Ingenierías de requisitos:

 Entender lo que el cliente quiere.

 Analizar las necesidades.

 Evaluar la factibilidad.

 Negociar una solución razonable.

 Especificar la solución sin ambigüedades.

 Validar la especificación.

 Administrar los requisitos conforme éstos se transforman en un sistema operacional.

Ciclo de vida del software

Fases y Objetivos:

1. Análisis o Planeación o definición de requisitos:

Se formulan los requerimientos del cliente. Se intenta descubrir lo que se necesita,

Las características que el sistema debe poseer.

| **Fase** | **Actividades** | **Artefactos** |
| --- | --- | --- |
| Análisis (definición de requisitos). |  |  |
| Definición del alcance del proyecto. |  |
| Identificación del negocio. | Modelo del negocio. |
| Toma de requerimientos. | Análisis y realización de casos de uso. |
| Estudio de procesos de negocio. | Modelo de procesos y actividades de negocio. |
| Calendarización del proyecto. | Cronograma del proyecto. |

1. Fase especificación o diseño:

Se identifica las soluciones del problema, y se deben asignar los recursos.

1. Validación de requisitos o Pruebas:

Verificar que los requisito están acorde

1. Fase de Mantenimiento:

Mantenimiento preventivo o correctivo.

ISO (Organización Internacional de Normalizacion)

IEE E(Instito de de Ingenienieros Electricos y Electronicos). Asociacion mundial de ingenieros dedicada a la normalización y desarrollo en areas recnicas)

Ciclo de vida de software según ISO 12207, es el marco común donde nos indica la metodología de cómo debemos implementar los procesos del ciclo de vida del software.

El IEE es el estándar que regula que los software realizados estén acorde al stadard establecido,

Tiene que ver con los requerimiento del ley como minimo que debecumplir según la calidad del software.

IEEE 1074 : el conjunto de actividades que constituyen  
los procesos que son obligatorios para el desarrollo y mantenimiento de software. Se  
encuentra organizado en 17 procesos, que comprenden un total de 65 actividades. Los  
procesos se dividen en cuatro secciones lógicas o grupos de procesos:

El primer grupo está compuesto por el Proceso de Modelo del Ciclo de Vida del  
Software que proporciona actividades que se necesitan para identificar los modelos de  
ciclo de vida software candidatos y para seleccionar aquel modelo que se vaya a  
utilizar en el proyecto.  
El segundo grupo está conformado por el Proceso de Gestión del Proyecto, que  
propone un conjunto de procesos de iniciación, supervisión y control del proyecto a lo  
largo de ciclo de vida del software.547547547

El tercer grupo está compuesto por los procesos Orientados al Desarrollo, los  
Procesos de Pre-Desarrollo, los Procesos de Desarrollo y los Procesos de Post-  
Desarrollo del software.  
El último grupo está compuesto por los Procesos Integrales, son aquellos procesos  
que se necesitan para completar con éxito las actividades de un proyecto

El proceso de implantación está tratado en el proceso de instalación del grupo de  
procesos post-desarrollo perteneciente al grupo de Procesos orientados al Desarrollo.  
Este proceso implica el transporte y la instalación de un sistema software desde el  
entorno de desarrollo al entorno de destino [1].  
Las actividades del proceso de instalación propuestas en el estándar son: la  
distribución del software, la instalación del software, la carga de la base de datos (si el  
proyecto lo requiere), la aceptación del software en el entorno de operación, la  
realización de las actualizaciones y finalmente la instalación del software probado.  
Los Procesos Integrales que articulan con el proceso de implantación son los  
procesos de verificación y validación, de gestión de configuración, de desarrollo de la  
documentación y de formación.  
Los Procesos de Gestión del Proyecto relacionados al proceso de implantación son:  
el proceso de iniciación del proyecto, el proceso de supervisión y control del proyecto  
y el proceso de gestión de la calidad

Fases: conjunto de actividades que están relacionadas entre si y que forman parte de un proceso de software y que tiene como objetivo en desarrollo del proceso que se va a realizar.

Entregables: documentos,

Normas 1074 IEEE,

Modelos o paradigmas del ciclo de vida de software.

Con la finalidad de proporcionar una metodología común entre el cliente y la empresa de software, se utilizan los modelos de ciclos de vida o paradigmas de desarrollo de software para plasmar las etapas y la documentación necesaria, de manera que cada fase se valide antes de continuar con la siguiente.

Paradigma tradicional: Cascada, espiral, prototipos

Paradigma orientado a objetos:

Paradigma de desarrollo ágil: scrum, kamba, XP

Hay cuatro (4) etapas en un proceso usual de ingeniería de requisitos y que son utilizadas para el desarrollo de un producto único, a saber: elicitación, análisis, especificación y validación de los requisitos.

1. Eliitacion: descubrir el problema que el sistema debe resolver

 Conocer el dominio del problema, de forma tal que los analistas puedan entenderse con los clientes y usuarios y sean capaces de transmitir dicho conocimiento al resto del equipo.

 Descubrir necesidades reales entre clientes y usuarios, haciendo énfasis en aquellas que la mayor parte de las veces se asumen y toman por implícitas.

 Consensuar los requisitos entre los propios clientes y usuarios hasta obtener una visión común de los mismos.

1. **Analisis:**

Detectar conflicto en los requisitos que suelen provenir de distintas fuentes y presentar contradicciones o ambigüedades debido a su naturaleza informal.

Profundizar en el conocimiento del dominio del problema puede facilitar el proceso de construir un producto útil para clientes y usuarios (Durán, 2000).

1. Especificacion:

En la práctica, esta etapa se realiza conjuntamente con el análisis, por lo que se puede decir que la especificación es el “pasar en limpio” el análisis realizado previamente aplicando técnicas y/o estándares de documentación, como la notación UML (Lenguaje de Modelado Unificado), que es un estándar para el modelado orientado a objetos, por lo que los casos de uso y la obtención de requisitos basada en los casos de uso se utilizan cada vez más para la obtención de requisitos.

1. Validacion

garantiza que los requisitos, una vez analizados y resueltos los posibles conflictos, correspondan realmente a las necesidades de clientes y usuarios

En esta fase, el analista proporciona un sistema de retroalimentación que refina el entendimiento conseguido en la etapa de obtención.

**Priorización de requisitos:**

**La priorización de requisitos implica la identificación de los requisitos más importantes para el proyecto de software y la asignación de un orden de prioridad a cada uno de ellos. Esto ayuda a los desarrolladores a enfocar sus esfuerzos en los requisitos más críticos para el éxito del proyecto y a asignar recursos adecuados para su implementación.**

**Técnicas de análisis de requisitos**

En el documento de especificación de requisitos se describe lo que sistemas debe hacer, pero no el cómo.

Existen 5 criterios principales que se tienen en cuenta para priorizar los requerimientos como son:

1. como el valor que los usuarios le dan a la visión del producto,
2. la urgencia,
3. las limitaciones de tiempo.
4. la técnica de complejidad
5. las preferencias de las partes interesadas.

Además, muy a menudo los proyectos deben ser priorizados adecuadamente tanto para los **objetivos principales** del proyecto como para las **tareas específicas** que lograrán los objetivos. Por lo tanto, nos ocupamos de la priorización en dos niveles: **nivel de producto** y **nivel de tarea**. Especialmente teniendo en cuenta que los clientes normalmente tienden a no entender que no pueden obtener todas las características que desean en la versión 1.0 de un nuevo producto de software.

**Analiza las áreas clave que se tienen en cuenta antes de tomar una decisión importante**

**Beneficio:** una ventaja que obtiene la empresa como resultado de la implementación requerida.  
**Penalización**: una consecuencia de no implementar un requisito.  
**Costes – esfuerzo** y recursos que se requieren para implementar un requisito.  
**Riesgo:** una probabilidad de que el requisito no entregue el valor esperado.  
**Dependencias:** una relación entre los requisitos, por ejemplo, cuando el requisito requiera completar otro requisito para su implementación.  
**Sensibilidad de tiempo:** fecha de caducidad, urgencia.  
**Estabilidad:** la probabilidad de que el requisito permanezca estático.  
**Cumplimiento de políticas:** requisitos que deben implementarse para cumplir con los requisitos reglamentarios.

Técnicas para priorizar requisitos:

1. Clasificación de lista:

Cada requerimiento se prioriza con relación al resto de requerimiento, asignándole a cada requerimiento un valor único numérico del 1 hasta el ultimo requerimiento. De tal forma que no habrán 2 requerimiento con el mismo número de importancia.

1. **Tecnica punto de historia y valor de negocio:**

**La técnica de puntos de historia es una metodología utilizada en la estimación del esfuerzo y la planificación del desarrollo de software ágil. Esta técnica se basa en la identificación y medición de los requisitos del usuario o historias de usuario, donde se asignan puntos de historia a cada requisito en función de su complejidad y tamaño.**

**El valor del negocio se refiere al beneficio que el software proporciona al negocio y a los usuarios finales. Este valor puede ser medido en términos de retorno de inversión (ROI), aumento de la satisfacción del cliente, reducción de costos y aumento de la eficiencia, entre otros indicadores.**

**La relación entre la técnica de puntos de historia y el valor del negocio es importante en la gestión de proyectos de software. Al priorizar los requisitos de los usuarios, la técnica de puntos de historia ayuda a los equipos de desarrollo a enfocarse en los requisitos más importantes y a asignar los recursos adecuados para su implementación. Al mismo tiempo, los requisitos que aportan un mayor valor de negocio pueden ser identificados y priorizados para garantizar que el software entregado satisfaga las necesidades del negocio y de los usuarios finales.**

**Por lo tanto, la técnica de puntos de historia puede ayudar a los equipos de desarrollo a identificar y priorizar los requisitos que tienen el mayor valor de negocio, lo que permite una mejor asignación de recursos y una mayor eficiencia en la implementación del software. De esta manera, el valor del negocio se puede maximizar a través de la entrega de software de alta calidad que cumple con los requisitos y necesidades de los usuarios finales.**

¿Qué es una historia en Scrum?

Las historias de usuario son **unas pocas frases en lenguaje sencillo que describen el resultado deseado**. No entran en detalles, ya que los requisitos se añaden más tarde, una vez acordados por el equipo. Las historias encajan perfectamente en marcos ágiles como scrum y kanban.

1. **Mapa de dependencia**

Siempre es una buena idea crear un mapa de dependencia para comprender mejor las dependencias entre los requisitos. Especialmente teniendo en cuenta que la mayoría de los requisitos son interdependientes y difícilmente encontrarás un requisito que sea independientemente.

Entender los requisitos de dependencia es tan importante como la priorización. Sin entender la dependencia de los requisitos, es muy poco probable que llegue al orden correcto de la implementación de los requisitos. Por lo tanto, es una buena idea tener el mapa de dependencia de requisitos en tu lugar antes de priorizar los requisitos.

Por ejemplo tenemos 2 requerimiento con igual importancia, por ejemplo ambos tiene una importancia de 3:

Por ejemplo en un programa de inventario, tenemos 2 requerimientos que son:

Listado de stock mínimo,

Entrada de datos por teclado,

Ambos tienen la misma importancia 3, el cliente nos dice que sin el listado de stock minino, el no podría tomar decisiones a tiempo.

Y el otro es una entrada de datos por teclado:

Así que el listado depende de la información digitada, por lo tanto primero se debe hacer la entrada de datos y después el listado de stock.

1. **MoSCoW**

El método MoSCoW funciona mejor que el sistema de calificación numérica, ya que es mucho más fácil para las partes interesadas calificar los requisitos como Must, Should, Could or Would

El acrónimo representa lo siguiente:

Must – obligatorio  
Should – de alta prioridad  
Could – Preferido pero no necesario  
Would – puede ser pospuesto y sugerido para futura ejecución



Estimación de punto de historia.

Para que estimamos?

Para no comprometernos con mas actividades de las que podemos realizar. Y tratar de realizar la mayor cantidad de actividades posibles.

Tambien nos ayuda a identificar la velocidad de trabajo del equipo de desarrollo.

Para saber con cuantas historias de usuarios se puede comprometer un equipo en un sprint

para estimar hay que tener en cuenta el grado de dificultad,

se escoge un pivote, que es una historia de usuario que todos

la saben realizar.

y se le estima un tiempo, que vamos a utilizar como patrón de comparación.

después comparamos este patrón con cada una de las

demás historias, por ejemplo si es el doble de dificulta o

la mitad de dificultad comparándolo con el patrón (pivote)

scrum póker

Bibliografia: tomado de internet <https://apiumhub.com/es/tech-blog-barcelona/priorizacion-requerimientos-software/>

**Punto de Historia y Estimación**

Hacer estimaciones es complicado. Para los desarrolladores de software, es uno de los aspectos más difíciles de su trabajo, por no decir el más difícil. Conlleva tener en cuenta un montón de factores que ayudan a los propietarios de los productos a tomar decisiones que afectan a todo el equipo, así como a la empresa. Con todo eso en juego, no es de extrañar que todos, desde los desarrolladores hasta la alta dirección, tiendan a perder los estribos sobre este tema. Craso error. La estimación en la metodología ágil no es más que eso, un cálculo: no es un pacto de sangre.

No es obligatorio trabajar los fines de semana para compensar el tiempo de más que nos lleva un trabajo que habíamos subestimado. Dicho eso, veamos algunas maneras de realizar estimaciones con la mayor precisión posible.

## Colaboración con el propietario del producto

En un desarrollo ágil, el [propietario del producto](https://www.atlassian.com/es/agile/product-management) se encarga de priorizar el [backlog](https://www.atlassian.com/es/agile/scrum/backlogs), es decir, la lista ordenada de trabajo que contiene descripciones breves de todas las funciones y correcciones de un producto. Los propietarios del producto capturan los [requisitos](https://www.atlassian.com/es/agile/product-management/requirements) empresariales, pero no siempre entienden los detalles de la implementación. Por ello, una buena estimación puede informar al propietario del producto sobre el nivel de esfuerzo de cada elemento de trabajo, que a su vez sirve para evaluar la prioridad relativa de cada elemento.

Cuando el equipo de ingeniería empieza su proceso de estimación, normalmente surgen preguntas sobre los requisitos y las historias de usuario. Esto es algo positivo: las preguntas ayudan a todo el equipo a entender el trabajo mejor. Específicamente en el caso de los propietarios de los productos, la división granular de los elementos de trabajo y las estimaciones les ayudan a priorizar todas las áreas del trabajo, incluidas las que pueden estar ocultas. Con las estimaciones del equipo de desarrollo en la mano, no es extraño que un propietario del producto reordene los elementos del backlog.

## La estimación ágil es un trabajo en equipo

Involucrar a todo el mundo (desarrolladores, diseñadores, testers, deployers... todos) en el equipo es clave. Cada miembro del equipo aporta una perspectiva diferente sobre el producto y el trabajo necesario para entregar una historia de usuario. Por ejemplo, si la gestión de productos quiere hacer algo que parece sencillo, como admitir un nuevo navegador web, el desarrollo y el control de calidad deben dar su opinión también, ya que su experiencia les ha enseñado qué dragones pueden estar al acecho bajo la superficie.

Asimismo, los cambios de diseño requieren no sólo la aportación del equipo de diseño, sino también la del de desarrollo y la del de QA. Dejar a parte del equipo de producto más amplio fuera del proceso de estimación crea estimaciones de menor calidad, baja la moral porque los contribuyentes clave no se sienten incluidos y compromete la calidad del software.

No dejes que tu equipo sea víctima de las estimaciones poco precisas. Es un camino seguro al fracaso.

https://www.atlassian.com/es/agile/project-management/estimation

<https://concepto.de/teoria-de-sistemas/>

<https://www.processmaker.com/es/blog/process-mapping/>