Requisito: necesidad documentada sobre el contenido, forma o funcionalidad de un producto o servicio. Datos de entrada para diseñar qué hace el sistema, no cómo. Pueden ser esenciales o no

* Requisito funcional: establecen los comportamientos. Funciones (entradas, comportamientos y salidas) que el sistema debe realizar
* Requisito no funcional: características de funcionamiento de las funciones
* Limitaciones externas: leyes, sistemas que interactúan
* Pseudorrequisitos: entorno donde será implementado el sistema

Características de los requisitos

* No ambiguo, texto claro, única interpretación
* Conciso: lenguaje natural (deben entender los interesados), aspectos importantes
* Consistente: mismo lenguaje entre requisitos
* Completo: deben contener toda la información, no remitir a fuentes externas
* Alcanzable: objetivo realista, posible de alcanzar con los recursos disponibles
* Verificable: verificar si fue satisfecho o no (inspección, análisis, testeo)

Metodología: pasos sistemáticos, predecibles y repetibles que mejoran la productividad y calidad del sw. El conjunto de pasos se llama ciclo de vida

* Obtención de requisitos: motivo de creación del programa, contexto del negocio, identificar recursos, usuarios y dispositivos finales, almacenamiento de datos, puntos críticos, respuesta del sw ante situaciones inesperadas
* Análisis de requisitos: ver que no estén incompletos, ambiguos, contradictorios. Documento ERS (especificación de requerimientos) y MER
* Limitaciones: el sw tiene un conjunto de soluciones limitado. El proceso mecánico puede requerir más rendimiento, lo que requiere mejores máquinas
* Especificación: comportamiento esperado, necesidades de negocio e interacción con usuarios para clasificar, identificar, priorizar y especificar requisitos
* Arquitectura: integrar infraestructura, bases de datos y herramientas gerenciales. Tener en cuenta aspectos lógicos y físicos, modelos de organización y representación de datos, procesos del sistema. Patrones de arquitectura
* Programación: implementar en código. Complejidad y duración relacionada al lenguaje usado
* Desarrollo de la app
  + Desarrollo y organización de infraestructura
  + Adaptación del paquete: entender detalladamente el funcionamiento para maximizar el rendimiento y verificar que no hay errores
  + Desarrollo de unidades de diseño interactivas: procedimientos de diálogos usuario-sistema. Establecen las acciones que debe hacer la unidad de diseño. Crear componentes para procedimientos. Pruebas unitarias y de integración
  + Desarrollo de unidades de diseño batch: definir clara y objetivamente las especificaciones para que el programador entienda al programar y testear
  + Desarrollo de unidades de diseño manuales: proyectar los procedimientos administrativos que habrá en torno al uso de los componentes
* Pruebas de sw: verificar que realiza correctamente las tareas. Pruebas unitarias (alguien distinto al programador en un área de pruebas. Para el área de pruebas es un grupo sin experiencia para probar que la documentación es clara), pruebas de integración (grupo experimentado que sabe en qué condiciones puede fallar),
* Implementación: convertir una especificación en un sistema ejecutable
* Documentación: documentar desarrollo y gestión de proyecto con diagramas, casos de uso, pruebas, manuales de usuario y técnicos. Sirve para correcciones, usabilidad, mantenimiento y ampliaciones
* Mantenimiento: mantener (corregir errores nuevos) y mejorar (incorporar nuevos requisitos). 2/3 del tiempo del ciclo de vida es acá. 80% del mantenimiento son mejoras de funcionalidad

Casos de uso: descripción de acción o actividad que hace un actor (humano con rol o sistema. Puede ser el tiempo, ejecutar algo periódicamente) para llevar a cabo algún proceso. Un sistema son interacciones entre casos de uso y actores en respuesta a un evento iniciado por un actor principal. Ilustran los requisitos funcionales. Se hace un modelo (diagramas de cada caso de uso) con glosario

* <<communicates>>: relación entre actor y caso de uso, la participación del actor
* <<uses>>, <<includes>>: dependencia entre dos casos, inclusión del comportamiento de un escenario en otro. Para evitar repetición del comportamiento en casos distintos
* Generalización: un caso varía cualquier atributo de un caso base
* <<extend>>, <<extiende>>: generalización más controlada, caso base especifica puntos de extensión que son modificables (y solo esos). Comportamientos similares pero uno hace más que el otro

Características de los casos de uso

* Lenguaje natural
* Cada caso de uso se centra en una única tarea. Cada uno describe una característica del sistema, para definir el sistema se precisan varios
* Son funcionalidades externas, lo que hace el actor para usar la funcionalidad
* Sencillo, un desarrollador lo debe elaborar en un único lanzamiento
* Centrados en el cliente (para los requisitos)
* Priorizar según el valor aportado al negocio
* No establecen completamente requisitos funcionales y no funcionales
* Se deben complementar con información adicional (reglas de negocio, requisitos no funcionales, diccionarios de datos)
* Los casos críticos de uso deben tener un requisito no funcional asociado

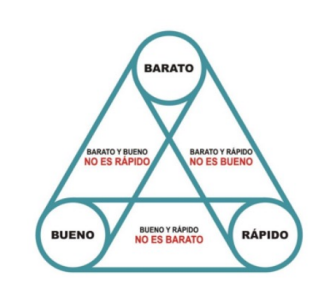
Historias de usuario: explicación general e informal de una funcionalidad desde el punto de vista del usuario final. Sirven para ver cómo aporta valor al usuario

* Mínima unidad de trabajo en entornos ágiles
* Centrarse en el usuario
* Escritas por product owner o producto manager
* Desarrollo de historias en base a prioridades
* As a persona, I want to, so that
  + Persona: para quién se está construyendo
  + Want to: intención, no funcionalidad usada
  + So that: objetivo, por qué quiere hacer esa acción
* As Max, I want to organize my work, so I can feel more in control
* As a manager, I want to understand my colleagues, so I can report successes and failures

Requerimientos SMARTT:

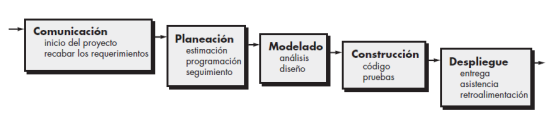
* Specific: concreto y bien definido. Si es algo grande, dividirlo en partes pequeñas
* Measurable: criterios para medir y evaluar progreso. Indicadores cuantificables
* Achievable: alcanzable, realista. Tener en cuenta presupuesto, personal, recursos
* Relevant: alineado con otras metas y tiene sentido en el contexto del negocio
* Time-bounded: estableces objetivos con límite de tiempo
* Traceable: dónde surge y registro de lo que pasó en el tiempo

Triángulo de hierro



Modelo cascada

* No se vuelve a pasos anteriores, lo que implica que no hay iteraciones y que todo tiene que estar perfecto antes de pasar de fase



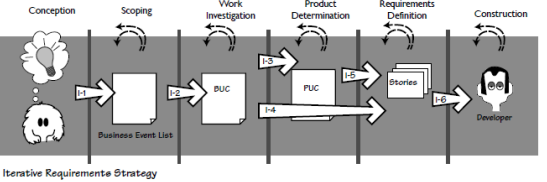
Estrategia externa

Diagrama, Esquemático

Descripción generada automáticamente

* Concepción 🡪 alcance: idea, solicitud de cambio, nueva restricción, etc. Estimaciones preliminares de alcance y costos ROM
* Alcance 🡪 investigar el trabajo: límite entre trabajo y universo, entradas, datos producidos, interesados y nivel de involucramiento. Se definen los BE (algo que ocurre que dispara una respuesta planificada llamada BUC)
* Investigar el trabajo 🡪 determinación del producto: identificar BUCs (casos de uso del negocio, lo que hace. Para cada BE hay un BUC)
* Determinación del producto 🡪 definición de requerimientos: definir PUCs (qué parte de los BUC voy a automatizar, uno o más PUC por BUC)
* Definición de requerimientos 🡪 construcción: requisitos bien definidos (funcionales, no funcionales, restricciones, criterios de aceptación, historias de usuario) con trazabilidad
* Investigar el trabajo 🡪 definición de requerimientos: cuando hay mucho conocimiento de BUCs o es mantenimiento y hay buen conocimiento del negocio
* Investigar el trabajo 🡪 construcción: vínculo, entendimiento y confianza con el proveedor. Relaciones comerciales de larga duración. A partir del BUC se implementan productos que cumplan con las necesidades

Estrategia iterativa



* Se construye en incrementos pequeños. Cada iteración libera funcionalidades que aportan valor y tienen feedback. SI hay algo para corregir, es más barato

Blastoff/kick off del proyecto

* Reuniones con tareas previas y entregables. En la presentación del proyecto se acuerda quiénes y cómo van a trabajar. A lo largo del proyecto va cambiando
* Entregables
  + Propósito del proyecto: descripción cualitativa de las intenciones y beneficios. Justifica el objetivo y por qué se invierte
  + Alcance del trabajo: áreas de negocio afectadas con la puesta en marcha
  + Stakeholders: interesados en el producto
  + Restricciones: limitan el proyecto
* Glosario: terminología usada en el proyecto, no puede tener dos significados
* Costo estimado (ROM): difícil de dimensionar, lo que más le interesa al cliente
* Riesgos: lista preliminar y significativa, los que tengan más probabilidad o impacto
* Go/no go decisión: luego del análisis se decide si es viable y si cumplirá los objetivos. Es el momento de cancelar si no

Evolucionando requisitos

* Corrección: corregir defectos
* Adaptación: modificar por cambios externos
* Mejora: ampliar los requisitos funcionales originales a petición del cliente
* Prevención: facilitar el cambio en el futuro

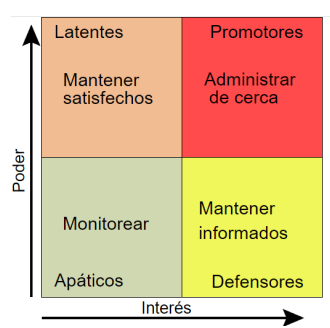
Alcance

* Área afectada por el proyecto, límites, BEs. Área de trabajo: qué sección del cliente corresponde esto

Diagrama

Descripción generada automáticamente

Matriz Mendelow



BUC (business use case): lo que hace el negocio

* Particionar el trabajo a través de los BE da buenos resultados. Las respuestas a BE son los BUC
* Alcance y reglas definidas, procesos identificables, datos que se recuperan y/o almacenan, salida generada, mensajes enviados, o combinación de estos
* Unidad de funcionalidad para escribir los requerimientos funcionales y no funcionales

PUC (producto use case): automatización de un BUC

* Lo que está fuera del alcance del producto es el actor, manipula la funcionalidad del PUC. Al determinar el PUC se determinan los actores

Técnicas de relevamiento

* Apprenticing: cuando se quiere reimplementar partes del trabajo y los sistemas actuales
  + Asumir papel de aprendiz, hacer preguntas y observar/hacer el trabajo
  + Cuesta explicar el trabajo que se hace con detalle suficiente, es mejor ver el trabajo como sucede
* BUC Workshop: para cambios fundamentales en el trabajo y debe incluir a los interesados especializados adecuados
  + Es un escenario de caso normal que describe el trabajo deseado realizado por el BUC. Contempla excepciones
* Entrevistas: recopilar requisitos del proyecto
  + No depender solo de ellas, usar otras técnicas
  + Entrevistador depende del entrevistado para obtener información, pero el entrevistado está limitado al conocimiento de su propia área
* Reutilización
  + Supuestos:
    - Es probable que alguien de la organización haya hecho un trabajo similar y escribió requisitos para un producto similar
    - Base de conocimiento sólida como para navegar y encontrar esos requisitos
  + Establecer similitudes entre procesos y usar requisitos abstractos para ahorrar tiempo en proyectos posteriores
* Dirty process modeling: cuando se reemplaza gran parte del sistema legacy
  + Modelos rápidos de procesos de negocios, son informales y construidos con cosas físicas (ejemplo: post it). Los modelos no son para diseñar pantallas o detalles, sino para ver procesos
* Prototipos low/high: para elicitar requisitos
  + Esbozar un producto y desglosar los requisitos (simplificarlos y aclararlos)
  + Útil cuando el producto no existe, los interesados no tienen experiencia con el producto, analistas no entienden lo que se quiere
  + Prototipos desechables, no evolucionan
* Mind map: organizar pensamientos, ver ideas en un diagrama y obtener detalles y visión general al mismo tiempo
  + Se pone un tema central y se escribe/dibuja alrededor de él. Los dibujos reemplazan palabras y son más fáciles de recordar
* Wikis, blogs, discusión forums: para descubrir requisitos en proyectos grandes
  + A la gente le gusta contribuir
* Document archaeology: modificar o renovar un sistema existente o heredado, útil en modelado de datos
  + Buscar requisitos en informes y archivos existentes. Se debe usar en conjunto de otras técnicas
  + Al investigar, se hacen preguntas sobre lo que se encuentra, y se obtienen requisitos, material de fondo, sugerencias para futuras investigaciones
* Family therapy: que todos vean las posiciones del resto aunque no estén de acuerdo
  + No busca que todos estén de acuerdo, “disagree and commit”
  + Útil en grupos diversos de personas
* Persona analyisis: productos atractivos, relevantes, útiles y centrados en usuarios
  + Para hacer personas se investigan a los usuarios, entrevistas, estudios de mercado y análisis de datos. Se usan patrones y características comunes para agrupar en arquetipos
  + Las personas guían las decisiones de diseño y funcionalidades. Desarrolladores las usan para pruebas de usabilidad, evaluar escenarios de uso, definir requisitos
  + Categorías de las personas
    - Habilidades: educación, habilidades, pasatiempos, alfabetización, familiaridad con la información, intereses
    - Preocupaciones: necesidades y preocupaciones
    - Objetivos: lo que buscan lograr con el producto
    - Tecnología: competencia tecnológica, tecnologías con las que interactúan normalmente

Escenarios

* Se identifica el BE, se descubren las funcionalidades que responden al BE y se escribe un escenario de BUC. Luego se determina cuánto del BUC se automatiza con PUC
* Herramienta que a los analistas explorar y definir un BUC de forma simple para las partes interesadas
* Plantilla
  + Titulo
  + Business event
  + Business use case name
  + Trigger
  + Precondiciones
  + Interesados
  + Interesados activos
  + Pasos del caso normal
  + Alternativas (varias opciones posibles)
  + Excepciones (desviaciones no deseadas pero pasarán en algún momento)
  + Resultado

Valor: lo que el cliente está dispuesto a pagar

* Recompensa: lo que se gana al tener una funcionalidad
* Penalización: lo que se pierde al no tenerla
* Costo: lo que hay que pagar para tenerla
  + R y P >= C priorizar
  + R o P >= C evaluar
  + R y P < C descartar

Design thinking: proceso no lineal e iterative para entender a los usuarios, cuestionar supuestos, redefinir problemas y crear soluciones innovadoras a través de prototipos y pruebas

* Abordar problemas mal definidos o desconocidos y centrarse en las necesidades de los usuarios
* Empatizar: entender empáticamente el problema, en general al investigar usuarios
* Definir: analizar lo anterior para definir los problemas centrales
* Idear: generar ideas creativas e innovadoras para los problemas
* Prototipar: producir versiones reducidas para investigar las ideas generadas
* Probar: poner a prueba los prototipos, iterar, refinar, recibir feedback de usuarios

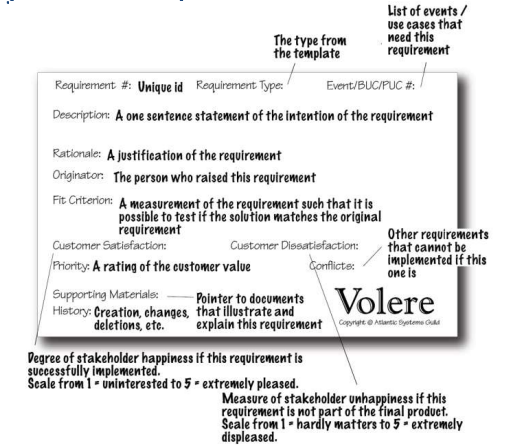
Consider the users

* Producto atractivo para el público objetivo, lo que es más adecuado para ellos. Importa más cómo se siente el usuario que las funcionalidades
* Metodologías (descubrimiento, no dan soluciones)
  + Fly on the wall: analizar el contexto sin interactuar
  + Entrevista contextual: corta, para ver lo que está pasando
  + Mystery shopper: probar productos con rol de persona normal

Innovación: pensar diferente para encontrar una mejor manera de hacer el trabajo o encontrar un mejor trabajo por hacer

* Conexión: brindar una mejor conexión al cliente con los productos (buena atención al cliente, recordar los datos)
* Conveniencia: clientes dispuestos a pagar más por productos con comodidad
* Información: clientes necesitan mucha información y rápido (dar información necesaria para tomar decisiones, información útil sin confusiones, información relevante al cliente)
* Feeling: productos aceptados o rechazados según se siente el usuario

Sistemas adyacentes

* Los BE casi siempre son externos al sistema, cuando los sistemas adyacentes hacen algo
  + Activos: humanos interactuando con el sistema, fuera del alcance del sistema. Proporcionan datos, responden preguntas, indican opciones, etc
  + Autónomos: organismos no automatizados externos al trabajo que no interactúa directamente con el trabajo y es independiente de él. Se comunica de forma unidireccional (no esperan respuesta, no hay ida y vuelta). Se busca involucrarlo más para mejorar el resultado final
  + Cooperativos: sistemas automatizados que colaboran con el trabajo durante el BUC (forman parte del BUC y del sistema). Respuestas rápidas bidireccionales (solicitud–respuesta), cajas negras (no se suelen cambiar)
* id
* tipo, evento/buc/puc
* descripción
* razón fundamental (rationale): razón de existencia, da contexto
* criterio de ajuste (fit criterion): medir estándares de calidad
* satisfacción del cliente
* dis satisfacción del cliente
* prioridad
* conflicto: si algún requerimiento depende de este
* supporting materials

historia

Requisitos no funcionales

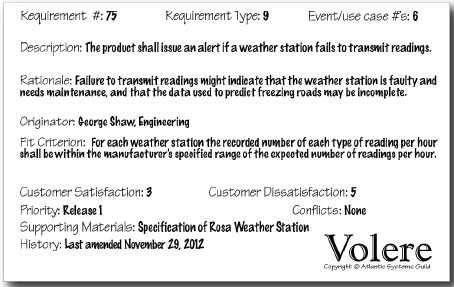
* atractivo, usabilidad, rapidez, confiabilidad, seguridad
* deben ser medibles, verificables, limitantes
* pueden ser la razón de desarrollo del producto
* por más funcionalidades que hayan, si al usuario no le gusta no lo va a usar
* para encontrarlos: apoyarse en la comunidad, en algunos requerimientos funcionales pueden aparecer, cliente
* tipos
  + look and feel: espíritu, sensación, estilo
  + usabilidad: qué tan fácil de usar es la aplicación
  + rendimiento: funcionar a cierta velocidad o poder de procesamiento
  + operacional y ambiental: condiciones para operar correctamente en su ambiente
  + mantenimiento y soporte: servicio que se ofrece una vez terminado el desarrollo, enfatizando los tipos de tratamiento en caso de fallos, prevención de usuario o necesidad de apoyo
  + seguridad
    - acceso: accesible para usuarios autorizados
    - privacidad: datos protegidos contra divulgación no autorizada
    - integridad: datos son los mismos que la fuente, no se corrompen
    - auditoría: verificar operaciones y movimientos de datos
  + cultura y política: factores que no hagan aceptable el producto por costumbres, religiones, lenguajes, tabús. Debe adaptarse a cultura, institución, contexto
  + legal: restricciones legales

Criterios de aceptación: ver que los requisitos son satisfechos y que no tienen ambigüedades ni subjetividades. Los requisitos tienen validaciones de satisfacción de los clientes

* Para los funcionales, basta con test cases. Para los no funcionales, hay que probar fallas de producto, pruebas subjetivas, estándares, rendimiento, UX/UI, operacional y ambiental, mantenimiento y soporte, seguridad, cultura y política, legal
* Definir un criterio de aceptación, tener una cuantificación del requerimiento y el estándar que el producto debe alcanzar. Los requerimientos no funcionales deben tener una razón y criterio adecuado
* A pesar de que el criterio de aceptación pueda ser costoso de probar, es mejor hacerlo lo más temprano posible para no desperdiciar recursos. Ejemplo: encuesta luego de lanzar el producto, ya se lanzó y se perdió mucho si no le gusta a la gente
* Pruebas subjetivas: ejemplo “no ofensivo para ningún grupo”. El criterio de ajuste puede ser “no ofensivo para al menos el 85% de un panel de prueba que represente a los usuarios”
* Estándares: dan un punto de referencia. Son conocidos, se reduce el tiempo de capacitación y se evitan discusiones sin sentido porque las cosas son así
* UI: impresión del usuario al usar el producto. El requisito puede ser “usar los colores de la marca”, la justificación “mejorar el reconocimiento de la marca”, el criterio de aceptación “60% de los usuarios reconocerán la marca en los primeros 5 segundos”
* UX: en general requiere ser fácil de usar o aprender, capaz de ser usado por ciertos usuarios, certificado para cumplir con leyes de discapacidad. Los criterios de aceptación pueden cuantificar tiempo permitido para una tarea, tasa de error, calificaciones de satisfacción
* Performance: velocidad, precisión, capacidad, disponibilidad, confiabilidad, escalabilidad. En general se cuantifican por sí mismos, debería ser fácil escribir criterios de aceptación. Ejemplo: un criterio de ajuste puede ser “el tiempo de respuesta no debe ser superior a 0,5 segundos para el 95% de las respuestas y no más de 2 segundos para el resto”
* Operacional: entorno en el que el producto funcionará, pueden ser condiciones adversas o inusuales. El criterio de ajuste es una cuantificación del uso exitoso en el entorno requerido, capacidad de resistir las condiciones. Ejemplo de criterio de aceptación: un médico de covid con 2 guantes puestos debe poder usar la app
* Mantenibilidad/soporte: tiempo promedio de resolución de incidencias, porcentaje de código con comentarios, disponibilidad de pruebas automatizadas, nivel de satisfacción del usuario
* Legales: cumplir con la ley

Quality Gateway (puerta de calidad): intenta que cada requisito sea lo más perfecto posible antes de incluirlo en la especificación. Debe haber criterios de aceptación y tienen que pasar pruebas. Lo que sea ambiguo se hace de nuevo

* Los requisitos fuera de alcance pueden generar descontrol y ser costosos en tiempo y presupuesto. Si un superior pide una funcionalidad cuando está avanzado el proyecto, hay que validarlo antes de incluirlo
* Test de completitud: la plantilla debe estar completa. Si falta algo, decir por qué



Gold plating: requisitos innecesarios que contribuyen más al costo que a su funcionalidad. No importa si el requisito está o no

Requirements creep: agregación de nuevos requisitos a la especificación luego de que se completen. Pueden tener hasta un 30% del costo total del proyecto como impacto en el presupuesto

* QG controla esto
* Surge porque
  + Requisitos no recopilados adecuadamente al principio (no son relevantes para el objetivo del proyecto ni para el alcance)
  + Presupuesto original es irreal
  + Cambios en el negocio o avances tecnológicos