INTRODUCCIÓN A LA INGENIERÍA DE REQUISITOS

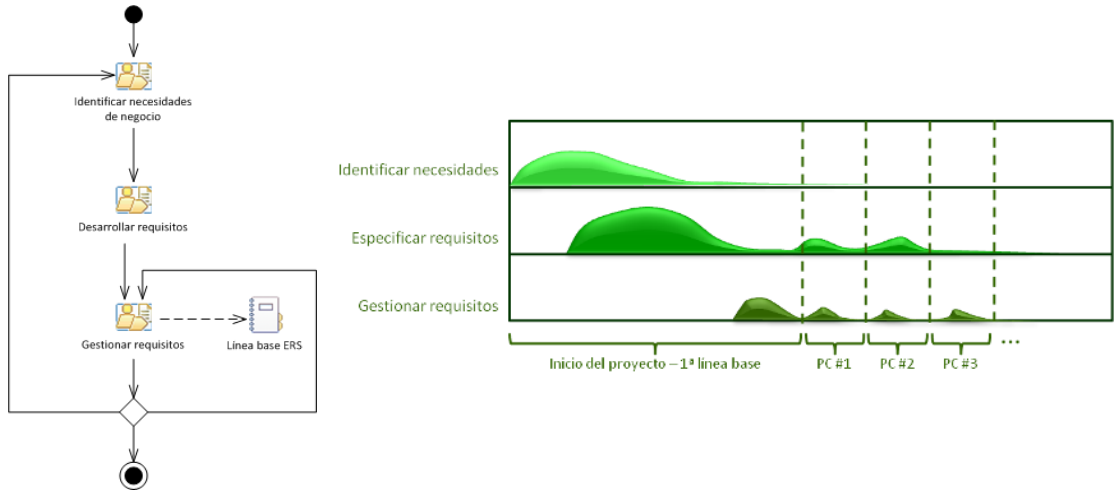
Introducción

- La ingeniería de requisitos es el conjunto de actividades y tareas del proceso de desarrollo de sistemas software que tiene como objetivos:
  - Definir, con la mejor calidad posible, las características de un sistema software que satisfaga las necesidades de negocio de clientes y usuarios y que se integre con éxito en el entorno en el que se explote. La definición de dicho sistema se realiza mediante lo que se conoce como una especificación de requisitos.
  - Gestionar las líneas base y las peticiones de cambios que se vayan produciendo en la especificación de requisitos, manteniendo la trazabilidad entre los requisitos y otros productos del desarrollo.
- La ingeniería de requisitos es una pieza clave para proporcionar un sistema con calidad, entendida como la satisfacción del usuario ante el sistema proporcionado, que cubre las expectativas, deseos y necesidades que los usuarios manifestaron y que se supieron recoger e implementar.
- El resultado de esta tarea o actividad no es estático, ya que a lo largo del proyecto pueden aparecer nuevos requisitos, ampliaciones, incluso eliminaciones o modificaciones de los existentes. Cuanto más tarde descubramos requisitos nuevos o haya desviaciones entre los requisitos y el producto, mucho mayor impacto tendrá en tiempo y coste. Desde este punto de vista, se tiene que considerar la trazabilidad de los requisitos como aspecto fundamental en la gestión de un proyecto. Es decir, actualizar los requisitos del proyecto conforme se vayan produciendo tales cambios, pero sin olvidar la actualización, el impacto y la coherencia de la documentación asociada al mismo: análisis del sistema, diseño, pruebas de validación, etc.

Ingeniería de Requisitos

- La Ingeniería de Requisitos es una de las partes cruciales en el éxito de todo proyecto software.
- La aparición de errores o carencias durante la recogida de requisitos implica un descenso en la productividad del proceso de desarrollo y, por lo tanto, un incremento del coste del mismo.
- Incluir una adecuada ingeniería de requisitos en el ciclo de vida del software minimizará la posibilidad de que esto ocurra.
- La Ingeniería de Requisitos se convierte en pieza clave para poder medir la calidad de un sistema informático al poder iniciar la definición de la batería de pruebas que el sistema debe pasar, garantizando que éstas satisfacen los requisitos establecidos y por lo tanto el sistema es válido y funcionalmente es correcto.
- Actividades:
  - Identificar las necesidades de negocio de clientes y usuarios
  - Desarrollar los requisitos de un sistema software que satisfaga las necesidades de negocio
  - Gestionar los requisitos del sistema software a desarrollar

Ingeniería de Requisitos



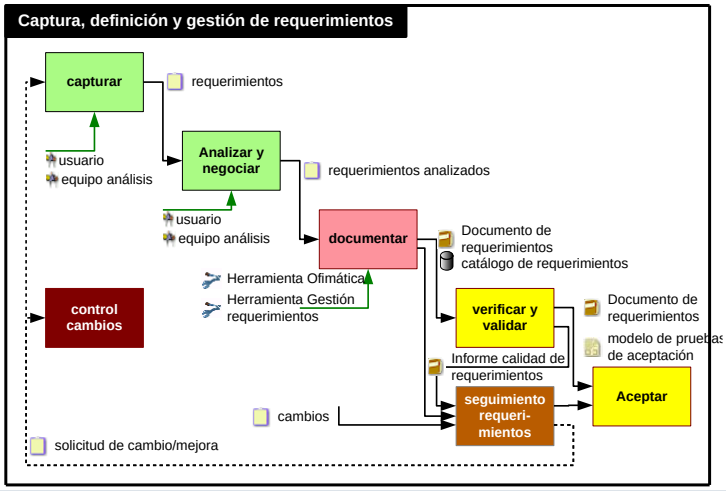
Captura de los requerimientos

- La captura de los requerimientos es la fase inicial del desarrollo de un sistema de información. La parte más difícil de construir un sistema es precisamente saber qué construir.
- Un requerimiento puede ser:
  - Una condición o capacidad necesaria para un usuario para resolver un problema o alcanzar un objetivo
  - Una condición o capacidad que debe estar presente en un sistema o componentes de un sistema para satisfacer un contrato, estándar, especificación u otro documento formal
- El proceso de captura de requisitos es la fase más crítica y complicada del desarrollo:
  - Crítica porque es la base de todo el desarrollo.
  - Complicada porque la gran variedad de posibles situaciones impide desarrollar técnicas formales de elaboración.
- Pese a lo anteriormente expuesto detallaremos una serie de pasos que nos permitan la captura de los requisitos y la elaboración de la documentación de requerimientos. La documentación obtenida sirve como base a las restantes fases del desarrollo.

Proceso

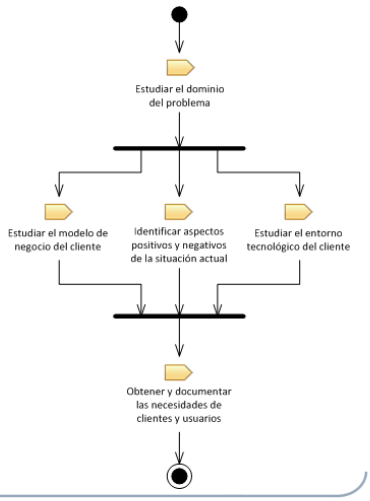
- El proceso de definición de los requisitos cuenta con una serie de pasos:
  - Preparación, captura, análisis, documentación y verificación.
- El proceso se repite tantas veces como sea necesario para la obtención de todos los requisitos. Este procedimiento se denomina refinamiento sucesivo.
- No intentamos obtener toda la información desde el principio, sino que empezamos por lo más básico y lo vamos completando hasta obtener el resultado final.
- Los pasos a seguir no son una secuencia completa, normalmente damos los tres primeros y volvemos al segundo y en paralelo vamos documentando. Aprovechamos el segundo para realizar las verificaciones, y seguimos así sucesivamente.

Proceso



Preparación

- Normalmente lo más complicado es saber por dónde empezamos; en nuestro caso podemos usar como partida la petición del cliente. En dicha petición se han definido una serie de necesidades y requisitos con los que debe contar el nuevo desarrollo. Asimismo el cliente puede aportar documentación adicional que nos permita conocer su negocio. Entre la documentación podemos encontrar:
  - El modelo de negocio, el manual de calidad, los manuales de los sistemas de información que actualmente se encuentran funcionando, peticiones de usuarios, etc., e incluso podemos buscar productos similares que ya existan en el mercado.
- De especial relevancia es el manual de calidad, dado que es un documento que especifica clara y detalladamente cómo se deben realizar todos los procesos para asegurar la calidad final. Todos los desarrollos que hagamos deben estar en consonancia con dicho manual.
- Una vez recogida la información pasamos a su estudio y estructuración, lo que nos permitirá hacernos una composición de lugar.



Captura

- Mediante entrevistas sucesivas se irán identificando las necesidades del cliente y detallando el proceso seguido para realizarlas. Las entrevistas se realizarán al cliente y a los usuarios implicados en el nuevo sistema.
- Es de vital importancia que prepares un guion de la reunión que vayas a tener, en él fijarás las líneas maestras de la información que deseas obtener.
- Las primeras reuniones serán más generales y ambiguas, y según se vaya avanzando se irá concretando mucho más los detalles. Esto te evitará reuniones interminables, llenas de divagaciones y con unos resultados escasísimos. En muchos casos el cliente no sabe lo que quiere o tiene unas nociones muy vagas de lo que pretende conseguir. En otros casos no sabe cómo resolverlo y nos utiliza para descubrirlo.
- Hay que tener en cuenta que muchos clientes como conocen su negocio asumen que el resto también, por lo que con dar unas indicaciones piensan que es suficiente, y es necesario que las describa.
- La obtención de la información del cliente es una de las cosas más complicadas, dado que excede el ámbito propiamente informático e implica, en muchos casos, que adquieras un conocimiento exhaustivo de su negocio.

Análisis

- El análisis de los requisitos comienza con la enumeración de los requisitos candidatos. Debes comenzar elaborando una lista de características con todas las ideas que podrían convertirse en requisitos. La lista crecerá según vayan apareciendo nuevas características y menguará según se descarten o se conviertan en requisitos.
- A cada característica le das un nombre corto y una breve descripción. Debes asignar a cada una un estado (propuesto, aprobado, incluido o validado), una prioridad (crítico, importante o secundario) y un coste (alto, medio o bajo).
- El estado y la prioridad deben ser negociados con el cliente. Sólo los requisitos validados podrán pasar a las siguientes fases de desarrollo. La prioridad servirá para determinar la planificación del proyecto.
- El análisis de las características debe llevarte a la comprensión del contexto del sistema.
- Cuando tengas las características definidas, pasarás a convertirlas en requisitos mediante la captura de las especificaciones funcionales, donde se detallará cómo se usará el sistema.
- Adicionalmente, en muchos casos, es necesario capturar los requisitos no funcionales. En dichos requisitos especificarás todas las propiedades del sistema que imponga el cliente, como son las restricciones del entorno o la implementación, rendimientos, plataformas, etc. Estas características definen cómo debe ser el producto final en vez de decir qué debe hacer.

Documentación

- Todos los requerimientos se deben plasmar en un documento que recoja las especificaciones que permitan elaborar el sistema de información. Dicho documento contará con múltiples usos:
  - Formará parte de la relación contractual con el cliente, es la base del desarrollo, y servirá de referencia para posteriores modificaciones y mantenimientos.
- Hay una gran variedad de formas de crear la documentación en cuanto a la estructura y el lenguaje empleado. Se puede utilizar el lenguaje natural o algunos de los lenguajes formales. Las restricciones impuestas en los lenguajes formales, como los casos de uso o las historias de usuario, permiten una mayor calidad en el resultado final.
- Existen toda una serie de criterios a seguir para asegurar la calidad de la documentación elaborada:
  - La documentación debe ser correcta y completa. Es correcta cuando todo requisito que aparece es necesario, representa algo requerido del sistema y que se puede implementar. Es completa cuando aparecen todos los requisitos del sistema.
  - No debe ser ambigua, es decir, el mismo segmento leído por personas diferentes no debe prestarse a múltiples interpretaciones. Los lenguajes formales evitan gran parte de las ambigüedades.
  - Todos los requisitos que aparezcan deben ser verificables, es decir, que se pueda realizar pruebas que validen su funcionamiento. Decir que “El producto debe trabajar bien”, es como no decir nada, porque ¿qué se entiende por “bien”?. O decir “habitualmente” que no indica métrica alguna. En caso de ser necesario definir un requisito de este tipo es necesario establecer la métrica en valores absolutos (3 de cada 4 veces) o porcentuales (en el 90% de los casos).
  - Los requisitos deben ser consistentes. No debes usar términos diferentes para la misma cosa, es preferible tener un documento más pesado que crear confusión al no saber si te refieres a la misma cosa o a cosas diferentes. Especialmente en los documentos largos debes cuidar el caer en contradicciones al definir en un punto un comportamiento o característica de una forma y en otro punto de otra forma diferente.

Documentación

- Hay una serie de atributos propios de la documentación en sí misma:
  - La documentación la debes redactar para que sea comprensible para todos los posibles consumidores, que como vimos anteriormente son muchos, desde el usuario hasta el personal técnico de desarrollo.
    - A ser posible, que sea concisa, en igualdad de condiciones la más corta es la mejor, dado que el nivel de atención es inversamente proporcional al número de hojas a leer.
    - No utilizar 'y/o' por que componen varios requisitos. Trata de hacer oraciones simples, cortas y significativas usando la voz activa que puedan ser probadas unitariamente.
    - Garantizar un nivel similar de consistencia y uniformidad al explicar los requisitos.
  - Utiliza un estilo y una estructura que permita su fácil modificación, y que dicha modificación no incumpla algo de lo anteriormente dicho. Es conveniente que se encuentre organizada correctamente, todos los elementos que aparezcan en la documentación deben encontrarse correctamente numerados, nominados y referenciados. Los diversos tipos de índices permiten acceder de una forma rápida a la información, es preferible que sobren a que falten (normalmente los procesadores de texto te permiten su generación de forma automática).
  - Al hilo de lo anterior, no olvides hacer referencias en todos los requisitos al origen de los mismos y a su destino.

Verificación

- La verificación de los requerimientos la puedes realizar de varias formas diferentes:
  - Las revisiones constituyen una forma de probar los productos de trabajo del software y pueden realizarse antes de ejecutar las pruebas dinámicas. Los defectos detectados durante las revisiones de los requisitos a menudo son mucho más baratos de eliminar que los detectados durante las pruebas realizadas ejecutando el código.
  - Puedes elaborar un plan de pruebas, una serie de casos prácticos, y estudiar cómo responden las especificaciones, detectando carencias o resultados inesperados.
  - Otra opción es elaborar un prototipo de las especificaciones. Los problemas en su elaboración detectarán las incorrecciones en las especificaciones. Una vez elaborado, su prueba por parte del cliente, detectará carencias o errores de entendimiento.
- Una vez superado el paso de validación, la documentación de requerimientos es presentada al cliente para su aceptación. Con dicha aceptación concluye la presente fase requerimientos. En refinamientos sucesivos repetirás la fase.

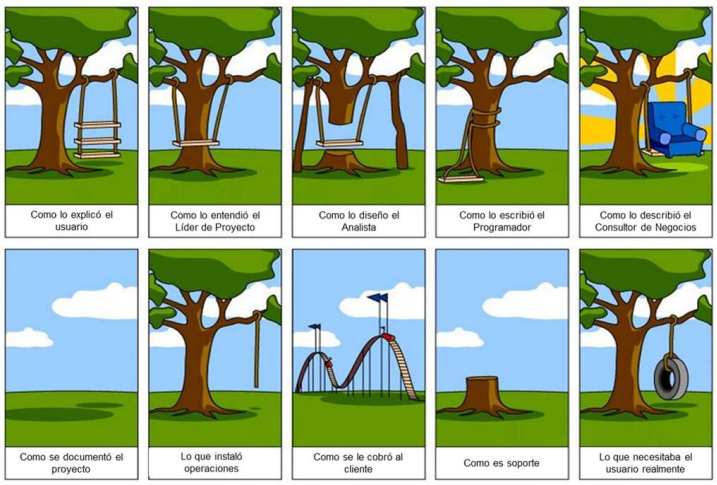
Verificación

- Hay muchas figuras y formas de requisitos, pero lo importante es que todos los involucrados en un proyecto que manejan requisitos deben asegurarse de comprender lo que significan. Hay muchas maneras de asegurarse de esto. Como mínimo, todos los involucrados deberían hacerse las siguientes preguntas:
  - Coherencia: ¿Existen requisitos que se contradicen entre sí de alguna manera?
  - Integridad: ¿Los requisitos describen todos los atributos del sistema a implementar?
  - Verificabilidad: ¿Es posible comprobar si un requisito se ha construido correctamente?
  - Trazabilidad: ¿Es posible comprobar el estado de este requisito en todas las etapas del desarrollo del software?
  - Atomicidad: ¿Es este requisito lo más simple posible (pero no más simple)? Una comprobación sencilla de esto es si el requisito contiene palabras como 'y', 'o' y pero'.
  - Estructuración: ¿Todos los requisitos tienen la misma estructura?.
  - Viabilidad: ¿Se puede lograr el requisito dado su estado actual de tiempo, presupuesto y capacidades?
  - Comprensibilidad: ¿Pueden todos los involucrados en el proyecto entender lo mismo? ¿Se usan sinónimo (varios términos para la misma cosa) o polisemia (mismo termino para distintas cosas)?

Gestión del cambio

- Las organizaciones no son entes estáticos que no cambian, son entes dinámicos que evolucionan y se adaptan, por lo que los requisitos no son tan estables como sería deseable.
- Es frecuente que sea necesario cambiar requisitos ya aceptados. Dichos cambios pueden ser de modificación, inclusión o eliminación de requisitos. Es fundamental para la buena marcha del proyecto realizar un control y seguimiento de los cambios mediante la Gestión de Requerimientos.
- Los pasos que debes dar para realizar la Gestión de Requerimientos son los siguientes:
  - Evalúas la petición del cambio, decidiendo si es o no es procedente.
  - Analizas el impacto que produce cualquier cambio de un requisito, basándote en las relaciones de los requisitos cambiados con elementos de fases posteriores del ciclo de desarrollo. Para ello debes realizar el paso de captura.
  - Formalizas el cambio de los requisitos, realizando la documentación pertinente.
  - Realizas el seguimiento del cumplimiento de los requisitos a través de las relaciones entre ellos con elementos de fases posteriores, y el seguimiento de los cambios desde el momento de su aprobación.

Ingeniería de Requisitos



Lenguaje de modelado unificado (UML)

- El lenguaje de modelado unificado (UML) se ha convertido rápidamente en el estándar de facto para construir software orientado a objetos.
- La especificación de Object Management Group (OMG) establece:
  - *"El lenguaje de modelado unificado (UML) es un lenguaje gráfico para visualizar, especificar, construir y documentar los artefactos de un sistema intensivo en software. El UML ofrece una forma estándar de escribir los planos de un sistema, incluyendo elementos conceptuales como procesos comerciales y funciones del sistema, así como cosas concretas, como declaraciones de lenguaje de programación, esquemas de bases de datos y componentes de software reutilizables"*.
- El punto importante a tener en cuenta aquí es que UML es un 'lenguaje' para especificar y no un método o procedimiento. El UML se usa para definir un sistema de software; para detallar los artefactos en el sistema, para documentar y construir: es el lenguaje en el que está escrito el plan. El UML se puede usar de varias maneras para soportar una metodología de desarrollo de software (como el Proceso Racional Unificado), pero en sí mismo no especifica esa metodología o proceso.

Modelos de comportamiento UML

- Los diagramas de comportamiento de UML representan los elementos de un sistema que dependen del tiempo y que transmiten los conceptos dinámicos del sistema y cómo se relacionan entre sí.
- Los elementos en estos diagramas se parecen a los verbos en un lenguaje natural y las relaciones que los conectan típicamente transmiten el paso del tiempo.
- Por ejemplo, un diagrama de comportamiento de un sistema de reserva de vehículos puede contener elementos como hacer una reserva, alquilar un automóvil y proporcionar detalles de la tarjeta de crédito.
- Los modeladores experimentados mostrarán la relación con los elementos estructurales en estos diagramas.

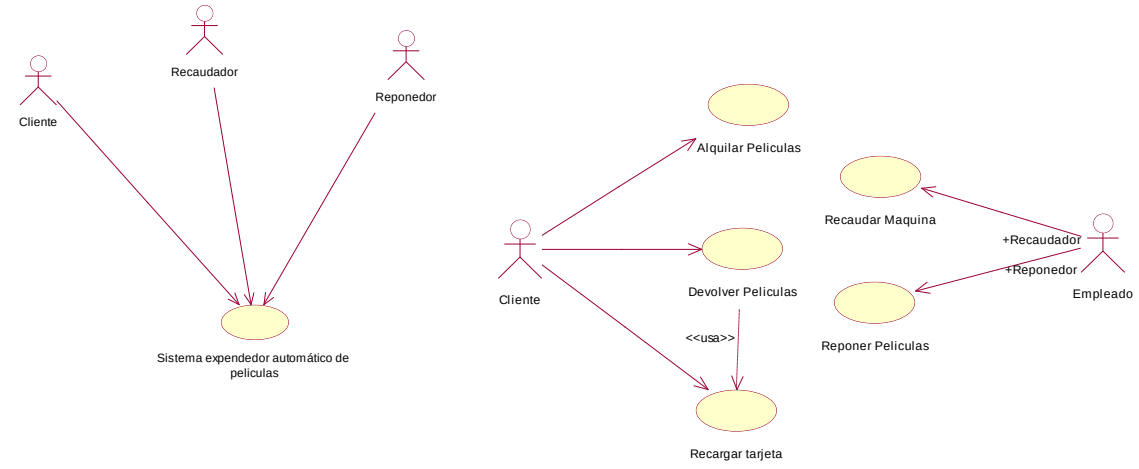
Modelo de caso de uso

- El modelo de casos de usos es un modelo de comportamiento que describe completamente la forma en que se va a utilizar al sistema, fundamentalmente desde el punto de vista del usuario. Un modelo de caso de uso describe la funcionalidad propuesta de un nuevo sistema.
- Permite que los clientes y los desarrolladores lleguen a un acuerdo sobre los requisitos. Sirve como acuerdo entre el cliente y los desarrolladores, y proporciona la entrada fundamental para el análisis, el diseño y las pruebas.
- Un actor (representado por un monigote) es un usuario del sistema; usuario puede significar un usuario humano, una máquina o incluso otro sistema o subsistema en el modelo.
- Un caso de uso (representado por un ovalo) representa una unidad discreta de interacción entre un actor (humano o máquina) y el sistema. Esta interacción es una sola unidad de trabajo significativo, como Crear cuenta o Ver detalles de la cuenta.
- Cada caso de uso describe la funcionalidad que se construirá en el sistema propuesto, que puede incluir la funcionalidad de otro caso de uso o extender otro caso de uso con su propio comportamiento.

Modelo de caso de uso

- El modelo de casos de uso puede hacerse bastante grande y difícil de digerir, de forma que es necesario algún medio de abordarlo en trozos más pequeños. El modelo está compuesto por una serie de diagramas de casos de uso.
- Crearemos un diagrama especial, denominado diagrama de contexto. En él representarás las relaciones de comunicación entre los actores y el sistema.
- A continuación crearías un diagrama general con los principales casos de uso con las relaciones entre ellos y con los actores.
- A partir de este momento iremos refinando el modelo, creando nuevos diagramas por cada caso de uso cuya complejidad lo requiera.
- ¿Hasta qué nivel?
  - El eufemismo es “hasta que sea necesario”, es decir, que el grado de complejidad de cada caso representado en el diagrama sea mínimo. El grado de complejidad es algo muy subjetivo, debes preguntarte si todos los lectores del documento serán capaces de entenderlo sin que tengan dudas; piensa que la audiencia es muy variada, desde el cliente con sus usuarios hasta tu propio personal técnico de desarrollo.

Diagrama de contexto y Diagrama general



Especificación de los casos de uso

- Hasta ahora hemos visto cómo representar de una forma muy simple y visual los requisitos del sistema pero, aunque una imagen vale por mil palabras, los diagramas no son suficiente para registrar toda la información.
- En este momento entramos en la parte más desagradecida del modelo, la parte textual, completar los casos de uso mediante una descripción.
- La especificación de cada caso de uso no la haces de golpe. Empiezas dando un nombre corto pero significativo y una breve descripción. Según vas profundizando introduces nuevos detalles hasta que se encuentra completamente definida.
- A continuación detallamos todo lo que debe contener, pero como vimos irán surgiendo según avancemos en el modelo, no quiere decir que surjan en ese orden:
  - La descripción del caso de uso debe definir el estado inicial (precondición) y los posibles estados finales (postcondiciones). Cómo y cuándo comienza el caso de uso, la primera acción a realizar, y cómo y cuándo puede terminar. El orden requerido (si hay alguno) en el que las acciones se deben ejecutar, en forma de secuencia numerada, es el camino básico.
  - En caso de existir, debes detallar los caminos alternativos que han sido extraídos de la descripción del camino básico, así como los caminos que no están permitidos.
  - Es necesario que reflejes la interacción del sistema con los cambios que producen. En caso de existir, reflejas la utilización de objetos, valores o recursos de sistema. Observa que debes describir explícitamente qué hace el sistema, las acciones que ejecuta, separando claramente las responsabilidades del sistema y las de los actores.
  - Ya por último, pasas a cuantificar la prioridad y coste del mismo, indicando en qué estado se encuentra.
  - El modelo se completa con un glosario de términos, una serie de catálogos de actores, casos de uso, actores que permitan el manejo eficiente de la documentación.

Descripción de un caso de uso

- **Comentarios** generales y notas que describen el caso de uso.
- **Requisitos**: los requisitos funcionales formales de las cosas que un caso de uso debe proporcionar al usuario final. Estos corresponden a las especificaciones funcionales que se encuentran en las metodologías estructuradas y forman un contrato en el que el caso de uso realiza alguna acción o proporciona algún valor al sistema.
- **Restricciones**: las reglas y limitaciones formales con las que opera un caso de uso, que definen qué se puede y qué no se puede hacer. Éstas incluyen:
  - Condiciones previas que ya deben haberse producido o estar vigentes antes de ejecutar el caso de uso
  - Condiciones posteriores que deben ser ciertas una vez que se complete el caso de uso
  - Invariantes que siempre deben ser ciertas a lo largo del tiempo que opera el caso de uso

Descripción de un caso de uso

- **Escenarios**: descripciones formales y secuenciales de los pasos tomados para llevar a cabo el caso de uso, o el flujo de eventos que ocurren durante una instancia de caso de uso. Estos pueden incluir múltiples escenarios, para atender circunstancias excepcionales y rutas de procesamiento alternativas. Por lo general, se crean en texto y corresponden a una representación textual del diagrama de secuencia.
- **Diagramas de escenarios**: diagramas de secuencia para representar el flujo de trabajo; similar a los escenarios pero representado gráficamente.
- **Atributos adicionales**, como fase de implementación, número de versión, clasificación de complejidad, estereotipo y estado.
- **Criterios de aceptación**: lista de criterios que debe cumplir un caso de uso para ser aceptado por un usuario, cliente u otra parte interesada.

Elementos del caso de uso

- Los casos de uso generalmente están relacionados con los 'actores', que son entidades humanas o de máquinas que usan o interactúan con el sistema para realizar un trabajo significativo que les ayuda a lograr un objetivo. El conjunto de casos de uso a los que un actor tiene acceso define su rol general en el sistema y el alcance de su acción.
- Un Caso de Uso puede incluir la funcionalidad de otro como parte de su procesamiento normal. Generalmente se asume que los casos de uso incluidos se llamarán cada vez que se ejecute el camino base. Un ejemplo puede ser listar un conjunto de órdenes de clientes de las cuáles poder elegir antes de modificar una orden seleccionada; en este caso, el Caso de Uso <listar órdenes> se puede incluir en el Caso de Uso <modificar orden> cada vez que éste se ejecute.
- Un caso de uso puede ser incluido por uno o más casos de uso, por lo que ayuda a reducir la duplicación de la funcionalidad al factorizar el comportamiento común en casos de uso que se reutilizan muchas veces.
- Un caso de uso puede extender el comportamiento de otro, generalmente cuando se encuentran circunstancias excepcionales. Por ejemplo, si un usuario debe obtener la aprobación de alguna autoridad superior antes de modificar un tipo particular de pedido de un cliente, entonces el Caso de uso <obtener aprobación> podría extender opcionalmente el Caso de uso normal de <modificar pedido>.

Documentación del caso de uso

Especificación del caso de uso				
Nombre		**Código**		
Descripción				
Estado	**Prioridad**	**Coste**	**F. Modificación**	**Versión**
Requerimientos				
Actores				
Relaciones				
Precondiciones				
Camino básico				
1.				
Camino alternativo				
Poscondiciones				
Criterios de aceptación				
Cuestiones				
Notas Técnicas				

Prototipo de casos de usos

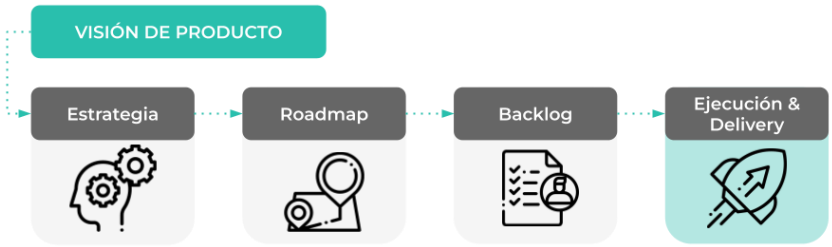
- Una práctica recomendable, aunque opcional, es la creación de un prototipo con el interfaz de usuario. El modelo de casos de uso es ideal para dicha creación.
- Fíjate que lo has elaborado desde el punto de vista del usuario y tienes claramente identificadas las interacciones con los usuarios.
- Bastará con que realices un bosquejo, aunque sea a mano, de dichas entradas y salidas para que tengas una primera aproximación del prototipo.
- Al cliente le permites ver, casi desde el primer momento, cómo va a quedar el sistema que solicita.

Principios de la metodología ágil

- Un proyecto ágil se segmenta en una serie de pasos incrementales que incluyen intervalos de feedback regulares. Cada requisito del proyecto se divide en fragmentos más pequeños, que luego se priorizan según su importancia. Promueve la colaboración, especialmente con el cliente. Permite hacer ajustes en intervalos regulares para conseguir que se satisfagan las necesidades del cliente. Integra la planificación con la ejecución, lo que permite al equipo responder de forma eficaz a los cambios de requisitos.
- Ventajas de la gestión de proyectos ágil
  - Ciclos de feedback más rápidos.
  - Los problemas se identifican más temprano.
  - Mayor potencial de satisfacción del cliente.
  - El tiempo de salida al mercado mejora drásticamente.
  - Mayor visibilidad y responsabilidad.
  - Los equipos dedicados mejoran la productividad con el tiempo.
  - Priorización flexible centrada en la entrega de valor.
- Inconvenientes de la metodología ágil
  - Es posible que las dependencias críticas entre proyectos y rutas no estén tan claramente definidas como con el modelo en cascada.
  - Coste extra de la curva de aprendizaje organizativa.
  - Implementar una ejecución verdaderamente ágil con una canalización de implementación continua supone muchas dependencias técnicas y costes de ingeniería.

Visión de producto

- La visión de producto es una definición que describe a alto nivel, el objetivo a largo plazo de tu producto. La visión de producto debe responder a la pregunta ¿Qué quieres que sea tu producto dentro de unos años?
- Una buena visión de producto facilitará el proceso de desarrollo del producto ya que es la base de otros aspectos importantes. Es importante que la visión de producto sea realista y, al mismo tiempo, lo suficientemente ambiciosa para que el equipo se entusiasme con la creación del producto.
- La visión del producto es el punto de partida, guía a la organización, a los equipos y a las personas hacia donde se quiere ir. Sin embargo, es crítico no confundir esta visión con un plan para seguir ciegamente un camino predefinido. Una definición de visión de producto eficaz debe ser clara, concisa, motivadora y, sobre todo, alcanzable. Debe inspirar al equipo y partes interesadas a crear un gran producto que satisfaga las necesidades del cliente.



Roadmaps, iniciativas y Requisitos

- La hoja de ruta describe cómo se desarrolla en el tiempo un producto o solución. En el desarrollo ágil, proporciona contexto muy útil para ayudar a los equipos a alcanzar objetivos tanto incrementales como a nivel de todo el proyecto. Las hojas de ruta están formadas por iniciativas, que son grandes áreas de funcionalidad, e incluyen cronogramas que indican cuándo estará disponible una función. A medida que el trabajo avanza y los equipos recaban nueva información, es normal que la hoja de ruta cambie para reflejarla, ya sea ligeramente o de forma más significativa. El objetivo es que la hoja de ruta siga centrada en las condiciones actuales que afectan al proyecto y en los objetivos a largo plazo, de modo que el equipo pueda trabajar eficazmente con las partes interesadas y seguir siendo competitivo.
- Las iniciativas de la hoja de ruta se dividen en una serie de requisitos. Los requisitos ágiles son descripciones breves de la funcionalidad necesaria, en lugar de los documentos extensos y formales que se asocian con los proyectos tradicionales. Evolucionan con el tiempo y aprovechan el entendimiento común que el equipo tiene del cliente y del producto deseado. Únicamente cuando la implementación está a punto de empezar, se concretan los pormenores con todo detalle.

Historias de usuario

- Un componente clave del desarrollo ágil es poner a las personas en primer lugar, y las historias de usuarios ponen a los usuarios finales reales en el centro de la conversación. Las historias utilizan un lenguaje no técnico para ofrecer contexto al equipo de desarrollo y sus esfuerzos. Después de leer una historia de usuario, el equipo sabe por qué está compilando lo que está compilando y qué valor crea.
- Se estructuran en:
  - Los epics son grandes cantidades de trabajo que se pueden desglosar en un número de tareas más pequeñas (llamadas "historias").
  - Las historias, también llamadas "historias de usuario", son breves requisitos o solicitudes escritas desde el punto de vista del usuario final.
  - Las iniciativas son conjuntos de epics que conducen hacia un objetivo común.
- Las historias de usuario son uno de los componentes centrales de un programa ágil. Ayudan a proporcionar un marco centrado en el usuario para el trabajo diario, lo que impulsa la colaboración y la creatividad y mejora el producto en general.
- Una historia de usuario es la unidad de trabajo más pequeña en un marco ágil. Es un objetivo final, no una función, expresado desde la perspectiva del usuario del software.

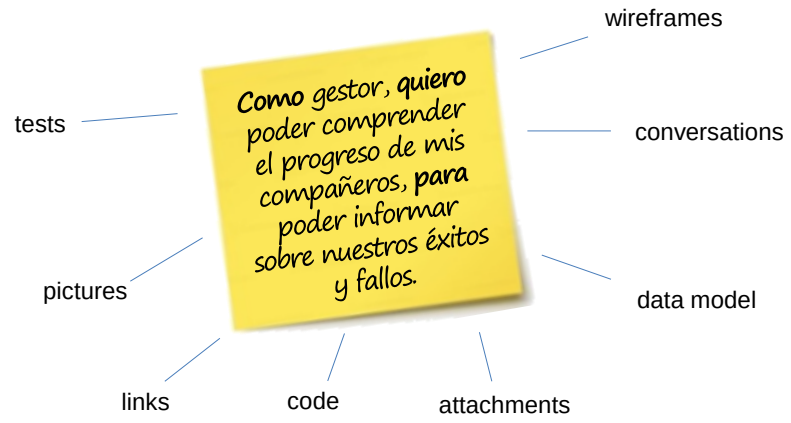
Historias de usuario

- Una historia de usuario es una explicación general e informal de una función de software escrita desde la perspectiva del usuario final o cliente.
- El propósito de una historia de usuario es articular cómo un elemento de trabajo entregará un valor particular al cliente. Ten en cuenta que los "clientes" no tienen por qué ser usuarios finales externos en el sentido tradicional, también pueden ser clientes internos o colegas dentro de tu organización que dependen de tu equipo.
- Sus características son:
  - Descripción breve de la funcionalidad que aporta valor al cliente
  - Es un recordatorio. Lo verdaderamente importante es la conversación cara a cara
  - Idealmente son independientes, aunque no siempre es posible
  - Las historias las escribe el cliente
  - El cliente prioriza las historias en función del valor aportado a la organización

Verbalización de historias

- Las historias de usuario suelen expresarse con una frase simple con la siguiente estructura (perfil + necesidad + propósito):
  - "Como [perfil] [quiero] [para]."
- Se colocan en un post-it y se le puede complementar con más datos "a la vista"
  - "Como [perfil]": ¿para quién desarrollamos esto? No solo buscamos un puesto, buscamos el perfil de la persona. Max. Nuestro equipo debería comprender quién es Max. Con suerte hemos entrevistado a muchos Max. Comprendemos cómo trabaja esa persona, cómo piensa y cómo se siente. Sentimos empatía por Max.
  - "Quiero": aquí describimos su intención, no las funciones que usan. ¿Qué es lo que están intentando lograr realmente? Esta descripción debería realizarse con independencia de las implementaciones; si describes algún elemento de la IU y no el objetivo del usuario, estás cometiendo un error.
  - "Para": ¿cómo encaja su deseo inmediato de hacer algo en la perspectiva general? ¿Cuál es el beneficio general que intentan lograr? ¿Cuál es el gran problema que debe resolverse?

Verbalización de historias



Cómo escribir historias de usuario

- Piensa en lo siguiente cuando escribas historias de usuario:
  - Perfiles de usuario: ¿para quién? Si hay varios usuarios finales, considera crear varias historias.
  - Describe tareas o subtareas: decide qué pasos específicos deben completarse y quién es responsable de cada uno de ellos.
  - Pasos ordenados: escribe una historia para cada paso en un proceso más grande.
  - Definición de "Listo": la historia suele estar "lista" cuando el usuario puede completar la tarea descrita, pero debes asegurarte de definir lo que representa completarla.
  - Escucha el feedback: habla con los usuarios y capta sus problemas o necesidades en lo que dicen. No es necesario tener que estar adivinando las historias cuando puedes obtenerlas de tus clientes.
  - Tiempo: el tiempo es un tema delicado. Muchos equipos de desarrollo evitan hablar sobre el tiempo, y en su lugar confían en sus marcos de trabajo de estimación. Dado que las historias deberían completarse en un sprint, aquellas que puedan necesitar semanas o meses deberían dividirse en historias más pequeñas o considerarse un epic independiente.
- Una vez que las historias de usuario estén definidas de forma clara, debes asegurarte de que todo el equipo pueda verlas.

INVEST

- El método INVEST es un enfoque utilizado en el desarrollo ágil de software para garantizar la calidad en la escritura de historias de usuario. Este método proporciona un conjunto de características clave que las historias de usuario deben poseer para ser efectivas en la comunicación, planificación y ejecución de proyectos ágiles.
- El método sirve para comprobar la calidad de una historia de usuario revisando que cumpla una serie de características:
  - I Independent (independiente para ser planificada e implementada en cualquier orden).
  - N Negotiable (negociable, sus detalles serán acordados).
  - V Valuable (valiosa para el cliente o el usuario).
  - E Estimable (estimable para priorizar y planificar su implementación).
  - S Small (pequeña para ser rápidamente implementada).
  - T Testable (comprobable con criterios de aceptación).

Elementos fundamentales de una historia

- Tarjeta (Card): La tarjeta refleja los elementos más importantes de la historia de usuario. Expresa el valor que se quiere conseguir desde el punto de vista del usuario. Expresar un historia de usuario desde el punto de vista del desarrollador (As a developer I want…) o desde la organización (As the organisation I want to…) es una mala práctica que refleja la incapacidad de pensar en el valor para el usuario final.
- Conversación (Conversation): Después de escribir la historia de usuario en la tarjeta, es necesario tener una conversación al respecto de su contenido. Hay que saber responder a cuestiones sobre el valor y sobre el resultado esperado de la implementación. Esta conversación puede suceder en cualquier momento; es habitual que se produzca durante el refinamiento del Backlog o el Sprint Planning.
- Confirmación (Confirmation): La confirmación es un acuerdo que refleja que todas las personas implicadas, Product Owner y Development Team, entienden cuales son los elementos, valor y resultado esperado de la historia de usuario. Muchos equipos cometen el error de no tener esta confirmación, lo cual lleva a fricciones y problemas durante el Sprint Planning.

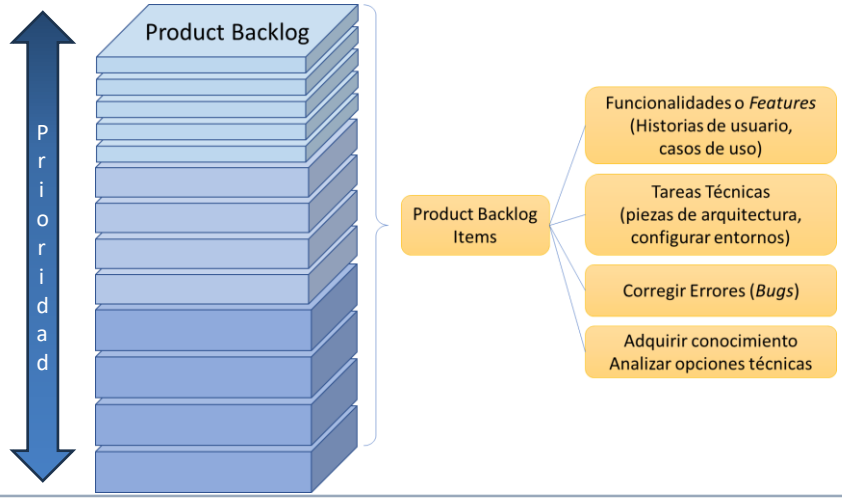
Product Backlog

- Las historia generan una lista de propiedades: el Product Backlog
  - Requisitos funcionales y no funcionales
  - Priorizados
  - Esfuerzo valorado a alto nivel
  - Al nivel de detalle suficiente para continuar
- El backlog define las prioridades del programa ágil y es mas que una lista de historias de usuarios. El equipo incluye todos los elementos de trabajo en el backlog: funcionalidades nuevas, bugs, mejoras, tareas técnicas o arquitectónicas, etc.
- El propietario del producto prioriza el trabajo del backlog para el equipo de ingenieros. Más adelante, el equipo de desarrollo utiliza el backlog priorizado como única fuente fiable para saber qué trabajo hay que hacer.
- El backlog está compuesto por todas las necesidades identificadas por el equipo de proyecto.
- El backlog es necesario para planificar las entregas e iteraciones.
- El Product Backlog es una extensión de la Visión del Producto

Product Backlog

- El Backlog de Producto en Scrum no es un mero listado de tareas.
- El Product Backlog y los elementos que lo integran, que denominaremos entradas o ítems, tienen una serie de características:
  - los ítems del Backlog deben agregar siempre valor para el cliente
  - los ítems del Backlog deben estar priorizados, esto es, ordenados según criterios de prioridad (valor de la funcionalidad para el cliente, tamaño, dificultad, etc.). Cuanto mayor sea su prioridad, más arriba deben estar en la pila.
  - el nivel de detalle de cada ítem del Backlog depende de su posición dentro de la pila.
    - Los de más arriba estarán más detallados (refinados), para poder abordarse antes.
    - Los de más abajo contendrán menor detalle, pues se abordarán más tarde y es posible que antes se modifiquen o incluso se descarten.
  - todos los ítems deben estimarse
  - el Backlog es un documento vivo, sujeto a cambios, en el que puede entrar o del que puede salir trabajo.
  - el Backlog no debe contener elementos de acción ni tareas de bajo nivel
- La mayor parte de estos puntos, se corresponden con los aspectos fundamentales que identifica el acrónimo DEEP (profundo) a la hora de elaborar un Product Backlog: **D**etallado adecuadamente (Detailed), **E**mergentes o evolutivo (Emerging), **E**stimados (Estimated) y **P**riorizados (Prioritized).

Product Backlog



Criterios de Aceptación

- Un equipo multifuncional acordará entregar un producto de TI con un nivel de calidad específico. Este nivel de calidad está definido por los criterios de aceptación. Tan pronto como el producto cumpla con estos criterios, estará "terminado". Los criterios de aceptación se especifican por caso de uso o historia de usuario. La Definición de Terminado (DoD: Definition of “Done”) definirá que una historia de usuario se completa cuando se han cumplido todos los criterios de aceptación. El propietario del producto es la principal parte interesada que participará en la definición de los criterios de aceptación, pero el equipo ayudará en esto.
- Los criterios de aceptación son los criterios que debe cumplir un objeto de prueba para ser aceptado por un usuario, cliente u otra parte interesada.
- Los criterios de aceptación ayudarán al equipo a comprender qué está incluido en el alcance y qué no está dentro del alcance de la historia del usuario.

Criterios de Aceptación

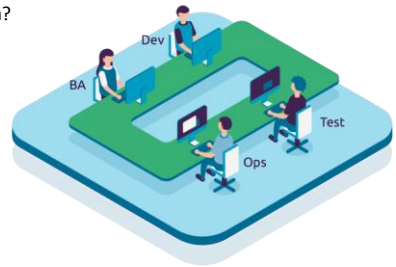
- Los criterios de aceptación pueden estar relacionados con una sola historia de usuario, por ejemplo cuando el criterio de aceptación está relacionado con una funcionalidad específica, o pueden estar relacionados con múltiples historias de usuarios, con épicas o incluso con todo el sistema o toda la organización, en cuyo caso se puede incluir en la Definición de Hecho para que no tenga que repetirse para cada historia de usuario.
- Los criterios de aceptación, al ser simples y accesibles, resuelven múltiples problemas a la vez: documentan las expectativas del cliente, brindan una perspectiva del usuario final, aclaran los requisitos y evitan ambigüedades y, finalmente, ayudan al control de calidad a verificar si se cumplieron los objetivos de desarrollo.

Definir los criterios de aceptación

- Cada caso de uso o historia de usuario debe tener al menos un criterio de aceptación.
- Los criterios de aceptación se escriben antes de la implementación de la historia de usuario.
- Cada criterio de aceptación debe poderse comprobar de forma independiente.
- Los criterios de aceptación deben tener un resultado claro de Pasa/No pasa.
- Se centran en el resultado final, el qué, y no en el enfoque de solución, el cómo.
- Se deben incluir criterios funcionales y no funcionales.
- El propietario del producto y otras partes interesadas deben colaborar estrechamente para que todos los involucrados respalden los criterios de aceptación. Los miembros del equipo escriben los criterios de aceptación y el propietario del producto los verifica.
- Los criterios de aceptación pueden especificarse como ejemplos.

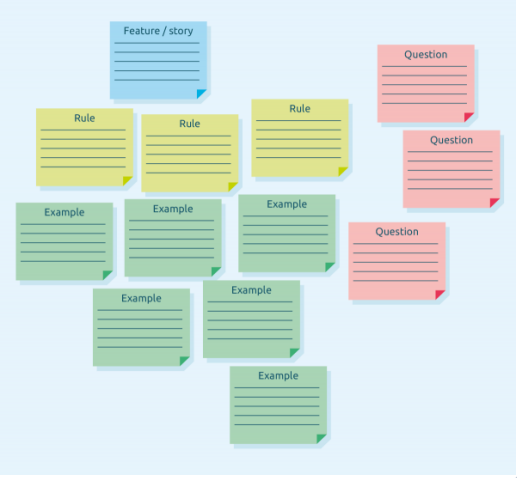
Especificación y ejemplo (SaE)

- El enfoque de Especificación y ejemplo (Specification and Example) es un enfoque colaborativo para definir requisitos y pruebas funcionales orientadas al negocio para productos de software, basado en capturar e ilustrar requisitos utilizando ejemplos realistas en lugar de declaraciones abstractas.
- Cuando se utiliza uno de los enfoques de SaE, se necesitan personas con perspectivas diferentes. Un enfoque popular es el de los "Tres amigos" [Dinwiddie 2009], que se refiere a las perspectivas principales para examinar un incremento de trabajo antes, durante y después del desarrollo. Los "Tres amigos" representan:
  - Negocios: ¿Qué problema estamos tratando de resolver?
  - Desarrollo: ¿Cómo podríamos construir una solución para resolver ese problema?
  - Pruebas: ¿Qué pasa con esto? ¿Qué podría pasar?
- En DevOps, se agrega un cuarto amigo:
  - Operaciones: ¿Cómo podríamos ejecutar y mantener la solución?
- Algunas de las metodología que utilizan este enfoque son:
  - Especificación por ejemplos (SbE)
  - Desarrollo basado en ejemplos (EDD)
  - Requisitos ejecutables
  - Desarrollo basado en pruebas de aceptación (ATDD)
  - Desarrollo impulsado por el comportamiento (BDD)
  - Pruebas de aceptación ágiles
  - Requisitos basados en pruebas (TDR)



Especificación por ejemplos (SbE)

- Cuando se aplica la técnica de Especificación por ejemplos (SbE) [Fowler 2004] (como ATDD y BDD), a menudo se utiliza un paquete de fichas de cuatro colores para mapear y capturar los diferentes tipos de información a medida que se desarrolla la conversación.
- La información se captura en fichas y se organiza en un mapa.
  - Para empezar, la característica o historia en discusión se escribe en una tarjeta azul y se coloca en la parte superior del mapa.
  - A continuación, cada uno de los criterios de aceptación, o reglas que ya se conocen, se escriben en una tarjeta amarilla y se colocan en el mapa debajo de la tarjeta de historia azul.
  - Para cada regla, se necesitan uno o más ejemplos para ilustrar la regla. Estos están escritos en una tarjeta verde y colocados bajo la regla correspondiente. A menudo se utiliza la sintaxis de Gherkin (Given-When-Then) que es un lenguaje específico de dominio (DSL) creado específicamente para descripciones de comportamiento (BDD).
  - A medida que se discuten estos ejemplos, es posible que se descubran preguntas que ninguno de los participantes pueda responder. Estas preguntas se plasman en una tarjeta roja y los participantes continúan con la conversación.



Técnicas de Estimación

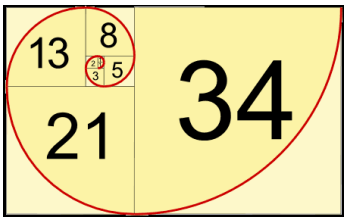
- La estimación del coste de los sistemas de información es una de las cosas más difíciles y propensas a error que te puedes encontrar. Las técnicas de estimación te ayudan en esta tarea, obtienes el número de horas necesarias para desarrollar el proyecto y así puedes calcular su coste.
- Una parte de los factores son conocidos o esperados y determinan la complejidad del proyecto, pero otra parte de los factores son desconocidos o no esperados y se pueden producir en cualquier momento, determinando la incertidumbre. La relación entre la parte conocida y la incierta determina el grado de dificultad. El estudio de estos factores de incertidumbre se denomina “Análisis de riesgos”.
- Una buena estimación es aquella que las partes interesadas consideran suficientemente fiable para utilizarla en la toma de decisiones. Y una buena estimación incluye todos los aspectos relevantes, involucrando (o representando) a todas las personas relevantes. Una estimación indica un rango en lugar de un número. Es un cálculo aproximado de cuál será probablemente la respuesta.
- Las técnicas de estimación acaban basándose en una valoración que tú debes realizar, que será mejor o peor en función a tu experiencia, es decir, es un método empírico. Una estimación nunca es exacta, estudiando por qué no se han cumplido las estimaciones aprendes de tus errores. La próxima vez que realices una estimación, intentarás no cometer los mismos errores, de tal forma que después de varios proyectos tus estimaciones serán bastante mas exactas.

Estimando con Story Points

- Una característica de la mayoría de técnicas de estimación agiles es que no se estima en horas exactas, sino en puntos. La estimación de puntos proporciona una estimación relativa de una actividad del pasado.
- Un Story Point es una unidad para dimensionar historias. Es una unidad arbitraria del ESFUERZO necesario para realizarla. Depende del esfuerzo, de la complejidad y de la incertidumbre asociada a la Historia de Usuario.
- Cada equipo tiene su propia "traducción" del esfuerzo necesario para realizar un Story Point
  - Para algunos equipos significa "1 día"
  - Para algunos equipos significa "1 semana"
  - Para algunos equipos significa "1 día ideal" (días laborables sin interrupciones, distracciones ni contratiempos)
  - Para algunos equipos significa "4 horas"
- Los puntos de historia representan una medida relativa con respecto a la cantidad de trabajo que se requiere para realizar los elementos de la cartera de productos. La estimación no se hace en términos de tiempo, sino en complejidad, esfuerzo y riesgo entre sí.
- La velocidad es la cantidad promedio de trabajo (a menudo medida en puntos de historia) que el equipo puede realizar en un sprint.

Secuencia de Fibonacci y otros métodos

- Nos interesa un "orden de magnitud" del esfuerzo, así que podemos utilizar diferentes enfoques:
  - T-Shirts: XL, L, M, S, XS (valores relativos)
  - Fibonacci: 1, 2, 3, 5, 8, 13, 21, 34,… (aumentar el tamaño aumenta el margen de error)
  - Pseudo Fibonacci (variación mas común): 0, ½, 1, 2, 3, 5, 8, 13, 20, 40, 100



Planning Poker

- El póquer de planificación es una forma efectiva de asignar puntos de historia relativos a elementos (por ejemplo, historias de usuarios, características, picos, etc.) en una reunión con todos los miembros del equipo usando tarjetas que tienen valores específicos para representar una cantidad de puntos de la historia.
- Basado en el hecho de que al aumentar el tamaño de las tareas aumenta también el margen de error, consiste en emplear números de la sucesión de Fibonacci para realizar las estimaciones, de forma que:
  - El juego de cartas está compuesto por números en sucesión de Fibonacci.
  - La estimación no se realiza levantando varias cartas para componer la cifra exacta, sino poniendo boca arriba la carta con la cifra más aproximada a la estimación.

Planning Poker

Reglas del juego

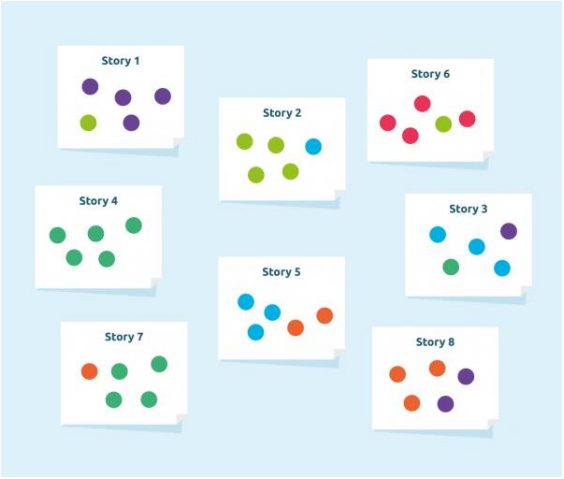
1. Cada miembro del equipo recibe un juego de cartas (ni el scrum master ni el propietario del producto participan, a menos que también tengan un rol diferente en el equipo).
2. Una persona, preferiblemente el propietario del producto, leerá en voz alta el artículo a estimar.
3. El equipo discute brevemente el tema.
4. Cada miembro del equipo decidirá por sí mismo qué tarjeta representa la cantidad correcta de trabajo (teniendo en cuenta la complejidad, el riesgo y el esfuerzo). Si un miembro del equipo no puede elegir, seleccione la tarjeta 0. Si el miembro del equipo cree que la cantidad de trabajo es enorme, seleccione la tarjeta +.
5. Una vez que todos hayan elegido, las cartas se colocarán sobre la mesa simultáneamente.
6. Si todas las estimaciones corresponden, la estimación puntual de la historia para ese elemento se considera realizada.
7. Si las estimaciones de los puntos de la historia difieren, las diferencias se discutirán dentro del grupo (la discusión se centrará en los valores mínimos y máximos que se desvían).
8. Repetir hasta que se haya logrado un consenso y se pase al siguiente punto.

Tamaño relativo (T-Shirts)

Estimación de afinidad

Votación por puntos

- Esta técnica permite a los equipos expresar el esfuerzo de forma rápida y visual, lo que la convierte en una forma eficaz de completar numerosos elementos en un tiempo limitado.
- Para utilizar la votación por puntos:
  - Configure sus historias de usuario visualmente y asigne un color a cada miembro del equipo.
  - Como grupo, decidan en una escala (digamos del uno al cinco) que represente el esfuerzo necesario para completar cada elemento.
  - Pide a todos que agreguen puntos a las historias según la escala elegida.



Otras técnicas

- Estimación basada en ratios
  - Para utilizar proporciones como base para crear una estimación de las tareas de la historia del usuario, es importante recopilar la mayor cantidad posible de cifras de experiencia para derivar proporciones estándar (Diseño:Construcción:Prueba) para historias de usuarios similares.
- Extrapolación
  - Esta técnica se utiliza para empezar a acumular experiencia desde el primer sprint. Con base en la evolución de estos números a lo largo del tiempo, es posible hacer una estimación (por extrapolación) para sprints futuros. Cuantas más cifras se conozcan, más precisa será la estimación.
- Puntos Función
  - Esta técnica permite la estimación descomponiendo aproximadamente el sistema en entregables a partir de un mínimo conocimiento de las funcionalidades y entidades que intervienen. Para proceder al cálculo de los puntos función de un sistema han de realizarse tres etapas:
    - Identificación de los componentes o entregables necesarios para el cálculo.
    - Cálculo de los Puntos Función no ajustados.
    - Ajuste de los Puntos Función.

GESTIÓN DE RIESGOS

Concepto

- Un **riesgo** es la posibilidad de daño o pérdida (Webster's Dictionary), contingencia o proximidad de un daño (RAE) o, coloquialmente, un problema que aparecerá en un futuro. Mucha gente siente que un riesgo es lo contrario de un beneficio o viceversa, aunque no es estrictamente cierto: sin riesgo no hay beneficio (No pain no gain).
- El **beneficio** es un valor comercial positivo que se puede derivar de la implementación de una funcionalidad. El **daño** es un valor comercial negativo que puede ser el resultado de no haber implementado una funcionalidad (valor 0) o por un evento que ocurre por haberla implementado. El **riesgo** es el producto del daño por la posibilidad de que el evento realmente ocurra. La implementación de una funcionalidad tiene un coste. La mayoría de las veces el beneficio debe ser mayor que el coste aunque el coste puede ser mayor que el beneficio si es menor que el daño potencial.
- Los riesgos existen y son inherentes a todos los proyectos: No son una certeza, sino algo que puede ocurrir con determinada probabilidad.
- El riesgo no es ni bueno ni malo en si mismo y por tanto no hay que evitar su reconocimiento: un riesgo identificado es una oportunidad de mejora. No hay que temer al riesgo, sino gestionarlo, minimizando la incertidumbre y definiendo las acciones necesarias para mitigarlo.
- Un riesgo cuya causa ocurre es una contingencia (o incidencia), dejando de ser un riesgo.

Análisis de riesgos del producto (PRA)

- Un proyecto nunca tiene tiempo y dinero ilimitados para su desarrollo. Estas limitaciones en términos de tiempo y dinero imponen restricciones sobre la cantidad de funcionalidad y/o calidad que se debe lograr dentro de un proyecto. Para las pruebas, esto se traduce en tomar decisiones con respecto a qué probar, cómo probarlo y con qué profundidad hacerlo.
- El PRA es una herramienta importante para ayudar al evaluador a dividir los recursos limitados. Esto comienza con los riesgos involucrados con el proyecto o cambio específico. Nos gustaría realizar más pruebas en los lugares donde el riesgo es mayor. La PRA ayuda a identificar y estimar esos riesgos.
- Para tomar decisiones de alta calidad en esto, el evaluador se beneficia enormemente de información proveniente de varios ángulos: qué partes del sistema ofrecen el mayor beneficio (o, si falla, causan los costos de daño más altos), qué característica o característica se considera más importante, ¿Qué partes son las más invocadas, en el tiempo o por los clientes?
- Un PRA puede considerar un sistema bajo prueba desde una variedad de puntos de vista, por ejemplo, desde un punto de vista comercial: cómo encaja el sistema en los procesos comerciales, qué características son las más importantes, pero también desde un punto de vista técnico/de TI: si se utiliza tecnología probada, ¿Qué tan estable es una base de código fuente instalada,…?
- La PRA tiene como objetivo identificar partes del sistema bajo prueba de alto riesgo, alto rendimiento o muy utilizadas para que los recursos de prueba puedan asignarse en consecuencia.

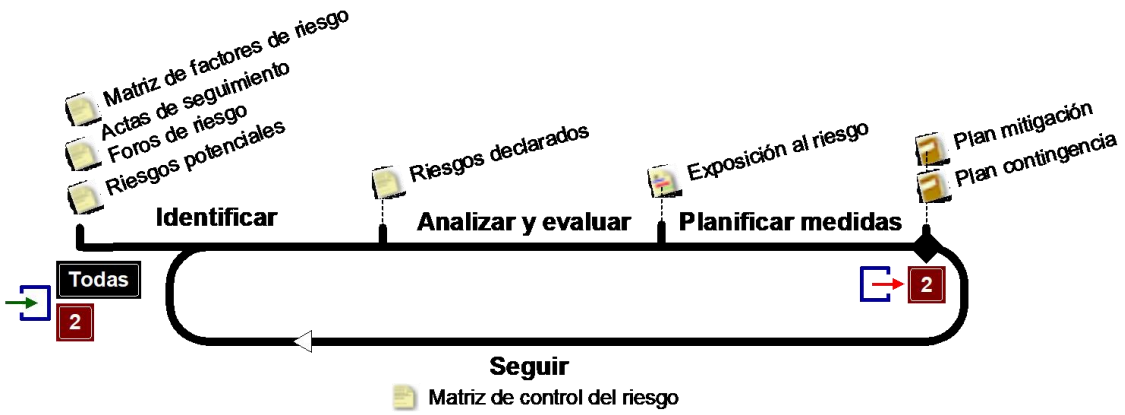
Principios del PRA

- Toma de decisiones basada en riesgos, definiendo las acciones necesarias para su mitigación.
- Formalizar el proceso de gestión del riesgo.
- Dar cobertura a todas las personas, productos y procesos involucrados.
- Considerar la identificación de riesgos como positiva, desarrollando un ambiente en el que las personas que identifiquen riesgos no sean pre-juzgadas.
- Gestionar los riesgos de forma pro-activa y reactiva.
- Evaluación continua.

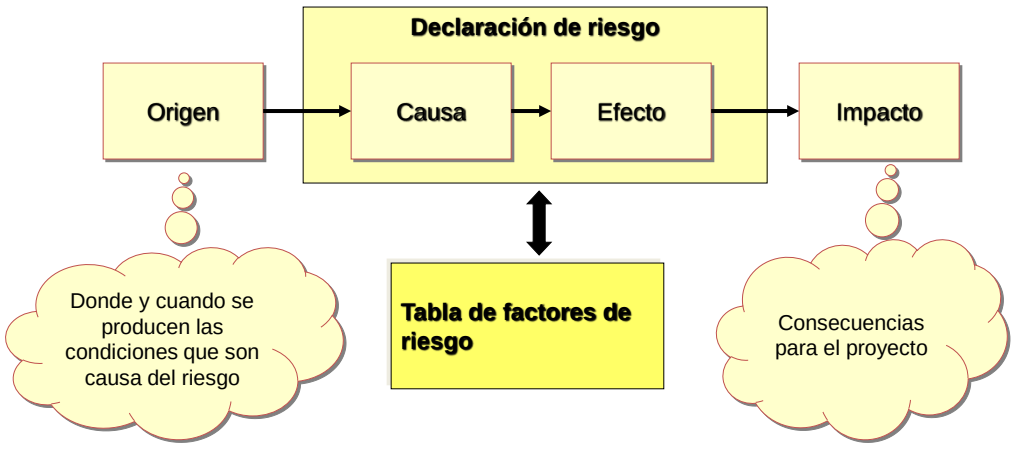
Pro-actividad vs Reactividad

- Pro-actividad = prevención
  - Prever riesgos de mayor impacto y probabilidad de ocurrencia, la exposición es el parámetro a controlar
    - Exposición=probabilidad*impacto.
  - Las actividades de prevención del riesgo son para mitigarlos y actúan sobre las causas del riesgo.
  - Las actividades pro-activas conforman el “plan de mitigación”, o “plan de prevención”, y se deben ejecutar para prevenir el riesgo.
- Reactividad = reacción ante contingencias
  - Cuando el efecto se ha producido se produce una contingencia (incidencia).
  - Las actividades reactivas son para actuar contra los efectos del riesgo.
  - Las actividades reactivas tienden a minimizar el impacto del riesgo.
  - Las actividades reactivas conforman el “plan de contingencias” y se deben ejecutar cuando se produce el efecto del riesgo.

Proceso de identificación y gestión de riesgos



Identificación



Orígenes y Localización

- Orígenes (fuentes de riesgo).
  - Las personas enroladas en un proyecto.
  - La tecnología, proveedores y consultores IT.
  - Clientes y usuarios.
  - La dirección de todos los anteriores.
- Localizaciones:
  - Momento (fase) en la que aparece el riesgo.
  - Procedimiento en el que se origina.
  - Línea de producto donde se origina (infraestructura, aplicación, implantación).
- Al detectar los orígenes se establecen oportunidades de mejora, sin embargo esto suele trascender al proyecto.

Riesgos más comunes en el desarrollo de software

- Cuestiones Asociadas al Producto:
  - El software podría no realizar las funciones previstas de acuerdo con la especificación.
  - El software puede no realizar las funciones previstas de acuerdo con las necesidades de los usuarios, clientes y/o implicados.
  - Una arquitectura de sistema puede no soportar de forma adecuada algunos requisitos no funcionales.
  - Un cómputo específico puede realizarse de forma incorrecta en algunas circunstancias.
  - Una estructura de control de un bucle puede estar codificada de forma incorrecta.
  - Los tiempos de respuesta pueden ser inadecuados para un sistema de procesamiento de transacciones de alto rendimiento.
  - La retroalimentación de la experiencia de usuario (UX67 por sus siglas en inglés) podría no cumplir con las expectativas respecto del producto.
- Cuestiones Asociadas al Proyecto:
  - Se pueden producir retrasos en la entrega, la finalización de tareas, el cumplimiento de los criterios de salida o la definición de hecho.
  - Las estimaciones inexactas, la reasignación de fondos a proyectos de mayor prioridad o el recorte general de gastos en toda la organización pueden dar lugar a una financiación inadecuada.
  - Los cambios tardíos pueden resultar en cambios sustanciales de reelaboración.
- Cuestiones Asociadas a la Organización:
  - Las competencias, la formación y el personal pueden no ser suficientes.
  - Las cuestiones relacionadas con el personal pueden causar conflictos y problemas.
  - Los usuarios, el personal de la empresa o los expertos en la materia pueden no estar disponibles debido a prioridades de negocio en conflicto.

Riesgos más comunes en el desarrollo de software

- Cuestiones de Carácter Político:
  - Los probadores pueden no comunicar adecuadamente sus necesidades y/o los resultados de la prueba.
  - Los desarrolladores y/o probadores pueden no realizar un seguimiento de la información obtenida en la prueba y las revisiones (por ejemplo, no mejorar las prácticas de desarrollo y prueba).
  - Puede haber una actitud y expectativas inadecuadas con respecto a la prueba (por ejemplo, no apreciar el valor de encontrar defectos durante la prueba).
- Cuestiones de Carácter Técnico:
  - Es posible que los requisitos no estén suficientemente bien definidos.
  - Es posible que no se cumplan los requisitos, dadas las restricciones existentes.
  - Es posible que el entorno de prueba no esté listo a tiempo.
  - La conversión de datos, la planificación de la migración y el soporte de herramientas se pueden retrasar.
  - Las debilidades en el proceso de desarrollo pueden afectar la consistencia o la calidad de los productos de trabajo del proyecto, como el diseño, el código, la configuración, los datos de prueba y los casos de prueba.
  - Una mala gestión de los defectos y problemas similares pueden dar lugar a defectos acumulados y otra deuda técnica69.
- Cuestiones Relacionados con los Proveedores:
  - Un tercero puede no entregar un producto o servicio necesario, o declararse en quiebra.
  - Las cuestiones contractuales pueden causar problemas al proyecto.

Impacto (parámetros expuestos al riesgo).

- Económicos:
  - Plazo.
  - Coste.
- Capacidades:
  - Funcional.
  - Técnica (parámetros de calidad de la arquitectura: rendimiento y escalabilidad, seguridad, disponibilidad, facilidad para el desarrollo, explotabilidad,...).
- Calidad:
  - Conformidad a requerimientos y capacidad para el buen uso del producto a obtener.
  - Nivel de defectos (en producto y en proceso).

Declaración de riesgo

- El riesgo se declara como una regla: con antecedentes y consecuencias.
  - Los antecedentes son la causa del riesgo.
  - La causa es el conjunto de condiciones que se deben dar para que se produzca el efecto.
  - El efecto tiene un impacto sobre el proyecto en forma de consecuencias.
  - Declaración: si se dan tal y cual condiciones se producirán este y aquel efecto "negativo" en el proyecto.
- Ejemplos.
  - si se producen enfermedades o no se planifican las vacaciones se producirán ausencias no planificadas (el impacto será la aparición de retrasos en la entrega de productos).
  - si no se participa en las discusiones y revisiones, se pierde la oportunidad de aportar valor al proyecto (el impacto será una reducción en la calidad).
  - si la plataforma tecnológica no es estable se producirán incidencias durante el desarrollo (el impacto será más retrasos en la construcción del producto y un mayor esfuerzo para resolverlas).

Análisis y evaluación.

- Discutir los riesgos identificados.
- Hallar probabilidad inter-subjetiva de que se dé el riesgo (de que aparezcan las condiciones o causas y se produzca el efecto).
  - **Probabilidad** = Posibles defectos * Frecuencia de uso
- Analizar impacto: magnitud del daño o pérdida.
  - Estimar coste de la pérdida, el desplazamiento de fechas, los defectos potencialmente introducidos, ... (opción difícil y costosa de llevar a cabo).
  - Establecer un nivel de impacto acordado (rango de valores con valores bajos para el mínimo impacto y altos para el máximo), p.ej: entre 1 y 5 (opción simple aunque introduzca factores subjetivos, para minimizar la subjetividad se realizan evaluaciones en grupos).
- Calcular **exposición** (probabilidad*impacto): es un indicador del nivel de amenaza del riesgo para el proyecto.

Planificación de control de riesgos

- Focalizar en reducir la exposición (probabilidad*impacto) al riesgo.
- Definir acciones de prevención o pro-activas. Ejemplos (para los ejemplos de declaración del riesgo):
  - Dotar inicialmente al proyecto de personas extra para mitigar el riesgo de ausencias; planificar las ausencias.
  - Utilizar herramientas tipo Foro para fomentar las discusiones sin necesidad de presencia física.
  - Probar la plataforma tecnológica previamente y si es inestable utilizar otra.
- Definir acciones para la contingencia. Por ejemplo: definir alertas para declarar el riesgo como incidencia
  - Contar con un “pool” de recursos disponibles para el caso de que se produzcan ausencias.
  - Definir una persona con autoridad para establecer y tomar las decisiones cuando no hay participación. Acudir a personal externo. Cambiar los participantes.
  - Contratar los servicios del proveedor de tecnología. Portar lo realizado a una nueva plataforma.
- Asignar responsabilidades.
- Acotar en tiempo y necesidades.

Seguimiento de riesgos

- Registrar riesgos en la hoja de control de riesgos.
  - Declaración.
  - Probabilidad.
  - Impacto.
  - Exposición.
  - Acciones preventivas: plan de mitigación.
  - Acciones correctivas: plan de contingencias.
  - Responsables.
  - Horizonte temporal.
  - Alarmas y disparadores.
- Revisar riesgos en las reuniones de seguimiento.
- Reevaluar prioridades continuamente.
- Seguir alertas.
- Replanificar programas de mitigación y contingencias.

Riesgo de calidad

- ¿Dónde debería centrar el equipo sus actividades de control de calidad y pruebas? ¿Cuáles deberían ser las prioridades de sus tareas? Para responder a estas preguntas, el equipo necesita investigar los riesgos de calidad involucrados con el sistema de TI que están creando o cambiando. Cuando hay un riesgo alto, se pondrá más esfuerzo en control de calidad y pruebas, cuando hay un riesgo bajo, se dedicará menos esfuerzo.
- Un riesgo de calidad es una posibilidad específica de que el producto falle en relación con el impacto esperado si esto ocurre. La probabilidad de fallo está determinada por los posibles fallos y la frecuencia de uso. El impacto está relacionado con el uso operativo del producto.
- El producto a ensayar se analiza mediante un análisis de riesgos de calidad con el objetivo de lograr una visión conjunta, para todos los interesados, de las características y partes de riesgo del producto a ensayar. Esta visión conjunta sobre los riesgos identificados sirve de base para determinar la estrategia de prueba.
- La estrategia de prueba es la asignación de medidas de calidad para equilibrar la inversión en pruebas y hacer una distribución óptima del esfuerzo entre las variedades de prueba y los enfoques de prueba para brindar información sobre la cobertura y la intensidad de las pruebas. A menudo esto se basa en los niveles de riesgo de calidad y el valor comercial buscado. Existen muchos enfoques para realizar un análisis de riesgos de calidad y determinar una estrategia de prueba.

Análisis de riesgos y estrategia de prueba

- Los pasos de análisis de riesgos de calidad y la estrategia de prueba son:
  - Reunir a los miembros del equipo.
  - Enumerar todos los elementos del trabajo pendiente (por ejemplo, historias de usuarios, características) del próximo sprint.
  - Identificar las características de calidad relevantes por artículo.
  - Analizar, utilizando el póquer de riesgo, el riesgo, que es una combinación del impacto si ocurre un fallo y la probabilidad de que ocurra, para cada combinación de artículo y característica de calidad (esto se llama riesgo de artículo).
- La aplicación de estos primeros cuatro pasos da como resultado una **tabla de riesgos** para:
  - Determinar la intensidad de la prueba por combinación de artículo y característica de calidad.
  - Asignar medidas de calidad por artículo a cada combinación de intensidad de prueba y característica de calidad.
- Para determinar la estrategia de prueba, los resultados del quinto y sexto paso se agregan a la **tabla de riesgos**. El resultado se denomina **tabla de estrategias de prueba**.

Características de calidad del producto (ISO 25010)

- Idoneidad funcional: el grado en que un producto o sistema proporciona funciones que satisfacen las necesidades declaradas e implícitas cuando se utiliza en condiciones específicas.
  - Integridad funcional, Corrección funcional, Adecuación funcional
- Eficiencia del desempeño: el desempeño relativo a la cantidad de recursos utilizados en las condiciones establecidas.
  - Comportamiento del tiempo, Utilización de recursos, Capacidad
- Compatibilidad: el grado en que un producto, sistema o componente puede intercambiar información con otros productos, sistemas o componentes y/o realizar las funciones requeridas, mientras comparte el mismo entorno de hardware o software.
  - Interoperabilidad de coexistencia
- Seguridad: el grado en que un producto o sistema protege la información y los datos para que las personas u otros productos o sistemas tengan el grado de acceso a los datos apropiado para sus tipos y niveles de autorización.
  - Confidencialidad, Integridad, No repudio, Responsabilidad, Autenticidad

Características de calidad del producto (ISO 25010)

- Usabilidad: el grado en que un producto o sistema puede ser utilizado por usuarios específicos para lograr objetivos específicos con efectividad, eficiencia y satisfacción en un contexto de uso específico.
  - Idoneidad reconocibilidad, Capacidad de aprendizaje, Operabilidad, Protección contra errores del usuario, Estética de la interfaz de usuario, Accesibilidad
- Confiabilidad: el grado en que un sistema, producto o componente realiza funciones específicas en condiciones específicas durante un período de tiempo específico.
  - Madurez, Disponibilidad, Tolerancia a fallos, Recuperabilidad
- Mantenibilidad: el grado de efectividad y eficiencia con el que los mantenedores previstos pueden modificar un producto o sistema.
  - Modularidad, Reutilizabilidad, Analizabilidad, Modificabilidad, Probabilidad
- Portabilidad: el grado de efectividad y eficiencia con el que un sistema, producto o componente se puede transferir de un hardware, software u otro entorno operativo o de uso a otro.
  - Adaptabilidad, Instalabilidad, Reemplazabilidad

Características de calidad del uso (ISO 25010)

- Eficacia: la precisión y exhaustividad con la que los usuarios logran objetivos específicos.
- Eficiencia: los recursos gastados en relación con la precisión y la integridad con la que los usuarios logran sus objetivos.
- Satisfacción: el grado en que se satisfacen las necesidades del usuario cuando un producto o sistema se utiliza en un contexto de uso específico.
  - Utilidad, Confianza, Placer, Confort
- Ausencia de riesgos: el grado en que un producto o sistema mitiga el riesgo potencial para la situación económica, la vida humana, la salud o el medio ambiente.
  - Mitigación de riesgos económicos, Mitigación de riesgos de salud y seguridad, Mitigación de riesgos ambientales
- Cobertura de contexto: el grado en que un producto o sistema puede usarse con eficacia, eficiencia, ausencia de riesgos y satisfacción tanto en contextos de uso específicos como en contextos más allá de los inicialmente identificados explícitamente.
  - Integridad del contexto, Flexibilidad

Risk Poker

- A menudo se encuentran diferencias importantes con respecto a los puntos de la historia asignados por los diferentes participantes. Esto se debe principalmente a las diferentes opiniones de los jugadores de póquer sobre los esfuerzos de prueba requeridos.
- Una solución para lograr una mejor comprensión de estos esfuerzos de prueba, así como para determinar una estimación del tamaño más inequívoca de una historia de usuario, es jugar al póquer de riesgo (evaluar el riesgo de la historia).
- El póquer de riesgo (Risk Poker) es una extensión del póquer de planificación y se utiliza a menudo en enfoques ágiles para preparar una estimación del riesgo relativa al tamaño de los elementos del backlog (historias de usuarios).
- El póquer de riesgo se realiza justo antes del póquer de planificación, el riesgo influye en la planificación, y requiere un enfoque de todo el equipo, lo que significa que todos los miembros obtendrán información y lograrán una mejor comprensión de todos los pros y pros de las diversas tareas de prueba.

Reglas del juego

1. Cada miembro del equipo recibe/selecciona las tarjetas 1, 2 y 3.
2. Una persona, preferiblemente el scrum master que no juega, lee el elemento y la característica en voz alta y solicita estimaciones de **probabilidad de fallo**.
3. Cada miembro del equipo determina por sí mismo qué carta cree que representa la probabilidad correcta de una estimación de error y la coloca boca abajo sobre la mesa.
4. Una vez que todos hayan hecho una selección, el scrum master les pedirá a todos que revelen sus cartas simultáneamente.
5. Si todos han seleccionado el mismo número de puntos de probabilidad de fallo, la estimación de probabilidad de fallo para este elemento se puede considerar completa.
6. Si los puntos de probabilidad difiere, el equipo discutirá estas diferencias (centrándose en los valores desviados) y se repetirán los pasos 3 a 6 hasta lograr un consenso, después de lo cual el scrum master anotará los puntos de probabilidad de fallo.
7. A continuación se solicita una estimación del **impacto del fallo** y se repiten los pasos 3 a 6.
8. El resultado son los puntos de riesgo del elemento, los puntos de la probabilidad por los del impacto, se anotan en la tarjeta de historia y se pasa al siguiente elemento.

Tabla de riesgos

- En el póquer de riesgo, el objetivo no es obtener una estimación perfectamente precisa aunque asignar números de riesgo puede dar una falsa sensación de precisión. Por eso muchos equipos trabajan con letras: por ejemplo, una A si el número de riesgo es 9 (alto), una B si es 6 o 4 (medio) y una C si es 1, 2 o 3 (bajo).

Item	Characteristic	Impact	Chance of failure	Risk Class	
US 1	Functionality	3	3	9	A
	Usability	2	1	2	C
US 2	Functionality	2	2	4	B
	Security	3	2	6	B
US 3	Functionality	2	1	2	C
Spike 1	Performance	2	1	2	C
Feature 1	Performance	1	1	1	C
Feature 2	Functionality	2	2	4	B
	Suitability	2	2	4	B
...	...	...	...	...	...

Estrategia de calidad

- La estrategia de calidad describe la asignación de medidas de calidad a los elementos de entrega de TI (por ejemplo, historias de usuario, características, etc.), para equilibrar la inversión en actividades de ingeniería de calidad y realizar una distribución óptima del esfuerzo sobre las actividades fundamentales y las variedades de prueba. La clase de riesgo de calidad se utiliza para asignar la intensidad de las medidas de calidad que deben aplicarse.
- El beneficio de aplicar una estrategia de ingeniería de calidad es que un equipo (o grupo de equipos colaboradores) crea una guía fácil y clara para todos los involucrados sobre qué actividades aplicar para lograr una calidad integrada en sistemas de TI adecuados para su propósito. Esta guía tendrá un enfoque diferente, algunas medidas de calidad están vinculadas a historias de usuarios individuales, otras medidas de calidad se aplican a características o epopeyas y la estrategia de ingeniería de calidad también contendrá medidas de calidad genéricas que el equipo aplica siempre.

Tabla de estrategias de prueba

- Los símbolos •••, •• y • se utilizan para especificar la intensidad de la prueba. Cuanto más •••, más intensa será la prueba: tres ••• para la A, dos •• para la B y uno • para la C. Se permite desviarse de esta regla, cuando exista una buena razón. Se puede agregar columnas por tipos de prueba.

Item	Characteristic	Risk Class	Static Testing	Dynamic Testing	Other Quality Measures
US 1	Functionality	A	••	•••	•••
	Usability	C	•	•	
US 2	Functionality	B	•	••	•
	Security	B	•	••	•
US 3	Functionality	C	•	•	
Spike 1	Performance	C	•	•	
Feature 1	Performance	C	•	•	
Feature 2	Functionality	B	•	••	•
	Suitability	B	•	••	•
...	...	...	...	...	...