Prototipos

El uso de prototipos ayuda a que los requerimientos sean más tangibles, a achicar la brecha entre la concepción del sistema por parte del cliente/usuario y el analista. Ayudan a los usuarios y desarrolladores a visualizar cómo los requerimientos deben ser implementados, lo cual puede revelar deficiencias en los requerimientos.

Un prototipo puede ser:

- Un experimento o simulación del producto final.
- Una porción o modelo del sistema real.

El feedback temprano sobre los prototipos ayuda a establecer el acuerdo base, reduciendo el riesgo de no satisfacer las expectativas de los clientes/usuarios.

Propósito:

Aclarar, completar y validar requerimientos: Permiten encontrar errores, omisiones, identificar malentendidos, evaluar calidad.

Explorar alternativas de diseño: Mejoran la usabilidad y la eficiencia, confirmar la visión del producto antes de construir la solución.

Crear un subproducto que evolucione hacia el producto final: Permite crear funcionalidades no entregables y hacer evolucionar el producto con estas a través de ciclos cortos hasta alcanzar un producto entregable.

Atributos o Descriptores:

Alcance → Maqueta o prueba de concepto.

Uso futuro → Descartable o evolutivo.

Formato → Papel o electrónico.

Alcance:

Maqueta (Mock-up)

- Hace foco en la experiencia de usuario.
- No hace hincapié en la arquitectura ni en la funcionalidad detallada.
- Muestra la interfaz del usuario, pantallas y su navegación, con muy poca o ninguna funcionalidad.
- Los usuarios pueden juzgar si podrán hacer su trabajo y refinar sus requerimientos.

Prueba de concepto (proof-of-concept)

- Implementa una porción de las funcionalidades del sistema.
- Funciona como va a funcionar el sistema real.
- Útil cuando se desea testear si la arquitectura propuesta es la correcta, cuando se desea optimizar algoritmos, evaluar un esquema de base de datos, confirmar la solidez de una solución existente en la nube, o testear requisitos críticos de sincronización.

Uso futuro:

Descartable (Throwaway prototype)

- Se descarta luego de ser utilizado para generar feedback.
- Hace énfasis en la implementación rápida para testear performance, robustez, confiabilidad, etc.
- Está enfocado en una porción del sistema.
- No debería evolucionar.
- Apropiado cuando hay incompletitud en los requerimientos o ambigüedades.

Evolutivo (Evolutionary prototype)

- Brinda una base para continuar construyendo sobre el prototipo de forma incremental.
- Crece hasta el producto final.
- La primera iteración de un prototipo evolutivo puede pensarse como una entrega piloto.
- Toma más tiempo que un prototipo descartable.

Formato:

Papel (Paper prototype)

- Dibujo en papel o herramienta de dibujo.
- Ayuda a probar si los usuarios y desarrolladores comparten las mismas ideas de los requerimientos.
- Es barato y rápido.
- Útiles para explorar funcionalidades y el flujo del sistema.

Electrónico (Electronic prototype)

- Sirve en especial para el maquetado de interfaces gráficas.
- Sirve para implementar y modificar interfaces gráficas, independientemente de la eficiencia del código.

Mapa de diálogo:

Modela el diseño de una interfaz de usuario a un alto nivel de abstracción. Muestra los elementos asociados a las tarea de los usuarios y los links de navegación entre ellos.

Evaluación de prototipos:

Elegir usuarios representativos. Observar cómo los usuarios interactúan con el prototipo. Crear guías. Elaborar encuestas.

Riesgos:

Presión para lanzar el producto. Generación de falsas expectativas de performance. El prototipo reemplaza a la especificación de requerimientos ← Nunca debe pasar!

Factores de éxito:

Crear prototipos descartables mientras más rápido y barato sea. No prototipar requerimientos que ya son comprendidos. Usar datos creíbles. No esperar que un prototipo reemplace un requerimiento escrito.

Atributos de calidad

Desde los primeros encuentros el analista captura las necesidades de clientes y usuarios.

¿Qué?: Requerimientos funcionales especifican el comportamiento del sistema.

¿Cómo?: Los **atributos de calidad** establecen las cualidades del comportamiento.

Importancia:

Influyen en la satisfacción que brinda el producto al cliente → Pueden determinar el éxito del proyecto.

Originan requerimientos funcionales.

Es más complicado rediseñar un sistema para que cumpla el estándar de calidad deseado que hacerlo desde un principio.

Deben abordarse las expectativas de calidad de clientes y usuarios.

Clasificación:

Externos:

- Son percibidos y valorados por el usuario.
- Describen características observables.
- Influyen en la satisfacción.
  - → Disponibilidad: La medida en que los servicios del sistema están disponibles.
  - → Instalabilidad: Cuán fácil es instalar y desinstalar y reinstalar correctamente el sistema.
  - → Integridad: Medida en que el sistema protege la pérdida de datos.
  - → Interoperabilidad: Con qué facilidad el sistema puede interconectarse e intercambiar datos con otros sistemas.
  - → Rendimiento: Cuán rápidas y predecibles son las respuestas del sistema a las entradas del usuario u otros eventos.
  - → Confiabilidad: Cuánto tiempo funciona el sistema antes de experimentar una falla.
  - → Robustez: Cuán bien responde el sistema a condiciones de operación inesperada.
  - → Safety: Cuán bien protege el sistema contra lesiones o daños físicos.
  - → Seguridad: Cuán bien protege el sistema contra el acceso no autorizado a la aplicación y a sus datos.
  - → Usabilidad: Cuán fácil es para las personas aprender, recordar y usar el sistema.

Internos:

- No son observables durante la ejecución del sistema.
- Describen propiedades que se perciben durante el desarrollo, verificación o mantenimiento.
- Afectan indirectamente a clientes y usuarios.
  - → Eficiencia: Cuán eficientemente el sistema utiliza los recursos informáticos.
  - → Modificabilidad: Cuán fácil de mantener, cambiar, mejorar y reestructurar el sistema.
  - → Portabilidad: Con qué facilidad se puede hacer que el sistema funcione en otros entornos operativos.

→ Reusabilidad: Hasta qué punto los componentes del sistema se pueden utilizar en otros sistemas.
→ Escalabilidad: Con qué facilidad el sistema puede crecer para manejar más usuarios, transacciones, servidores u otras extensiones.
→ Verificabilidad: Con qué facilidad los desarrolladores y testers pueden evaluar y confirmar que el software se implementó correctamente.

Tipos de proyectos:

Sistemas Embebidos.

Aplicaciones Web o Corporativas.

Aplicaciones de Escritorio o Móviles.

Exploración y captura:

Ciertos atributos se contraponen lo que hace imposible que se puedan dar todos en simultáneo, por lo que hay que buscar un balance y determinar cuáles son más importantes para el éxito del proyecto.

Captura: Lineamientos.

1. Comenzar con un listado amplio de atributos de calidad.

2. Reducir la lista seleccionando los atributos importantes.

3. Priorizar cada atributo de calidad.

4. Capturar expectativas específicas para cada atributo.

5. Las especificaciones de requerimientos correspondientes a atributos de calidad deben ser específicos, medibles, alcanzables, relevantes, sensibles al tiempo. (SMART)

Trade-offs: Idealmente, todo sistema debería exhibir el mayor valor posible, pero algunos atributos de calidad se contraponen. Para alcanzar el balance óptimo en las características del producto es necesario identificar, especificar y priorizar los atributos de calidad.

Calidad de los requerimientos

Características individuales: Aplican a cada requerimiento en forma individual, sin considerar los requerimientos restantes.

- Completo: Un requerimiento es completo si tiene toda la información necesaria para que un lector pueda comprenderlo.
- Correcto: Un requerimiento es correcto si describe exactamente una característica o funcionalidad del sistema que permite satisfacer las necesidades de alguna parte interesada.
- Factible: Un requerimiento es factible si puede implementarse en el entorno operativo del sistema y con los recursos establecidos en el proyecto.
- Necesario: Un requerimiento es necesario si corresponde a una característica, tarea, funcionalidad o atributo valioso para un participante.
- Priorizado: Un requerimiento está priorizado si se ha establecido un valor que indica cuán esencial es para el producto.
- No ambiguo: Un requerimiento es no ambiguo si solo tiene una interpretación posible o todas las interpretaciones posibles son equivalentes para clientes o usuarios.
- Verificable: Un requerimiento es verificable si es posible establecer un criterio objetivo para decidir si su implementación es correcta.

Características Globales: Aplican al conjunto completo de requerimientos.

- Completo: Un conjunto de requerimientos es completo si abarca todas las necesidades de las partes interesadas y todas las características del producto definidas en el alcance de la iteración actual.
- Consistente: Un conjunto de requerimientos es consistente si no hay conflicto entre los requerimientos del mismo nivel o de diferentes niveles.
- Modificable: Un conjunto de requerimientos es modificable si es posible reconocer e identificar con precisión cada requerimiento del conjunto.
- Trazable: Un conjunto de requerimientos es trazable si cada requerimiento puede vincularse hacia atrás hasta su origen y hacia adelante con los requerimientos derivados de él, su diseño, implementación y mecanismos de testing.

Recomendación para la Especificación de Requerimientos:

Adoptar y mantener un estilo de escritura uniforme.

Los requerimientos deben ser especificados con un nivel e detalle que provea a los desarrolladores y testers la información que necesitan.

Evitar ambigüedades.

Evitar incompletitud.

Priorización de Requerimientos:

La prioridad puede pensarse como u atributo que establece una relación de orden entre los requerimientos de acuerdo a su importancia o urgencia.

Es especialmente importante cuando:

- Los recursos son escasos.
- Los requerimientos son complejos.
- Existen múltiples criterios, e intereses diversos o contrapuestos.

Técnicas:

- In/Out: Se solicita a las partes interesadas que tomen una lista de requerimientos y tomen una decisión binaria.
- Comparación de a pares y asignación de ranking: Se construye una tabla de doble entrada y se comparan los requerimientos.
- Escala de 3 niveles: Se definen 3 niveles a ser considerados por las partes interesadas: alto, medio y bajo.
- Doble escala: Importancia y Urgencia: Se determina con una tabla de doble entrada que tan importante y urgente son los requerimientos.
- MoSCoW: Must, Should, Could, Won't.
- $100: Se distribuyen $100 pesos imaginarios a cada miembro del equipo de priorización y se "invierte" su dinero en los requerimientos que se consideran más importantes para la iteración actual.

Validación de Requerimientos:

Verificación: Evaluar si el producto satisface los requerimientos.

Validación: Evaluar si el producto satisface las necesidades de los usuarios y clientes.

Técnicas:

- Prototipos: Son una herramienta de validación que ayuda a hacer más tangible los requerimientos.
- Test Conceptual: Pueden construirse a partir de requerimientos de usuarios, requerimientos funcionales o prototipos. Ayudan a visualizar el comportamiento esperado del sistema.
- Revisión Informal: Útiles para recolectar feedback desestructurado. Ayuda a identificar errores grandes.
- Revisión Formal: Tiene una estructura más rígida y sigue un procedimiento sistemático. Se genera un reporte que incluye una descripción del material examinado. Todos los miembros del equipo de revisión formal son responsables de la calidad de la revisión.

Revisión Formal: Inspección.

La inspección es una de las técnicas de validación de requerimientos que más beneficios brinda, favoreciendo el desarrollo de software de calidad. Puede aplicarse a requerimientos, documentos de diseño, código fuente y planes de proyectos. Tiene un costo alto en recursos y tiempo.

Entre los participantes se tiene:

- Autor: Toma un rol más pasivo, escuchando comentarios y respondiendo preguntas.
- Moderador: Coordina las actividades, distribuye material y efectúa un seguimiento de los cambios que deban realizarse sobre los requerimientos.
- Lector: Parafrasea los requerimientos y modelos bajo inspección.

- Escriba: Utiliza documentos estándar para registrar los problemas identificados durante cada encuentro, documentando claramente todo.

Prerrequisitos: Antes de organizar una inspección de requerimientos el moderador debe chequear que:

- El documento sigue la plantilla adoptada por el equipo y no tiene errores de ortografía, gramática o formato.
- Están incluidos los números de línea y otros identificadores necesarios para facilitar referencias.
- Todos los aspectos no resueltos están marcados.
- El moderados no encontró más de 3 errores graves en una breve lectura.

Proceso: (Ver cuadro en clase de validación de requerimientos)

- Planificación: Se determinan participantes y roles, documentos a inspeccionar, fecha de la inspección, tiempo necesario.
- Preparación: El autor comparte con los inspectores información para brindarles contexto, y estos buscan los errores más grandes. En esta etapa se detectan durante esta etapa.
- Checklist de defectos: Preguntas que abordan algunos aspectos de calidad.
- Reunión: El lector lee cada requerimiento del documento, el escriba registra lo discutido, y luego de examinar la documentación se ve si se acepta el documento o si se modifica.
- Rehacer: Prácticamente todo el proceso de inspección va a revelar anomalías. El analista debe estar preparado para rehacer parte de los requerimientos luego de un encuentro de inspección.
- Seguimiento: Esta etapa corresponde al cierre del proceso y permite que el moderador determine si se cumplió el criterio de terminación.
- Criterios de terminación: El moderador verifica ciertos aspectos antes de dar por finalizado el proceso de inspección.

Tests de Aceptación a partir de Criterios de Aceptación:

Los criterios de aceptación permiten validar requerimientos individuales y otros aspectos generales de la solución propuesta. Definen condiciones mínimas para que el producto se considere listo para salir al mercado. Se pueden definir criterios de aceptación para los tests:

- El producto ofrece todas las funcionalidades de alta prioridad.
- El producto satisface las características no funcionales y las métricas de calidad esenciales.
- No quedan errores o ausencias por resolver o modificar, vinculados a los requerimientos prioritarios.
- El producto cumple todas las condiciones que imponen las leyes y regulaciones.

Mientras que los criterios de aceptación se enfocan en determinar qué características debe tener el producto para ser considerado aceptable, los tests de aceptación se centran en definir cómo serán satisfechas.

Los tests sobre aspectos funcionales deberían enfocarse en los escenarios más comunes (flujo normal de los casos de uso y sus excepciones correspondientes), quitando relevancia a flujos alternativos menos frecuentes. Para aspectos no funcionales, los tests deben asegurar que las metas de performance son satisfechas, que el sistema verifica los estándares de usabilidad establecidos, y que las medidas de seguridad adoptadas son adecuadas.

Reuso

Reusar implica aprovechar el trabajo realizado previamente, en el mismo proyecto o en otros. Brinda beneficios que permiten avanzar hacia la solución más rápidamente. Puede provocar algunos riesgos.

Componentes reutilizables:

Subprogramas, librerías, requerimientos, diseño, arquitectura, código, tests de aceptación.

Beneficios:

- Mejoras en la productividad del equipo.
- Aceleración de los ciclos de revisión, prueba y aprobación.
- Reducción de costos y errores.
- Mayor consistencia.

Dimensiones de Reuso:

Consideramos las siguientes dimensiones para el reuso de requerimientos:

- Grado de reuso: Cantidad y tipo de material a ser reutilizado. Alternativas:
  - Requerimiento individual.
  - Requerimiento y sus atributos.
  - Requerimiento, sus atributos e información asociada.
  - Conjunto de requerimientos relacionados.
  - Conjunto de requerimientos relacionados y piezas de diseño.
  - Conjunto de requerimientos relacionados, piezas de diseño, implementación y tests.
- Grado de modificación: Cuánto debe modificarse el componente para poder reutilizar los requerimientos. Alternativas:
  - Sin cambios.
  - Modificaciones en los atributos o los valores de los atributos.
  - Modificaciones en la especificación.
  - Modificaciones en la información o componentes asociados.
- Mecanismo de reuso: Forma en la que se implementa el reuso de requerimientos. Alternativas:
  - Copiar y pegar desde otra especificación del mismo proyecto.
  - Copiar y pegar de un repositorio de requerimientos reusables.
  - Utilizar referencias: Única copia local o referenciada dentro de un mismo proyecto, o única copia global referenciada desde otros proyectos.
    - Referencia directa: Cuando se modifica el requerimiento original, el cambio impacta en todos los documentos que lo referencian directamente.
    - Historial de versiones: Cuando se modifica un requerimiento se crea una versión del mismo, vinculada con la versión previa. Las versiones previas reusadas no se afectan.

Escenarios y oportunidades de reuso:

- Dentro de un mismo producto.
- Dentro de una línea de productos relacionados.

- Dentro de una organización.
- Dentro de dominio de aplicación.
- Al desarrollar sistemas de mejora o reemplazo.

Factores de éxito:

- Uso de repositorios o herramientas de administración de requerimientos.
- Mejorar los requerimientos reutilizados cuando sea posible.
- Mantener accesibles y actualizados los links.
- Uso de glosarios.
- Fomentar cultura de reuso: Especificación de requerimientos de calidad con potencial de reuso.
- Uso de patrones o plantillas.

Barreras u obstáculos:

- Los requerimientos no están documentados, o su documentación no está disponible o está incompleta.
- Diferencias en el estilo de escritura, convenciones adoptadas, vocabulario.
- Diferencias en la categorización y organización de requerimientos.
- Los requerimientos son propiedad del cliente que encomendó el producto inicial y no pueden ser reusados.
- Requerimientos muy específicos.
- NAH – “Not aplicable here”.
- NIH – “Not invented here”.

Administración de Requerimientos

La administración de requerimientos incluye todas las actividades destinadas a mantener la integridad, consistencia, precisión y vigencia de los requerimientos a través del proyecto:

- Seguimiento de estado de los requerimientos.
- Rastreo de requerimientos.
- Gestión de cambios en los requerimientos.
- Control de versiones de los requerimientos.

El analista lidera la administración de requerimientos, pero no es el único responsable.

Acuerdos y convenciones:

Definición de las actividades que el equipo debe llevar a cabo como parte del proceso de administración de requerimientos y establecimiento de:

- Atributos y estados de los requerimientos.
- Responsable de actualizar la información de rastreo de los requerimientos.
- Mecanismo para manejar la propuesta, evaluación, negociación y comunicación de nuevos requerimientos o cambios en los ya existentes.
- Mecanismo de evaluación de impacto.
- Mecanismo para la aprobación de nuevos requerimientos.
- Mecanismo de actualización del plan del proyecto.
- Uso de herramientas de administración de requerimientos.

Atributos de requerimientos:

Cada requerimiento puede pensarse como un objeto con propiedades únicas. Además, cada uno tiene un conjunto de atributos que determinan su contexto, por ejemplo:

- Id.
- Fecha de creación.
- Fecha de última modificación.
- Autor.
- Responsable de la última versión.
- Número de versión actual.
- Prioridad.
- Estado: Permite realizar un seguimiento de estado, así determinar el grado de avance del proyecto.
- Fuente.
- Justificación o racionalidad: Justifica el por qué el requerimiento es relevante para el producto. Un nuevo requerimiento puede provocar que la justificación de otro requerimiento previo deje de valer, y por lo tanto pueda eliminarse.

Utilizar una cantidad excesiva de atributos puede resultar inconveniente para el equipo de trabajo, pues posiblemente queden valores sin definir.

Seguimiento de estados:

Realizar un seguimiento del estado de los requerimientos requiere:

- Definir los posibles estados.
- Actualizar el estado de cada requerimiento.
- Analizar periódicamente la distribución de estados de los requerimientos.

También requiere documentar las dependencias y enlaces lógicos entre los requerimientos, así como también entre los requerimientos y otros elementos del sistema. Para que los requerimientos sean rastreables deben tener una etiqueta persistente que los identifique unívocamente.

La información de rastreo permite enlazar un requerimiento hacia atrás y hacia adelante, desde su origen hasta su implementación y viceversa.

Componentes hacia las que se puede rastrear: reglas del negocio, otros requerimientos, arquitectura y otros componentes de diseño, código fuente, componentes de testing, etc.

Beneficios del rastreo de requerimientos:

El rastreo de requerimientos incrementa la calidad del producto, reduce costos de mantenimiento y facilita el reuso. Entre otras cosas permite:

- Encontrar requerimientos ausentes.
- Encontrar requerimientos innecesarios.
- Certificar el cumplimiento de reglamentaciones o estándares.
- Evaluar el impacto de los cambios.
- Mantenimiento, reingeniería o reemplazo de componentes de código.
- Reusar componentes.
- Verificar componentes de código.

Links de rastreo – Multiplicidad:

Los links de rastreo puede definir distintos tipos de relaciones entre los componentes del sistema.

- Uno a uno.
- Uno a muchos.
- Muchos a muchos.

Matriz de trazabilidad:

La forma más común de representar los links entre los requerimientos y otros componentes del sistema es la matriz de trazabilidad o tabla de trazabilidad.

Gestión de cambios:

En un mundo ideal, todos los requerimientos están especificados y documentados y se mantienen estables durante el proceso de desarrollo.

En el mundo real durante el desarrollo se generan oportunidades de mercado, cambios en las reglamentaciones o nuevas necesidades, se detectan omisiones o inconsistencias.

Problema → El alcance inalcanzable:

- Si los requerimientos cambian y los tiempos se modifican, el proceso de desarrollo se alarga.
- Si se aceptan cambios sin ajustar la planificación, es muy probable que no pueda cumplirse.

En ambos casos la entrega final puede demorarse.

Aspectos a evitar:

- El cliente propone cambios y modificaciones sin considerar requerimientos.
- El usuario propone cambios y se atienden sin considerar su alineación con los requerimientos del negocio por el cliente.
- El desarrollador recibe propuestas de cambio y los atiende sin la participación del analista.
- Los cambios se aceptan sin evaluar impactos.
- Los cambios se aceptan y no se modifica el plan del proyecto.
- Los cambios se aceptan y no se modifica la documentación de los requerimientos.

El costo de los cambios:

Aceptar o rechazar un cambio cuando el producto está en desarrollo tiene un costo. La gestión del cambio en sí misma, ya sea cuando se acepta o se rechaza un cambio, también demanda tiempo y esfuerzo.

Actividades:

- Propuesta de cambios.
- Análisis de impacto de los cambios.
- Tomar decisiones sobre los cambios propuestos.
- Actualizar requerimientos individuales y/o conjuntos de requerimientos.
- Actualizar el plan del proyecto.
- Medir la volatilidad de los requerimientos.

Política de gestión de cambios:

Los administradores del proyecto deben definir y comunicar una política de gestión de cambios que establece las expectativas de cómo el equipo de trabajo debe manejar las solicitudes de cambio en los requerimientos y otros componentes. Adoptar una política de gestión de cambios involucra:

- Designar un Grupo de Control de Cambios (GCC).
- Definir un procedimiento y reglas para la gestión de cambios.
- Seleccionar herramientas para la gestión de cambios del proyecto.
- Mantener al administrador o líder del proyecto informado de las decisiones.

Declaraciones:

- Todos los cambios deben seguir el proceso establecido. Si una solicitud de cambio no es registrada de acuerdo al proceso, no será considerada.
- Solicitar un cambio no garantiza que será llevado a cabo. El Grupo de Control de Cambios decidirá qué cambios serán implementados.
- El contenido de la base de datos de cambios debe ser visible para todas las partes interesadas del proyecto.

- Cada solicitud de cambio deberá ser sometida a un análisis o evaluación de impacto.
- Cada cambio incorporado debe poder rastrearse a una solicitud de cambio aprobada.
- Debe registrarse la justificación detrás de la aprobación o rechazo de cada solicitud de cambio.

Grupo de control de cambios:

El Grupo de Control de Cambios (GCC) es el encargado de decidir qué cambios sobre los requerimientos serán aceptados, cuáles serán aceptados con revisiones, y cuáles serán rechazados. También decide qué defectos reportados deberán ser corregidos, y cuándo deberán corregirse. Debe establecer un mecanismo para la toma de decisiones, los roles del grupo, la cantidad de miembros requeridos para sesionar. Debería conformarse antes de que surjan los cambios y, de ser necesario, incorporar miembros para atender modificaciones específicas. Al menos uno de los miembros del grupo debe tener conocimiento profundo del negocio y la organización.

Solicitudes de cambio:

Las solicitudes de cambio en los requerimientos u otros componentes del sistema deben contener la información necesaria para poder evaluar su aprobación o rechazo, y permitir trazarla a los demás elementos involucrados.

El GCC puede definir un conjunto de atributos para cada solicitud de cambio, cuyos valores serán determinados en distintas etapas del proceso.

Tareas del proceso de gestión de cambios:

- Evaluar una solicitud de cambio.
- Tomar una decisión sobre el cambio.
- Implementar un cambio.
- Verificar un cambio.

Estados de una solicitud de cambio:

Durante el proceso de gestión de cambios, una solicitud de cambio en los requerimientos atraviesa distintos estados.

Análisis de impacto:

El análisis de impacto permite comprender las implicancias de cada cambio propuesta. El resultado de este análisis permite renegociar los acuerdos con los clientes. Los pasos son:

- Entender las implicancias de aceptar el cambio propuesto.
- Identificar los componentes afectados por el cambio propuesto.
- Estimar el esfuerzo requerido para implementar el cambio.

Como parte del proceso del análisis de impacto del cambio solicitado el equipo evaluador puede utilizar checklists que le permiten:

- Entender las implicancias de aceptar el cambio propuesto.
- Identificar los componentes afectados por el cambio propuesto.
- Estimar el esfuerzo requerido para implementar el cambio.

Resultado de la evaluación:

A partir del análisis de impacto para una solicitud el GCC puede:

- Rechazar el cambio.
- No aceptar el cambio en la iteración actual, pero analizar de agregarlo en otra iteración.
- Aceptar el cambio.

Evaluación de la Actividad de Cambio:

Medir la actividad de los cambios permite evaluar la estabilidad de los requerimientos, revelando oportunidades para mejorar el proceso de gestión de cambios de manera de reducir los cambios a futuro.

Control de versiones:

El control de versiones está ligado a la gestión de cambios. Permite identificar unívocamente las distintas versiones de los requerimientos, tanto en forma individual como de conjuntos de requerimientos. Los cambios deben quedar documentados y ser comunicados a todos los afectados, asegurándose de que el identificador de la versión del requerimiento se actualice con cada cambio realizado. Sus objetivos son:

- Retener la historia de los cambios.
- Hacer un seguimiento histórico de los cambios a través de las versiones de requerimientos.
- Permitir deshacer cambios para restablecer un ítem que ha sido modificado.

Uso de herramientas:

Existen herramientas de software que integran funcionalidades para cada una de las tareas del desarrollo y administración de requerimientos.

El uso de herramientas de propósito general para la administración de requerimientos conlleva diversos desafíos:

- Mantener los documentos actualizados y sincronizados.
- Comunicar los cambios a los participantes y establecer vínculos entre los requerimientos, y entre los requerimientos y otros componentes.
- Rastrear el estado de los requerimientos individuales y de los conjuntos de requerimientos.

Al decidir utilizar una herramienta de propósito general para la administración de requerimientos el equipo de trabajo debe establecer convenciones:

- Mecanismo de sincronización o servicio de almacenamiento compartido.
- Pautas para la generación de versiones.
- Procedimiento para registrar y controlar cambios posteriores al acuerdo base.
- Mecanismo de rastreo y control de versiones.

Metodologías ágiles

Existe una profunda relación entre los modelos de ciclo de vida y el desarrollo y administración de requerimientos. Son adecuados para proyectos en los que los requerimientos son relativamente estables.

Modelos predictivos: Conducidos por planes, intentan anticipar los cambios para minimizar los riesgos de proyecto.

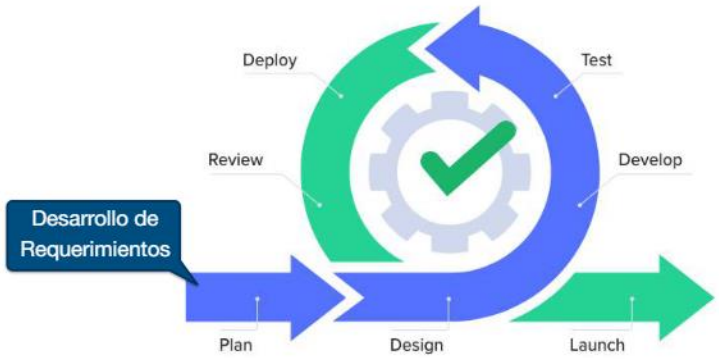
- Modelo en cascada: Las etapas suceden en forma secuencial.
- Modelo iterativo incremental: Permite superposición y retroalimentación en las etapas de desarrollo brindando cierto soporte para adaptarse a los cambios en los requerimientos.

Modelos adaptativos: Conducidos por los cambios, intentan reducir el impacto de los cambios. Son especialmente adecuados en ambientes en ambiente dinámicos en los que los requerimientos son volátiles, o cuando el sistema se desarrolla a partir de una idea de producto para el cual los requerimientos son inciertos.

Metodologías ágiles:

Las metodologías ágiles se basan en modelos iterativos incrementales que evolucionan hacia un modelo adaptativo. Se caracterizan por:

- Iteraciones breves, dando lugar a entregas rápidas.
- Pequeños incrementos, cada uno de los cuales incluye unas pocas características del producto prioritarias para un grupo de partes interesadas.



Clientes y usuarios:

Tienen un rol protagónico en todo el proceso, justamente porque cada iteración es muy breve. Se deben seleccionar usuarios representativos y con un entrenamiento mayor que en otros modelos.

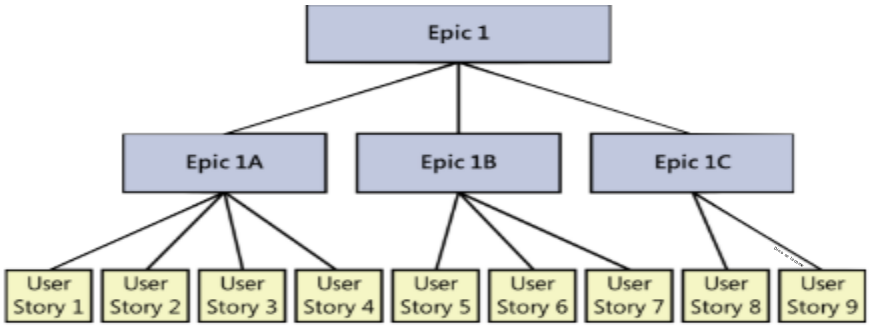
Historias de usuarios:

Los requerimientos se especifican a través de historias de usuario. Esta es una narración breve, realizada normalmente por un usuario que describe una característica del sistema que considera valiosa. Deben ser:

- Breves y simples.
- Independientes.
- Negociables.
- Valorables.
- Estimables.
- Testeables.

Historias épicas:

Es una historia de usuario muy grande para ser implementada en una sola iteración, por lo que se la divide en otras historias más pequeñas que puedan desarrollarse en una iteración.



Prioridades:

A partir del conjunto de historias de usuario se establecen prioridades. Aún las historias con menor nivel de prioridad deben estar alineadas con los objetivos del negocio y resultar valiosas para los usuarios.

En cada iteración se selecciona un conjunto de historias de usuario que poseen la mayor prioridad, y las demás pasan a conformar el **backlog** o **lista de pendientes**.

En ocasiones la prioridad entre las historias de usuario está determinada por dependencias de orden entre ellas.

Backlog o Lista de pendientes:

El backlog está compuesto fundamentalmente de historias de usuarios. Cuando termina una iteración el equipo retoma la lista de pendientes y la actualiza. Cada proyecto debería tener un único backlog.

Alcance del proyecto:

El alcance del proyecto queda definido por el conjunto de historias de usuario seleccionadas del backlog para ser abordadas durante la iteración actual. En cada iteración probablemente se identifiquen nuevos requerimientos que se seleccionarán para esa iteración o se incorporarán a la lista de pendientes para futuras iteraciones.

Documentación:

Se puede documentar usando herramientas específicas para metodologías ágiles o herramientas generales.

El principal documento de especificación de requerimientos es el backlog.

Responsabilidades:

De los **clientes** y **usuarios**:

- Establecer prioridades entre los requerimientos.
- Definir criterios de aceptación que se aplicarán en la etapa de verificación.

De los miembros del **equipo de desarrollo**:

- Seleccionar usuarios representativos.
- Formular historias de usuario.
- Asignar prioridades en función de los objetivos del negocio.
- Definir tests de aceptación.
- Moderar negociaciones.
- Evaluar el impacto de los cambios.

Del **product owner**:

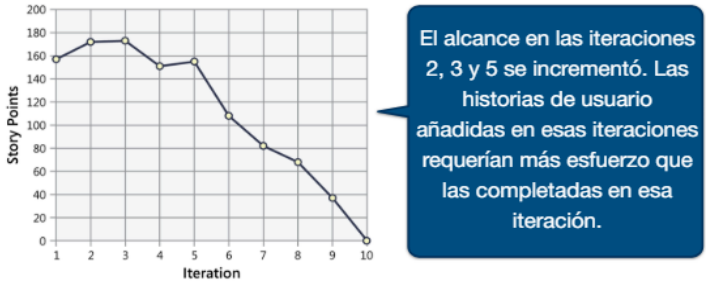
- Aceptar o rechazar el alcance del proyecto para cada iteración.
- Establecer la prioridad cuando hay conflictos.
- Aceptar o rechazar la entrega en cada iteración.

Esfuerzo:

El esfuerzo que demandará el desarrollo de cada historia de usuario puede medirse utilizando una medida denominada **story point**. La selección de historias de usuario a ser abordadas en una iteración suele hacerse considerando su prioridad y su tamaño en story points.

Progreso del proyecto:

El progreso del proyecto puede medirse efectuando un análisis sobre el backlog, a partir de un gráfico que ilustra los story points pendientes en cada iteración.



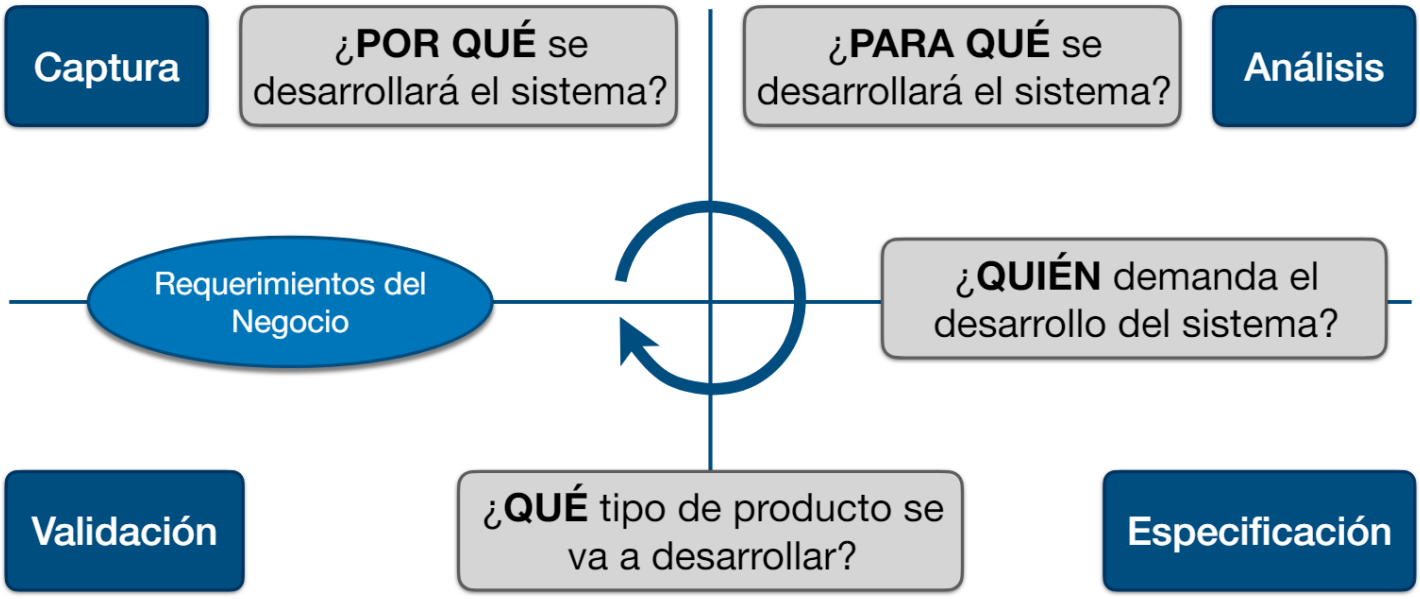
Gestión de cambios:

Existen equipos de trabajo que destinan algunos días de cada iteración al desarrollo de requerimientos y, una vez establecidos, la iteración se completa sin cambios, siguiendo etapas como las del modelo en cascada. En tales escenarios, cualquier propuesta de cambio se tratará en la siguiente iteración. Esta estrategia brinda cierta estabilidad y previsibilidad sobre el entregable de la iteración, manteniendo las expectativas de los clientes y usuarios.

Otros equipos admiten que durante una iteración se incorporen nuevas historias de usuario que provocan cambios en los requerimientos o en las prioridades. La ventaja en estos casos es que el equipo tiene mayor flexibilidad para adaptarse rápidamente a nuevas necesidades prioritarias de clientes o usuarios.

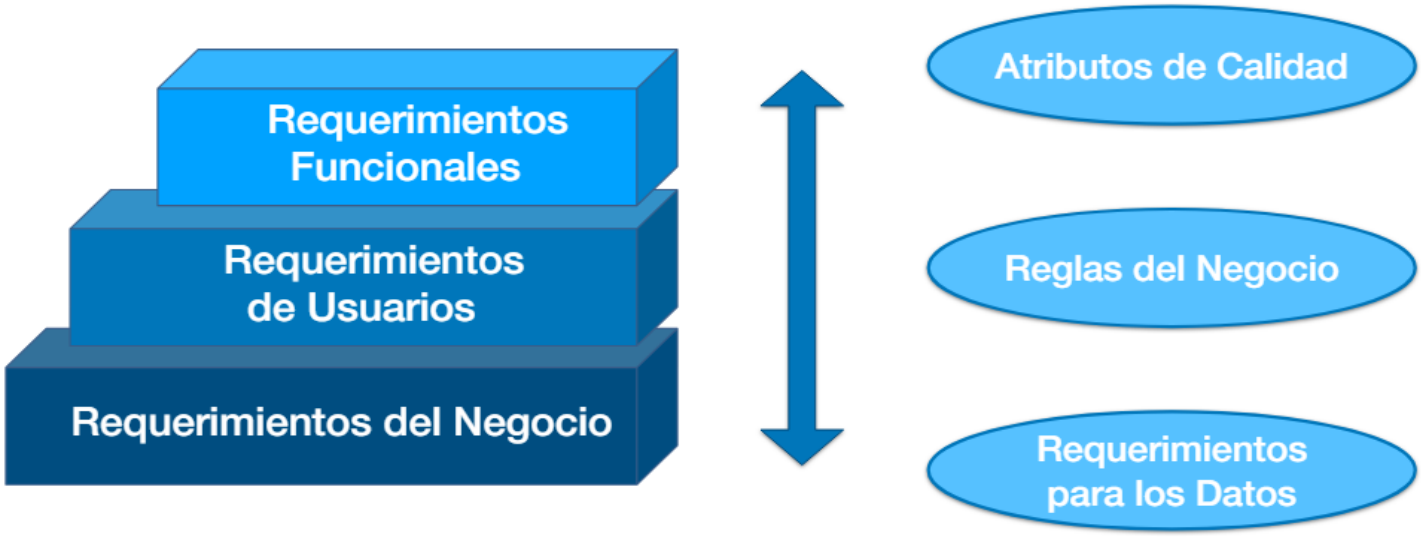
Encontrar un balance entre admitir cambios excesivos y la rigidez excesiva es esencial. La corta duración de las iteraciones y el tamaño limitado de los incrementos del producto como resultado de cada iteración hacen que la gestión de cambios pueda efectuarse de manera frecuente pero de manera limitada.

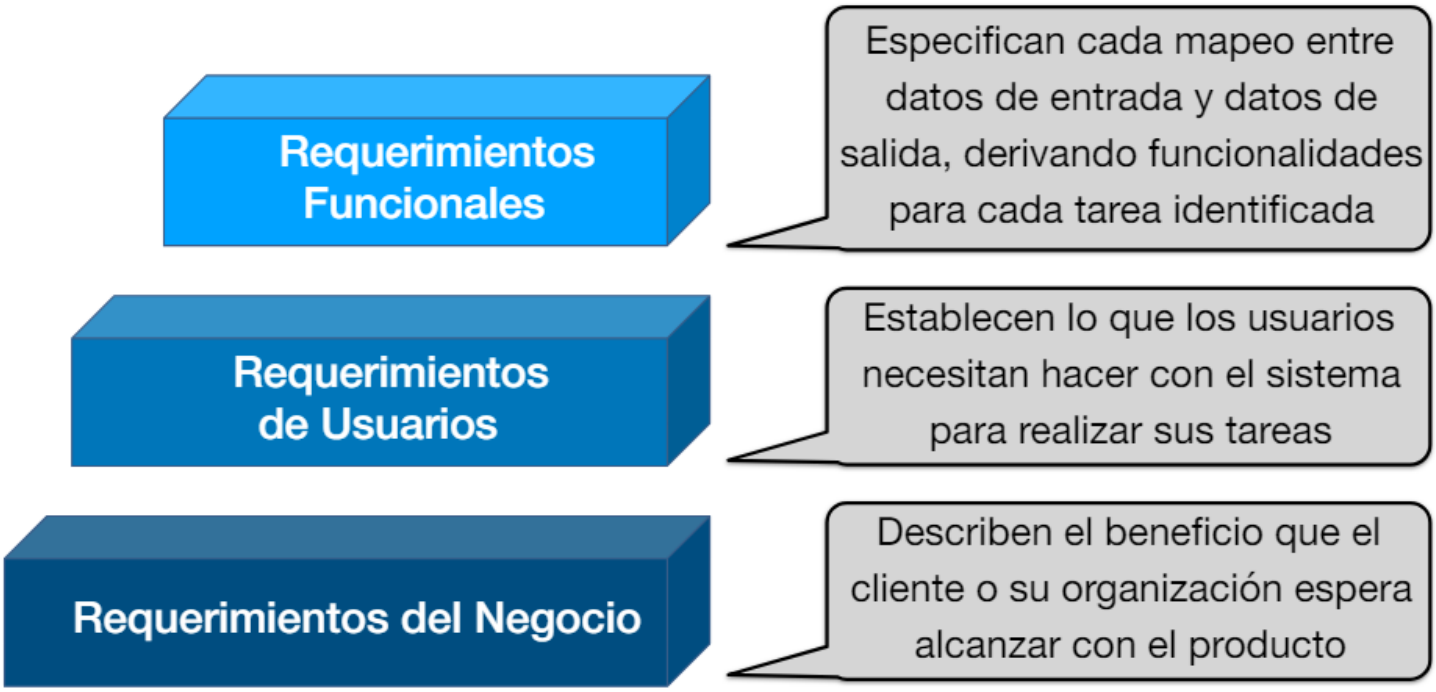
Primera parte de la materia



Visión del producto: Brinda el contexto para la toma de decisiones durante el desarrollo del producto y alinea a todos los participantes. Apunta a todo el proyecto.

Alcance del proyecto: Identifica qué porción del producto final será abordada por el proyecto en cada momento o iteración particular. Alcanza a cada iteración.





Limitaciones: Describen características o cualidades que no están contempladas por del alcance del proyecto.

Riesgos: Describen elementos que pueden provocar que no se alcancen los objetivos del negocio y, por lo tanto, atentan contra el éxito del proyecto.

Premisas: Definen condiciones que se dan por supuestas en el marco del desarrollo del proyecto. Consideramos aspectos que no son abordables con el desarrollo del producto.

Requerimientos del negocio:

- Documento de especificación de requerimientos.
- Documento de visión y alcance.

Requerimientos de usuarios: Tienen el enfoque basado en el usuario.

- Diagrama de casos de uso: Modela el conjunto de interacciones entre el sistema y uno o más actores.
- Historia de usuario.
- Prototipos.

Caso de uso: Tarea.

Reglas de negocio: Taxonomía.

- Hechos.
- Restricciones.
- Disparador de acciones.
- Inferencias.
- Reglas de cómputo.

Requerimientos funcionales:

- Diagrama de carriles: Modela el flujo u orden entre las tareas.
- Diagrama de actividad: Representa visualmente el flujo lógico de un proceso.
- Diagrama de estados: Modela todos los estados por los que puede pasar una entidad.