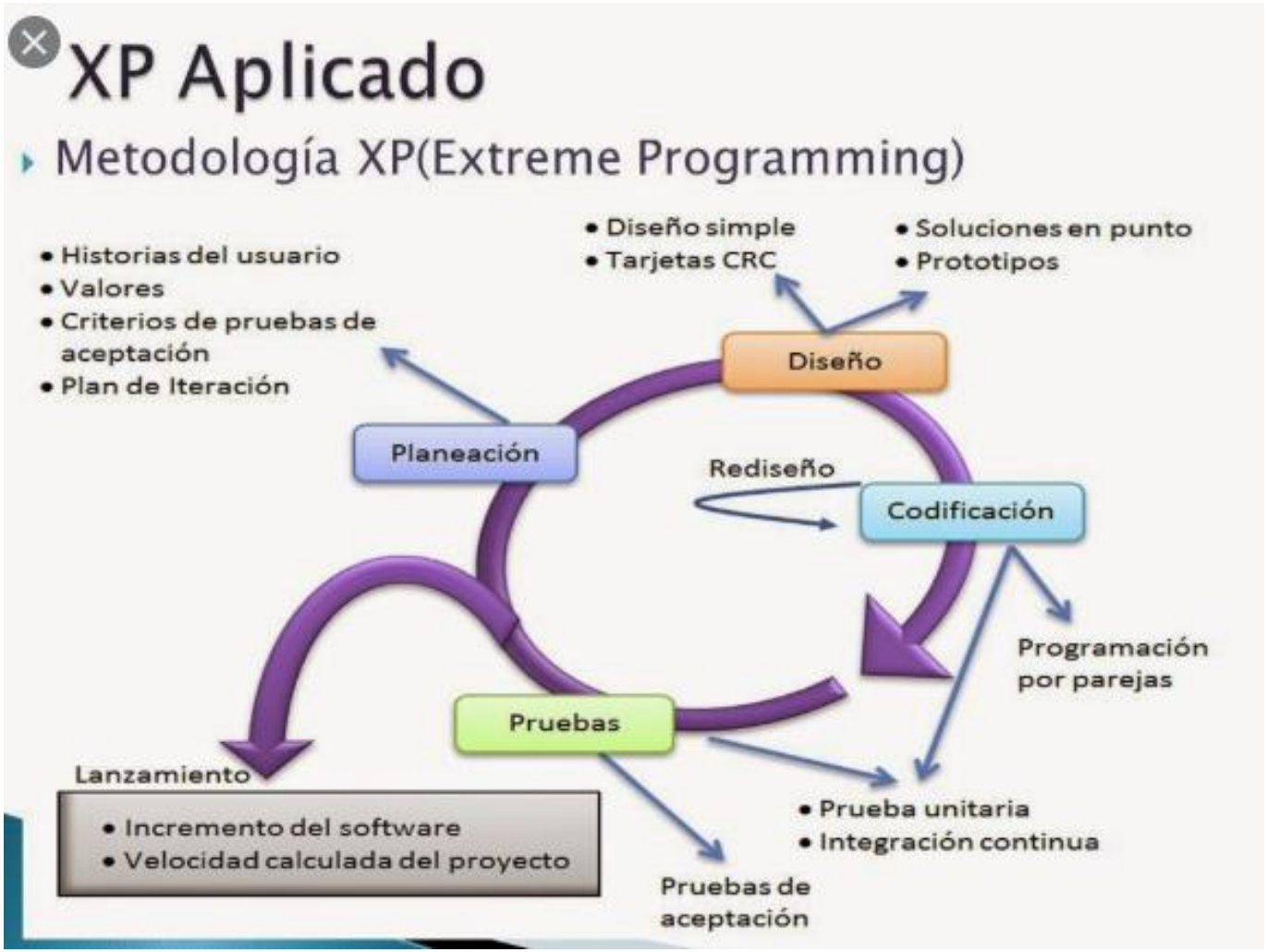
Modelo ágil – Programación Extrema (XP)

Comunicación
Para los programadores, el código comunica mejor cuanto más simple sea.
Si el código es complejo hay que esforzarse para hacerlo inteligible.
Es bueno comentar código, en especial solo aquello que no va a variar, por
Ejemplo el objetivo de una clase o la funcionalidad de un método.

Las pruebas unitarias son otra forma de comunicación ya que describen el
Diseño de las clases y los métodos al mostrar ejemplos concretos de como
Utilizar su funcionalidad.

La comunicación con el cliente es fluida ya que el cliente forma parte del equipo
De desarrollo. El cliente decide que características tiene prioridad y siempre
Debe estar disponible para solucionar dudas.

Modelo ágil


XP Aplicado
Metodología XP(Extreme Programming)
• Historias del usuario
• Valores
• Criterios de pruebas de aceptación
• Plan de Iteración
• Diseño simple
• Tarjetas CRC
• Soluciones en punto
• Prototipos
Diseño
Planeación
Rediseño
Codificación
Programación por parejas
Pruebas
Lanzamiento
• Incremento del software
• Velocidad calculada del proyecto
• Prueba unitaria
• Integración continua
Pruebas de aceptación

Modelo ágil – Feature-driven development

Feature-driven development (FDD) es un proceso iterativo e incremental
Para el desarrollo de software.
Su principal propósito es entregar software tangible y funcional repetidamente
De manera oportuna de acuerdo con los principios detrás del modelo Ágil
Centrada en el cliente.

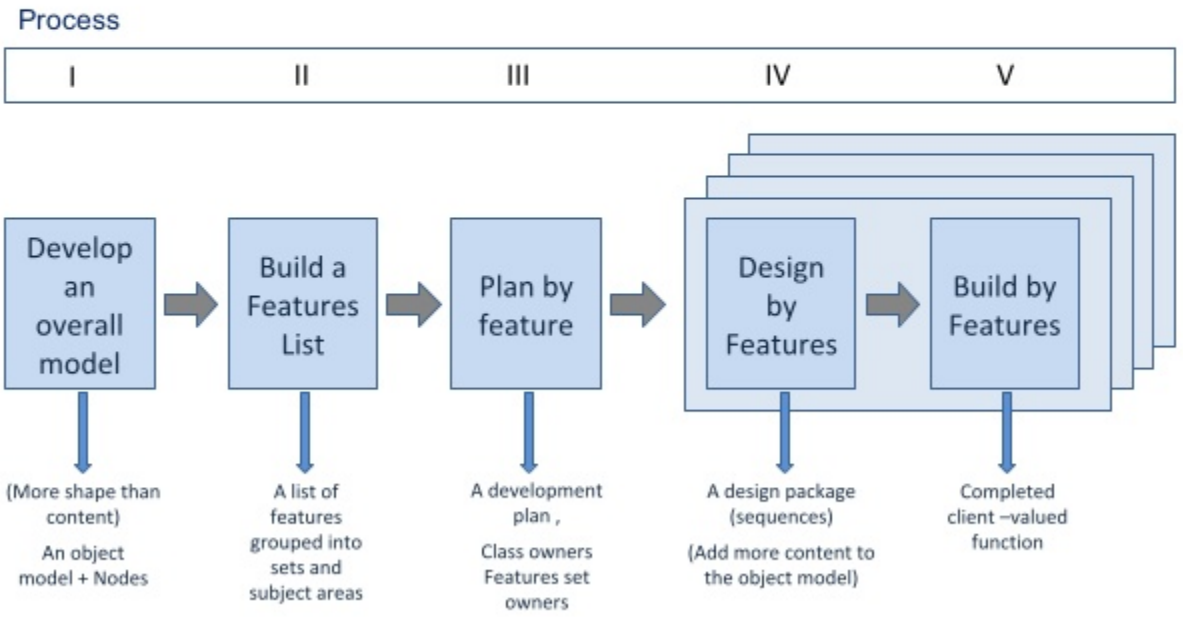
FDD permite a los equipos actualizar el proyecto regularmente e identificar
Los errores rápidamente. Además, los clientes pueden. FDD es un método
Favorito entre los equipos de desarrollo porque ayuda a reducir dos mortales
Conocidos en el mundo del desarrollo: confusión y retrabajo.

Mientras Scrum y nuevas variaciones de metodologías Ágiles son ampliamente
Reconocidas en la actualidad, FDD puede ser una buena opción para el
Desarrollo de software para equipos que buscan una metodología ágil
Estructurada y enfocada que se pueda escalar en toda la organización
Y que brinde resultados claros.

Modelo ágil

Processes

FDD consist five processes



Modelo ágil – Feature-driven development

¿Cuál es la diferencia entre FDD y Scrum?

FDD está relacionado a Scrum, pero como su nombre lo indica, es un método Centrado en características o funciones (en lugar de un método centrado en la entrega), en otras palabras “completar el proceso de login” será Considerado como una característica en el FDD.
Las funciones o características son una pieza fundamental de FDD: son para FDD lo que las historias de usuario son para Scrum.

“Durante la FDD, se debe entregar una función cada 2-10 días, que difiere de Scrum, en el cuál los sprints suele durar dos o a veces cuatro semanas”

FDD valora la documentación más que otros métodos (Scrum y XP incluidos), Lo que también crea diferencias en los roles de las reuniones. En Scrum, Los equipos generalmente se reúnen a diario, en FDD, los equipos dependen De la documentación para comunicar información importante y por lo tanto, No se reúnen con tanta frecuencia. Otra diferencia es que en FDD, el usuario Real es visto como el usuario final, mientras que en Srum el Product Owner Es visto como el usuario final.

Ingeniería de Requerimientos

La ingeniería de requerimientos es el proceso de definir, documentar y Mantener requerimientos en el proceso de diseño de ingeniería. Trata de Recopilar, analizar y verificar las necesidades del cliente para un sistema.

La primera vez que se usó el término de ingeniería de requerimientos fue Probablemente en 1964 en la conferencia de “Maintanance, Maintainability, And System Requirements Engineering”, pero no entró en uso hasta después En los 90s.

Cuando nos encontramos al frente de un proyecto de desarrollo de sistemas Es importante dejar claramente definidos los requerimientos del software, En forma consistente y compacta, esta tarea es difícil básicamente porque Consiste en la traducción de unas ideas vagas de necesidades de software En un conjunto concreto de funciones y restricciones. Además el analista Debe extraer información dialogando con varias personas y cada una de ellas Se expresará de una forma distinta, tendrá conocimientos informáticos Y técnicos distintos y tendrá unas necesidades y una idea del proyecto muy Particulares.

Ingeniería de Requerimientos – Requerimientos funcionales

Los requerimientos funcionales de un sistema, son aquellos que describen Cualquier actividad que este deba realizar.
Por lo general, estos deben incluir funciones desempeñadas por pantallas Específicas, descripciones de los flujos de trabajo a ser desempeñados por el Sistema y otros requerimientos de negocio, cumplimiento, seguridad u otra Índole.

Como se describen y clasifican los requerimientos funcionales

- Descripciones de los datos a ser ingresados en el sistema.
- Descripciones de las operaciones a ser realizadas por cada pantalla.
- Descripción de los flujos de trabajo realizados por el sistema.
- Descripción de los reportes del sistema y otras salidas.
- Definición de quien puede ingresar datos en el sistema.
- Como el sistema cumplirá los reglamentos y regulaciones de sector o
- generales que le sean aplicables.

Ejemplo de requerimientos funcionales de proceso o área de negocio:

- El sistema enviará un correo electrónico cuando se registre alguna de las
- siguientes transacciones: pedido de venta al cliente, despacho de mercancía
- al cliente, emisión de factura al cliente y registro de pago del cliente.
- Se permitirá el registro de pedidos de compra con datos obligatorios
- incompletos, los cuales podrán completarse posteriormente modificando el
- pedido. Antes de poder aprobarse los datos del pedido deben estar completos
- A cada orden se le asignará un identificador único, que será utilizado para
- identificarla en todos los procesos subsecuentes que se realicen sobre esta.
- Al ingresar ordenes de entrega, toda orden de entrega estará asociada
- a un pedido de venta.
- La facturación de pedidos de venta se realizará en lotes, por medio de una
- pantalla de pedidos pendientes de facturación, la cual mostrará los pedidos
- no facturados. Una vez facturados los pedidos no se mostrarán en esta lista.

Ejemplo de requerimientos funcionales de interfaz gráfica:

- El campo de monto acepta únicamente valores numéricos con dos decimales
- El campo fecha de transacción acepta únicamente fechas anteriores al día
- de hoy (día actual)
- El campo nombre acepta caracteres alfabéticos únicamente.
- El campo país consistirá en una lista de preselección. El país asociado a una
- dirección debe ser previamente registrado en el sistema.
- La pantalla de registro de pago puede imprimir los datos en pantalla a la
- impresora.

Ejemplo de requerimientos funcionales de seguridad:

- El sistema controlará el acceso y lo permitirá solamente a usuarios
- autorizados. Los usuarios deben ingresar al sistema con un nombre de
- usuario y contraseña.
- El sistema enviará una alerta al administrador del sistema cuando ocurra
- alguno de los siguientes eventos:
- 1. Registro de una nueva cuenta
- 2. Ingreso al sistema por parte del cliente
- 3. Cambio de contraseña del usuario

Los requerimientos no funcionales son los que se tratan de requisitos que no
Se refieren directamente a las funciones específicas suministradas por
El sistema, sino a las propiedades del sistema como rendimiento, disponibilidad,
Diseño, tiempos de respuesta, capacidad de almacenamiento, etc.

Estos se originan en la necesidad del usuario, debido a restricciones
Presupuestarias, políticas organizacionales, la necesidad de interoperabilidad
Con otros sistemas de software o hardware, o factores externos tales como
Regulaciones de seguridad, políticas de privacidad, entre otros.
En palabras más sencillas, no hablan de “lo que” hace el sistema, sino de
“cómo” lo hace.

Ejemplos de requerimientos no funcionales de producto

Eficiencia

- El sistema debe ser capaz de procesar N transacciones por segundo. Esto Se medirá por medio de la herramienta SoapUI aplicada al Software Testing De servicios web.

- Toda funcionalidad del sistema y transacción de negocio debe responder Al usuario en menos de 5 segundos.

- El sistema debe ser capaz de operar adecuadamente con hasta 100,000 Usuarios con sesiones concurrentes.

Ejemplos de requerimientos no funcionales de producto

Seguridad lógica y de datos

- El nuevo sistema debe desarrollarse aplicando patrones y recomendaciones De programación que incrementen la seguridad de los datos.
- Todos los sistemas deben respaldarse cada 24 hrs. Los respaldos deben Ser almacenados en una localidad segura ubicada en un edificio distinto Al que reside el sistema.
- Los integrantes del grupo de usuarios de analistas pueden ingresar solicitudes pero no pueden aprobarlas o borrarlas.
- Cualquier intercambio de datos vía internet que realice el software se realizará por medio del protocolo encriptado https.

Ejemplos de requerimientos no funcionales de producto

Usabilidad

- El tiempo de aprendizaje del sistema por un usuario deberá ser menos de 4 hrs.
- El sistema debe contar con manuales de usuario estructurados adecuadamente
- El sistema debe contar con un módulo de ayuda en línea.
- El sistema debe poseer interfaces gráficas bien formadas.
- El sistema debe usar los colores representativos de la empresa.

Son declaraciones en el lenguaje natural y en diagramas de los servicios
Que se espera que el sistema provea y de las restricciones bajo las cuales
Debe operar.
Describen los requerimientos funcionales y no funcionales del tal forma
Que sean comprensibles por los usuarios del sistema que no posean un
Conocimiento técnico detallado. Únicamente especifican el comportamiento
Externo del sistema y evitan, tanto como sea posible, las características de
Diseño del sistema. Por consiguiente, los requerimientos de usuario no se deber
Definir utilizando un modelo de implementación. Deben redactarse utilizando
El lenguaje natural, representaciones y digramas intuitivos sencillos.

Ejemplos

- El usuario debe poder dar de alta un producto ingresando únicamente Nombre, categoría y precio.
- El usuario debe ser capaz de generar un reporte mensual de las compras Ingresadas en el sistema por mes.
- El usuario debe ser capaz de generar una factura después de haber Registrado la venta.
- El usuario debe ser capaz de ver una tabla con todo el inventario registrado Al momento, indicando el número de productos que se tienen.

FORMATO LEVANTAMIENTO DE REQUERIMIENTOS POR HISTORIAS DE USUARIO

EJEMPLO NUEVO SERVICIO WEB

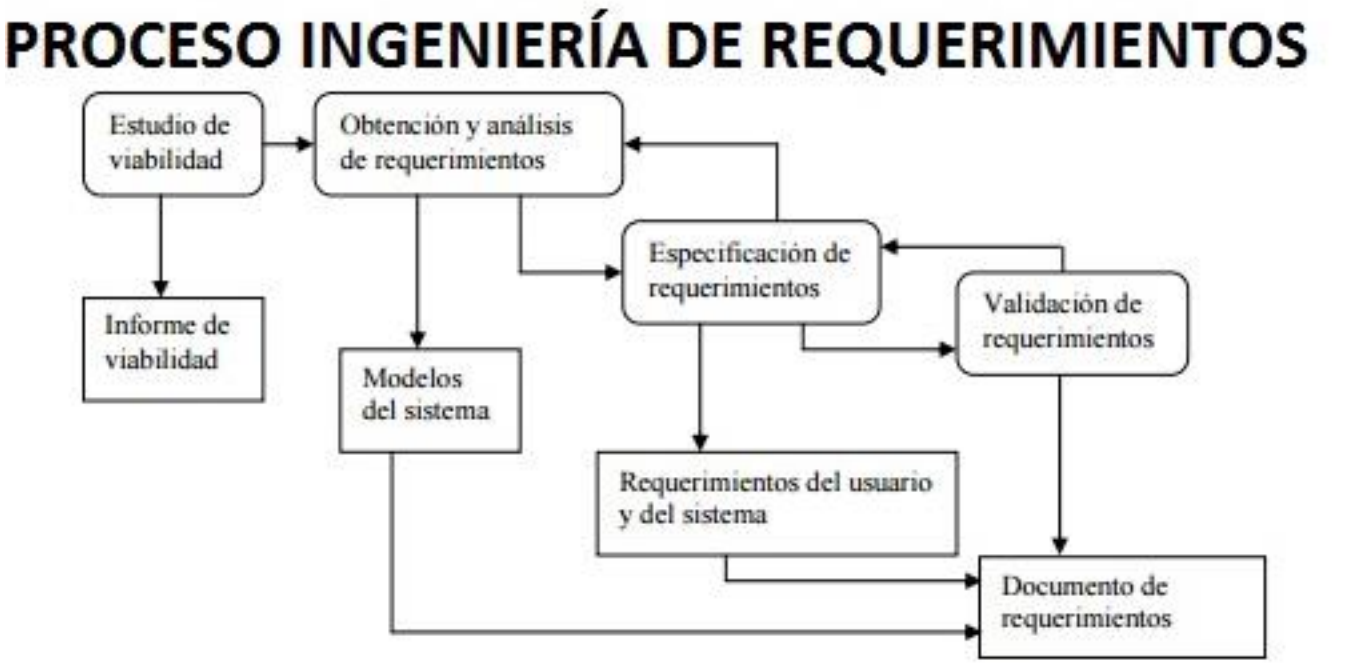
Versión:	1.0	**Solicitante:**	Juan David

HISTORIA DE USUARIO PRINCIPAL

COMO	NECESITO	PARA
Escritor del portal web	Publicar los artículos de forma automática.	Evitar la dependencia por parte del área de soporte para subir el artículo en el servidor.

¿Cómo opera el sistema actualmente?	*Esta casilla se debe diligenciar cuando el tipo de requerimiento corresponde a un ajuste.*
¿Qué problema ayudará a resolver?	Cada vez que se crea un artículo desde la aplicación de escritorio se debe subir todos los artículos en el servidor de forma manual.
Prototipo:	N/A
Flujo de proceso:	

Ingeniería de Requerimientos – Proceso de ingeniería de requerimientos



Obtención de Requerimientos

Existe un gran número de técnicas para obtener requerimientos. A continuación
Se describen las más utilizadas. Acabe aclarar que ninguna de las técnicas
Es suficiente por sí sola y que es recomendable cominarlas para obtener
Requerimientos completos.

Entrevistas

La entrevista es de gran utilidad para obtener información cualitativa como
Opiniones o descripciones subjetivas de actividades. Es una técnica muy utilizad
Y requiere una mayor preparación y experiencia por parte del analista.
La entrevista se puede definir como un “intento sistemático de recoger
Información de otra persona” a través de una comunicación interpersonal
Que se lleva a cabo por medio de una conversación estructurada.

Obtención de Requerimientos

Desarrollo de prototipos

Los prototipos suelen consistir en versiones reducidas, demos o conjunto
De pantallas (que no son totalmente operativas) de la aplicación solicitada.
Ésta técnica es particularmente útil cuando:

- El área de la aplicación no está bien definida (posiblemente por ser algo muy Novedoso)
- El costo del rechazo de la aplicación por los usuarios es muy alto.
- Es necesario evaluar previamente el impacto del sistema en los usuarios y En la organización.

Los prototipos de sistema permiten a los usuarios experimentar para ver como Éste ayuda a su trabajo. Fomentan el desarrollo de ideas que desembocan En requerimientos.

Obtención de Requerimientos

Observación

Por medio de esta técnica el analista obtiene información de primera mano Sobre la forma en que se efectúan las actividades. Este método permite Observar la forma en que se llevan a cabo los procesos y por otro verificar Que realmente se sigan todos los pasos especificados. Como sabemos, En muchos casos los procesos son una cosa en papel y otra muy diferente En la práctica.

Estudio de documentación

Varios tipos de documentación, como manuales y reportes, pueden proporcionar Al analista información valiosa con respecto a las organizaciones y a sus Operaciones. La documentación difícilmente refleja la forma en que realmente Se desarrollan las actividades o donde se encuentra el poder de la toma De deciciones. Sin embargo, puede ser de gran importancia para introducir Al analista al dominio de operación y el vocabulario que utiliza.

Obtención de Requerimientos

Cuestionarios

El uso de cuestionarios permite a los analistas reunir información proveniente De un grupo grande de personas. El empleo de formatos estandarizados para Las preguntas pueden proporcionar datos más confiables que otras técnicas. El inconveniente es que la respuesta puede ser limitada ya que es posible Que no tenga mucha importancia para los encuestados llenar el cuestionario.

Tormenta de ideas (Brainstorming)

Consiste en reuniones con cuatro a diez personas donde como primer paso Alguien sugiere toda clase de ideas sin juzgar su validez, por muy disparaadas Que parezcan y después de recopilar todas las ideas se realiza un análisis Detallado de cada propuesta. Esta técnica se puede utilizar para identificar Un primer conjunto de requisitos en aquellos casos donde no están muy claras Las necesidades que hay que cubrir.

Análisis de Requerimientos

Es el conjunto de técnicas y procedimientos que nos permiten conocer los Elementos necesarios para definir un proyecto de software. Es una tarea de Ingeniería del software que permite especificar las características operacionales Del software, indicar la interfaz del software con otros elementos del sistema Y establecer las restricciones que debe cumplir el software.

El análisis de requerimientos proporciona una vía para que los clientes y los Desarrolladores lleguen a un acuerdo sobre lo que debe hacer el sistema. La especificación, producto de este análisis proporciona las pautas a seguir A los diseñadores del sistema.

En esta etapa, se analisan los siguientes aspectos:

Análisis de Requerimientos

- Deben ser correctos: Tanto el cliente como el desarrolador deben revisarlos Para asegurar que no tienen errores.
- Deben ser consistentes: Dos requerimientos son inconsistentes cuando es Imposible satisfacerlos simultáneamente.
- Deben estar completos: El conjunto de requerimientos está completo si todos Los estados posibles, cambios de estado, entradas, productos y restricciones Están descritos en alguno de los requerimientos.
- Deben ser realistas: Todos los requerimientos deben ser revisados para Asegurar que son posibles.
- Deben ser verificables: Se deben poder preparar pruebas que demuestren Que se han cumplido los requerimientos.
- Deben ser rastreables: ¿Se puede rastrear cada función del sistema hasta El conjunto de requerimientos que la establece?

Especificación de requerimientos del sistema (ERS)

El objetivo principal del ERS, es servir como medio de comunicación entre Clientes, usuarios, ingenieros de requisitos y desarrolladores. En el ERS Deben recogerse tanto las necesidades de clientes y usuarioscomo los Requisitos que debe cumplir el sistema a desarrollar para satisfacer dichas Necesidades.
El ERS es el principal producto del proceso de la ingeniería de requermientos, Resultado del levantamiento de información con el usuario o cliente del producto Es un método para una comunicación más concisa y clara entre los encargados De desarrollar el software y el área de negocio o clientes que usarán el producto

Estructura del Documento de Requerimientos

- **Definición de Requerimientos No-funcionales**.
  - Definir las limitantes del sistema y el proceso de desarrollo.
- **Evolución del Sistema**.
  - Definir las suposiciones fundamentales en las cuales el sistema se basa y se anticipan los cambios.
- **Especificación de Requerimientos**.
  - Especificación detallada de los requerimientos funcionales del sistema.
- **Apéndices**.
  - Descripción de la plataforma de Hardware del Sistema.
  - Requerimientos de la base de Datos (quizá como un modelo ER)
- **Indice.**

Ingeniería de Requerimientos – Proceso de ingeniería de requerimientos

it-Mentor ITM

"capacitación y guía para el desarrollo de software"

1 Descripción general

1.1 OBJETIVOS

Este documento contiene la Especificación de Requerimientos del Software (ERS) destinado al proyecto **Ejemplo Proceso de Desarrollo** de la empresa **EMPRESA**.

1.2 DOCUMENTOS RELACIONADOS

En esta sección se indican todos los documentos entregados por el cliente. A medida que se entreguen nuevos documentos que se relacionen al presente documento, se agregarán a la lista que se detalla a continuación.

Documento	Carpeta	Nombre Documento
Descripcion del sistema a desarrollar		**DescripcionSistemaaDesarrollar.doc**

1.3 DESTINATARIOS

En esta sección se indica a cuales personas o roles está dirigido este documento.

Lector	Sector o Rol
Lector 1	Líder de Proyecto de **EMPRESA**
Lector 2	Lider de Proyecto
Lector 3	Gerente de Proyecto
Lector N	Equipo de Desarrollo

1.4 PARTICIPANTES

La tabla siguiente lista todos los miembros involucrados con distintos roles en este proyecto.

Rol	Recurso	Responsabilidad	Reemplazo de recurso
Líder de Proyecto		Gestión y seguimiento del proyecto. Coordinación del trabajo del equipo de desarrollo, asignación de tareas, seguimiento	
Analista		Relevamiento, análisis y modelado de requerimientos. Diseño de casos de prueba y análisis de resultados.	
Líder de Proyecto EMPRESA		Gestión del proyecto	

GOBIERNO DE COLOMBIA — UNIDAD PARA LAS VÍCTIMAS

FORMATO DE ESPECIFICACIÓN DE REQUERIMIENTOS DE SOFTWARE			
PROCESO GESTIÓN DE LA INFORMACIÓN			
PROCEDIMIENTO: DESARROLLO DE SISTEMAS DE INFORMACIÓN			
Código: 130,06,15-27	Versión: 1	Fecha: 20/12/2018	Página 2 de 9

1. DESCRIPCION GENERAL DEL REQUERIMIENTO

PROYECTO	Diligenciar el nombre del proyecto por parte del área o proceso solicitante
Nombre Requerimiento:	Diligenciar el nombre del proyecto o desarrollo de software por parte del área o proceso solicitante
Fecha Solicitud:	DD/MM//AAAA
Responsable(s) Solicitud:	Nombre del responsable de la solicitud
Dependencia(s) Solicitante:	Nombre del área o dependencia a la que hace parte el responsable de la solicitud
Responsable Funcional designado por el equipo de desarrollo de software:	Nombre del responsable del análisis funcional de la solicitud, establecido por el equipo de desarrollo de software

2. FASE DE FORMALIZACIÓN

Descripción de la Solicitud
Usuario Solicitante
El usuario solicitante debe diligenciar este campo dando una definición detallada, clara y concisa de la solicitud evitando ambigüedades y utilizando lenguaje natural y herramientas que crea pertinentes, tales como gráficos, diagramas, tablas, catálogos.
Lider Funcional
El líder funcional del equipo de desarrollo de software debe diligenciar este campo dando una definición detallada, clara y concisa de lo que entendió de la solicitud evitando ambigüedades y utilizando lenguaje natural y herramientas que crea pertinentes, tales como gráficos, diagramas, tablas, catálogos.

Validación

El proceso de validación consiste en analizar y comprobar si el Sistema cumplirá con los requerimientos establecidos y las necesidades del Cliente, de manera que se pueda llegar a identificar errores e inconsistencias En la especificación de los requerimientos.
Mediante la validación se logra detectar si se omitió algún requerimiento o, si Al contrario, se agregó algo que no pidió el cliente y simplemente consumirá Tiempo y recursos innecesariamente.
Una ventaja de la validación es que permite saber que si el cliente pidió un Sistema “azul” no se le entregará un sistema “rojo”, hipotéticamente hablando.