Introducción a la ingeniería de requisitos

Tabla de contenido

Introducción

Una de las dificultades más grandes que enfrentamos en el desarrollo de software es decidir qué es lo que se va a desarrollar. El proceso de entender cuáles son las necesidades del usuario, cuáles son sus expectativas y definir qué es lo que debe hacer el producto para satisfacerlas, está plagado de infinidad de problemas: Falla en la comunicación, diferencias culturales, poder, política. Alan David señala que entre el 40 y el 50% de los defectos encontrados en el software tienen su origen durante estas actividades. Para resolver estos problemas dentro de la ingeniería de software contamos con lo que se conoce como ingeniería de requisitos.

[**V** / F] Dentro de lo más difícil en el desarrollo del software es decidir el qué se debe desarrollar.

En una entrevista con un usuario podríamos escuchar algo como esto: "Nuestros clientes deben poder retirar efectivo de cualquiera de los cajeros de nuestra red, hasta un monto de $5.000 diarios". A partir de esta simple declaración que aparentemente es clarísima, surgen un montón de interrogantes, por ejemplo: ¿Quiénes son los clientes? ¿Qué información tenemos de ellos? ¿Qué implica que haya una red propia? ¿El monto es fijo, podrá variar en el futuro? ¿De qué manera medimos los días? ¿Contamos 24 horas?

Para ayudarnos a entender qué es lo que los usuarios necesitan y transformar esto que acabamos de ver en una definición del comportamiento del sistema a construir, contamos justamente con la ingeniería de requisitos.

¿Qué es formalmente un **requisito**? Lo que un producto o sistema debe hacer para satisfacer las necesidades y expectativas de los usuarios, independientemente de cómo lo implementemos. Hay otras definiciones, por supuesto, cada uno de los autores que ha escrito acerca del tema ha dado la propia. Quizás la más completa, sean la que dan Sommervile y Sawyer:

Los requisitos son una especificación de lo que se debe implementar. Constituyen una descripción de cómo el sistema se deberá comportar o de una propiedad o atributo que deberá poseer.

Requerimientos y requisitos

Definiciones

Es importante aclarar que la palabra en inglés "requirement" se traduce muchas veces al español como "requerimiento". Pero lo correcto, sería utilizar la palabra "requisito". En castellano "requerir" implica "intimar con la autoridad pública o solicitar algo de alguien". "Requisito" parece ser la palabra más adecuada para traducir requerimiento, al menos en la mayoría de los casos, ya que "requisito" es "una condición necesaria para algo". De todas maneras, no hay que perder de vista que en este contexto requisito tiene un significado muy especial que es el que acabamos de explicar anteriormente y que no necesariamente sigue al pie de la letra a la definición del diccionario.

Requisitos

Distintos tipos de requisitos

Hay 2 grandes tipos de requisitos:

- Los **funcionales** que describen lo que necesitamos que el sistema haga, independientemente de cómo lo implementemos. En el ejemplo, necesitamos que el sistema nos permita registrar los pedidos de los clientes informando costos e impuestos aplicables, que nos permita mantener información de los clientes, razón social, número de CUIT, dirección; que nos permita mantener actualizado un catálogo de productos. Este sistema lo va a hacer independientemente de cómo lo implementemos, por eso algunos llaman a estos requisitos funcionales también, **requisitos esenciales** o **requisitos de comportamiento**.

- La otra gran categoría son los requisitos **no funcionales**, llamados también **calidad de servicio**, que describen determinadas propiedades que el sistema deberá tener o restricciones con las que deberá cumplir. En el ejemplo, hablamos de disponibilidad, de tiempo de respuesta, de sistemas operativos sobre los que deberá poder operar.

Lo que se puede ver de los ejemplos es que, independientemente del lenguaje que empleemos o del método o la notación que utilicemos para la especificación, los requisitos:

- Definen un objeto, una función o un estado.
- Limitan o controlan las acciones asociadas con un objeto, una función o un estado.
- Definen relaciones entre esos objetos, funciones y estados.

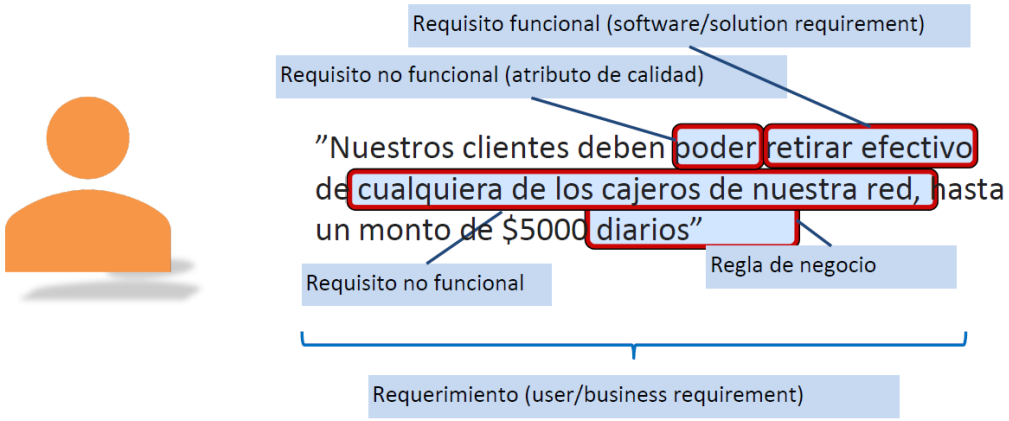
Es importante aclarar que aquí estamos hablando de objetos, que no tienen absolutamente nada que ver con los objetos de la orientación objetos, sino que son abstracciones de una entidad del mundo real que es importante para la discusión de los requisitos.

Distintas perspectivas

También podemos clasificar a los requisitos según distintos puntos de vista:

- Los objetivos y las necesidades del negocio o de la organización que dan origen a los requisitos de los usuarios, son llamados **requerimientos de negocio** o **requisitos de negocio**.
- A partir de esos requisitos podemos derivar los **requisitos del usuario** que describen lo que necesita cada uno de los usuarios con respecto al sistema a construir.
- Y finalmente, hay una perspectiva que es la que les va a interesar a los constructores del sistema: Los **requisitos de la solución** o **requisitos del software**.

[**V** / F] La ingeniería de requisitos ayuda a traducir las necesidades y expectativas de los interesados y transformarlas en requisitos del sistema o solución a construir.



Volviendo a nuestro ejemplo inicial, a partir de esta necesidad manifestada por nuestro usuario, se pueden derivar varios requisitos: Por un lado, la declaración podría ser clasificada como un requerimiento de negocio o como un requisito de usuario. Si examinamos más detalladamente nos vamos a encontrar con un requisito funcional que es “poder retirar dinero de la red”. Esto describe lo que el software tiene que hacer, lo que el sistema tiene que hacer independientemente de cómo lo implementemos. También aparece un requisito no funcional: ¿qué significa “poder” en este contexto? ¿Las 24 horas del día? ¿O hay ventanas de mantenimiento que nos limitan la posibilidad de retirar efectivo? Hay otro requisito funcional que es, hay una “red de cajeros”, ¿a dónde está esa red de cajeros? ¿Dónde se encuentran distribuidos esos cajeros? ¿Están siempre disponibles: están adentro de una sucursal bancaria, están en la vía pública? Ahí normalmente tenemos que indagar, tenemos que volver a entrevistar, probablemente a esta gente para poder profundizar estos detalles. Y también aparece una regla de negocio que es la posibilidad de retirar esos $ 5.000 diarios. ¿Esta regla de negocio podría llegar a cambiar? ¿Tiene algo que ver con la categoría del cliente? Es muy interesante esto porque a partir de una declaración relativamente sencilla aparecen y se derivan un montón de interrogantes. Adicionalmente, hay un problema grave en todo esto: el usuario está

asumiendo que la única manera de obtener dinero en efectivo será mediante un cajero automático. Entonces quizás debamos plantear a los interesados otras alternativas, como por ejemplo retirar dinero en efectivo de la caja de supermercado. Lamentablemente, al enunciar una necesidad, muchas veces los usuarios, los stakeholders o interesados, están influenciados por soluciones preconcebidas. Nuestra tarea es llegar al fondo de la cuestión e identificar el verdadero problema que se debe resolver independientemente de estas ideas iniciales o preconcebidas que puedan llegar a tener a nuestros interlocutores.

Los requisitos son una pieza fundamental en el desarrollo de software porque constituyen la base que vamos a emplear para, entre otras cosas, diseñar y construir la solución, definir los casos de prueba y planificar y organizar las actividades de desarrollo de software.

Ingeniería de requisitos

¿Qué tipos de actividades incluye?

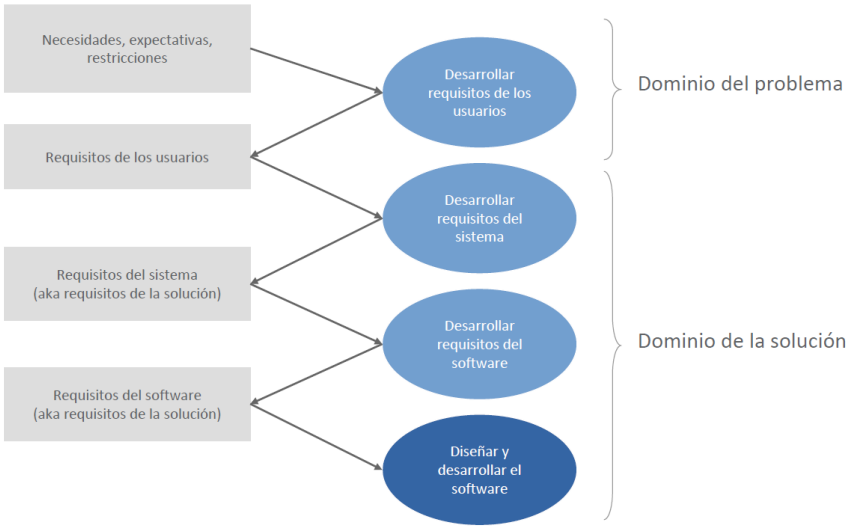
La ingeniería de requisitos incluye varias actividades:

- Por un lado, las relacionadas con el **desarrollo de requisitos** que van a incluir:
  - Aquellas que tienen que ver con el descubrimiento, es decir, determinar cuáles son los requisitos.
  - Las relacionadas con el análisis, qué tienen que ver un poco con lo que hemos hecho recién, detectar y resolver conflictos entre los requisitos, derivar nuevos requisitos, identificar la interacción con el medio ambiente.
  - La especificación que consiste en documentar los requisitos para facilitar su revisión, su evaluación, su aprobación, para facilitar compartir esa definición entre los distintos interesados.
  - La verificación que consiste en evaluar si escribimos correctamente los requisitos.
  - La validación que consiste en evaluar si entendimos correctamente los requisitos que nos han formulado.
- La otra gran actividad, es la **administración de los requisitos**. Inevitablemente, los requisitos van a cambiar a lo largo del desarrollo del producto. No está necesariamente mal eso, pero tenemos que administrarlo, tenemos que estar seguros de estar trabajando sobre la versión actualizada de los requisitos.

Los requisitos se emplean para…

- ☐ Definir el equipo de trabajo
- ☑ **Organizar el trabajo**
- ☐ Diseñar las interfaces
- ☑ **Identificar casos de prueba**

Podemos ver a la ingeniería de requisitos como una actividad que a través de sucesivas transformaciones refina las necesidades y expectativas de los usuarios e interesados en una definición de los requisitos que deberá satisfacer el software a desarrollar y que van a constituir la base para el diseño y el desarrollo del producto. Una parte del trabajo tiene que ver mucho con el dominio del problema y otra buena parte del trabajo tiene que ver con el dominio de la solución.



¿Qué implica?

Estas actividades que acabamos de anunciar no son disjuntas. No suceden, tampoco en forma secuencial, sino que se superponen con variados grados de intensidad a lo largo de todo el ciclo de vida, se realimentan entre sí. Es claramente un proceso iterativo e incremental.

Los requisitos y el ciclo de vida del proyecto

Normalmente en los proyectos de desarrollo de software, el entendimiento de los requisitos es general al comienzo y se va refinando conforme se avanza en el entendimiento del problema y en el desarrollo del producto. Es usual, como decíamos anteriormente, que haya cambios en los requisitos por las más diversas causas. A veces cambia el contexto del negocio, se entienden mejor las necesidades, aparecen regulaciones, todo eso de alguna manera se tiene que reflejar en el proceso de ingeniería de requisitos.

El proceso de ingeniería de requisitos es muy sensible a aspectos humanos, a aspectos relacionados con la comunicación, con el aprendizaje, la política, el poder.

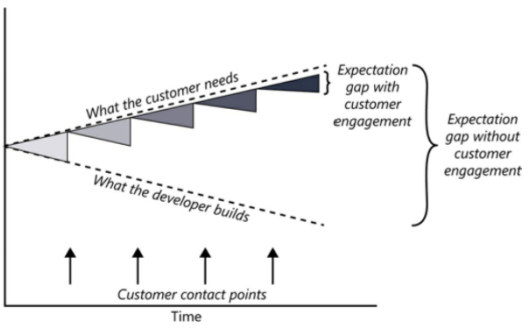
Complicaciones

- *Falta de conocimiento nuestro acerca del dominio* del problema que estamos atacando.
- *Dificultades para consensuar qué es lo que se quiere*, ya que hay distintos actores, distintos interesados, distintos usuarios con objetivos, muchas veces contrapuestos con respecto a la solución que hay que desarrollar.
- Es *difícil involucrar* activamente *a los interesados*.
- *Omitimos algún interesado* clave, simplemente por desconocimiento.
- Es complicado consensuar las agendas, lo que lleva a tener *dificultades en la planificación* de las actividades de ingeniería de requisitos. Recordemos que hay que entrevistar a usuarios potenciales del producto a desarrollar para entender qué es lo que necesitan, entender cuál es el problema que necesitamos resolver.
- Los *requisitos* son naturalmente *ambiguos*, hay muchos problemas en cuanto al vocabulario del dominio del problema.
- Los *requisitos* presentan un fenómeno *inflacionario*, tienden a crecer, tienden a multiplicarse, y de alguna manera hay que acotar ese fenómeno para poder entregar el producto deseado.
- Hay *dificultades de comunicación*. Algunos de los problemas de comunicación tienen que ver con el diferente vocabulario, el diferente lenguaje. La otra es no saber bien qué preguntar. Justamente por ignorancia acerca del área de aplicación.
- Por supuesto que hay *temas culturales* y ni que hablar de los *temas políticos*. Detrás de cada usuario, de cada interesado que tiene un objetivo contrapuesto con respecto al software a desarrollar, hay muchas veces intereses políticos, intereses que tienen que ver con espacios de poder, intereses que tienen que ver con manejo presupuestario, a veces con el manejo de la agenda estratégica.
- *La incorrecta gestión de los cambios*. Los cambios en los requisitos son inevitables. En países como Argentina muchas veces hay cambios regulatorios de la noche a la mañana y eso tiene que estar reflejado en la definición de los requisitos que va a tener que implementar el software.

[V / **F**] Es inusual que existan cambios en los requisitos.[1]

Expectativas e involucramiento

Es muy importante entonces, manejar adecuadamente las expectativas y el involucramiento de todos los interesados. Cuanto más contacto tengamos con los clientes, con los usuarios y con otros interesados, tanto mejor. Si mantenemos distancia, si no nos reunimos frecuentemente vamos a empezar a tener una brecha importante entre lo que nosotros percibimos que el cliente necesita y lo que terminamos construyendo. Por eso es



[1] Inevitablemente, los requisitos van a cambiar a lo largo del desarrollo del producto

importante tener un proceso de desarrollo de software iterativo e incremental que haga énfasis en este feedback lo más frecuente posible.

Ventajas

La ingeniería de requisitos correctamente realizada tiene múltiples ventajas:

- Tenemos *menos defectos en el producto entregado*. Sabemos que detectar efectos en los requisitos nos produce una economía de escala muy importante.
- Nos permite *reducir el retrabajo*.
- Nos permite *desarrollar y entregar software* de manera *más rápida*.
- Nos permite *acotar la funcionalidad* que desarrollamos si es *que deja de ser necesaria*.
- Genera *menos costo de mantenimiento/evolución*.
- Hay *menos malentendidos*. Esto que mencionábamos antes de los contactos frecuentes, ayudan a que vayamos entendiendo progresivamente el vocabulario del problema y vayamos ofreciendo y validando soluciones posibles.
- *Se acota el tema de los cambios indiscriminados* a los requisitos.
- Hay *menos caos* en los proyectos.
- Los *clientes* están más *contentos*. Los miembros del *equipo* están más *satisfechos*.
- Nos *ayuda a tener productos que hacen lo que se supone que deben hacer*.

¿Quién hace ingeniería de requisitos?

Un rol, muchos roles, diferentes nombres

A veces es un rol de dedicación exclusiva, a veces es un sombrero diferente que usa alguien que tiene otro rol. En algunos casos, en algunas organizaciones, nos vamos a encontrar con que hay roles con el título de analista funcional, analista de negocios, analista de requisitos, más recientemente, jefe de producto o dueño de producto. Lo importante es que este rol es un traductor que debe entender lo que el usuario final está diciendo para transformarlo en una especificación de lo que el sistema, la solución o el producto a construir va a tener que hacer. Tienen que ser un buen observador, tiene que entender e interpretar lo que el usuario hace o quiere hacer. Tiene que tener una vocación para idear mejores maneras de hacer lo que el usuario está haciendo. Debe registrar, de alguna manera y con cierto nivel de detalle, los resultados obtenidos mediante una especificación de requisitos o algún tipo de modelo.

Habilidades necesarias

Más allá de las habilidades técnicas, tiene que tener muchas **habilidades blandas**:

- Saber escuchar
- Saber entrevistar
- Saber preguntar activamente
- Ser analítico en su forma de pensar
- Pensar sistémicamente: tiene que poder ver el bosque y si hace falta también ver el detalle
- Capacidad y voluntad de aprender
- Tener muy buenas habilidades para facilitar reuniones y talleres: ustedes saben que las reuniones suelen ser una de las actividades que más tiempo consume en las organizaciones y que, en general, traen muchos dolores de cabeza si no se manejan adecuadamente porque se quema el tiempo y se queman los recursos en reuniones mal organizadas y mal realizadas.
- Poder liderar estas actividades
- Capacidad de observación
- Muy buena habilidad para comunicarse y para organizarse (FUNDAMENTAL).
- Saber crear y preparar modelos.

[**V** / F] Detectar tempranamente defectos en los requisitos reduce el trabajo.

¿Cuán formal debe ser?

El proceso de ingeniería de requisitos puede ser muy formal, muy informal, o algo en el medio. Muy informalmente nos vamos a encontrar con organizaciones en las que por ahí no hay ningún proceso, no hay estándares, no se documenta. En la otro apunta, nos vamos a encontrar con procesos muy burocráticos, muy pesados, con estándares críticos, con requisitos de documentación que hay que generar, con revisiones rigurosas. En definitiva, el punto medio es el mejor en donde vamos a encontrar el punto de equilibrio que obviamente va a depender muchísimo del tipo de software que estemos desarrollando, el tipo de cliente que tengamos, el tipo de interesados que tengamos.

Diferentes contextos

También nos vamos a encontrar en situaciones que van a tener que ver con el tipo de involucramiento del cliente:

- En algunos casos **el cliente/usuario no va a formar parte del equipo** entonces se suceden situaciones como las que mencionábamos antes:
  - Nos es difícil alcanzar el consenso.
  - Los requisitos de muy alto nivel que nos formulan estos usuarios hay que bajarlos a nivel de detalle como para poder transmitírselo a un desarrollador, lo cual implica que haya una posibilidad ahí de error de interpretaciones de traducción.
  - A veces la organización que formula la solicitud o el pedido, el requerimiento, lo hace y la organización que desarrolla es otra en otra entidad, lo cual implica la necesidad de formular a lo mejor algún tipo de contrato, algún tipo de documento formal y todo eso complica obviamente las cosas.
- Si tenemos la suerte de **contar con un representante del usuario**, con un representante del cliente en el equipo:
  - Las cosas son mucho más fáciles porque ese experto está disponible para que lo consultemos si tenemos alguna duda, está ahí con nosotros la mayoría del tiempo.
  - Se evita la necesidad de comunicarse formalmente, disminuye la necesidad de armar documentos demasiado detallados
  - El feedback es mucho más fluido, y de alguna manera forma parte del equipo.
  - Por supuesto que tiene esto algunas contraindicaciones: si este usuario representante no es realmente de peso y no representa adecuadamente a los usuarios, estamos en problemas; también es cierto que tener a alguien experto en el negocio sentado en el equipo de desarrollo es una especie de claudicación, de entender el problema por parte de la gente del equipo de desarrollo. En definitiva, hay que buscar ahí un escenario intermedio.

También es cierto que el tipo de software que desarrollemos va a tener una profunda influencia, como decíamos antes, en el proceso. No es lo mismo desarrollar software en forma interna, en un área de sistemas en donde mis clientes pertenecen a la misma organización, que desarrollar software a medida cumpliendo a lo mejor con los requerimientos explicitados en una licitación pública. A veces estamos desarrollando un producto de software que va a impactar en el mercado masivo, a veces estamos desarrollando software que va a ser usado internamente, claramente ahí hay un tema importante de escala. A veces estamos desarrollando un servicio que va a estar basado en software. A veces vamos a estar desarrollando software para un producto sobreembebido o software en tiempo real que va a estar embebido en un producto, lo cual complica muchísimo la definición de los requisitos y el proceso de desarrollo en general, porque una vez que ese software está incluido en el dispositivo, va a ser probablemente muy difícil de ajustar y de cambiar, con lo cual es crítico entender cuáles son los requisitos.

Actividades y técnicas

Hay múltiples técnicas que vamos a ir viendo en futuros vídeos aquí simplemente vamos a presentar algunas de ellas.

- **Descubrimiento**: Algunos autores hablan de obtención de requisitos, como si los requisitos estuvieran ahí y los pudiéramos ir a recoger. Lo cierto es que los requisitos se van descubriendo paulatinamente. ¿Cómo?
  - Entrevistas a los usuarios.

- Talleres de requisitos. Los talleres son reuniones un poco más estructuradas en donde hay un facilitador que organiza las actividades en donde hay exposición de algunos expertos en algunos casos.
- Encuestas que vayan indagando acerca de cuáles son los problemas actuales, que vayan indagando acerca de las complicaciones actuales con los procesos internos y que nos permitan identificar posibles mejoras.
- Sesiones de brainstorming. Ante un problema, formular soluciones. Sesiones de brainstorming y los Focus Group son muy adecuados para productos de consumo masivo. Los Focus Group son reuniones de trabajo con posibles usuarios del producto que queremos desarrollar. En el caso para el mercado masivo son muy útiles.
- Observación. Ir a ver qué es lo que está haciendo el usuario con las herramientas actuales, también nos permite entender cuáles son sus problemas y cuáles son las posibles soluciones. En definitiva, nos permite derivar cuáles son los requisitos que va a tener que resolver el software. Una versión extrema de la observación es sentarse y hacer el trabajo uno mismo en lugar del usuario para entender realmente cuáles son sus problemas.
- Analizar documentos actuales, analizar sistemas actuales en el caso de que tengamos que reemplazar algún sistema ya existente.
- Plantear escenarios de utilización.
- Presentar prototipos que ayudan a discutir y a entender si descubrimos bien los requisitos.

[**V** / F] Los *focus groups* son adecuados para descubrir requisitos en productos de consumo masivo.

- Una vez que tenemos entendido cuáles son los requisitos, tenemos que analizarlos. **Análisis** y **especificación** usualmente se realiza en simultáneo. El **análisis** se enfoca más bien en identificar y resolver conflictos entre los requisitos, descubrir los límites del sistema y la interacción entre el sistema y el medio ambiente, y también, el análisis se encarga de derivar requisitos del sistema a desarrollar. No todo lo que nos dicen los usuarios van a automáticamente transformarse en requisitos. Van a surgir otra vez probablemente requisitos. Al usuario final probablemente no se le ocurran temas que tengan que ver con las regulaciones del mercado en el que operan, o temas de compliance con algún estándar. Por eso son requisitos con los que vamos a tener que cumplir, y que probablemente no nos lo digan en forma explícita. El modelado forma parte del análisis, tiene como propósito producir modelos abstractos del sistema del software o de la solución a desarrollar. No incluyen aspectos de diseño esos modelos. Normalmente vamos a encontrarnos con que va a hacer falta desarrollar más de un modelo. Vamos a hablar, por ejemplo, de modelos de dominio, de modelo de casos de uso, de historias de usuario, etc. La **especificación** consiste en producir uno o varios documentos, no necesariamente físicos, pueden ser obviamente electrónicos, de formalidad variable, con diferentes niveles de detalle, con la descripción del comportamiento que debe tener la solución a desarrollar. Normalmente no es algo que tenga que ver con el diseño, sino que va a tener que ver con el comportamiento, con la esencia, con lo que el sistema tiene que ser, independientemente de cómo lo implementemos.
  Algunas de las técnicas de análisis-especificación, son las siguientes:
  - La clasificación de requisitos. No todos los requisitos son igualmente importantes, deberíamos poder estar en condiciones de priorizarlas, priorizarlas de acuerdo a por ejemplo, “esto tiene que estar sí o sí”, “esto podría estar” y “esto definitivamente podría esperar para más adelante”. Para poder hacer esa clasificación y esa priorización, necesitamos poder estar en condiciones de negociar.
  - Negociar no es imponer una visión, sino llegar a un acuerdo entre las partes acerca de un tema determinado.
  - El modelado de escenarios, el modelado de dominio, el prototipado, la especificación utilizando lenguaje natural o algún otro tipo de notación, también son técnicas aplicables al análisis y a la especificación de requisitos.
  - Utilizar reglas de negocio que son mecanismos claros para especificar políticas, para especificar determinado tipo de conductas en la organización también son muy útiles.

- Criterios de aceptación y casos de prueba. Definir bajo qué condiciones vamos a dar por satisfechas las necesidades y expectativas del usuario que hemos especificado mediante requisitos es un mecanismo espectacular para asegurarnos de haber entendido qué es lo que tenemos que hacer, qué es lo que tenemos que construir. Si podemos definir casos de prueba, o criterios de aceptación, en conjunto con los interesados con los usuarios de alguna manera vamos a estar definiendo bajo qué condiciones el sistema va a tener que operar y de alguna manera eso también constituye una especificación de cómo se tiene que comportar el sistema.

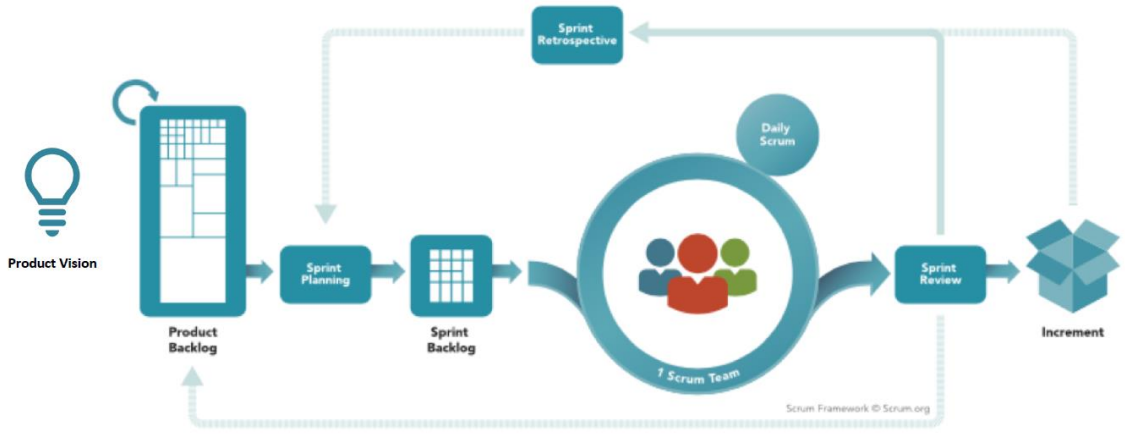
[V / **F**] El análisis y la especificación de requisitos generalmente se realizan secuencialmente en forma independiente.[2]

- Finalmente vamos a necesitar **verificar** los requisitos. Verificar implica asegurarnos de que hemos descripto correctamente los requisitos y para eso vamos a utilizar:
  - Revisiones.
  - Inspecciones.
  - Prototipos.
  - Los mismos criterios de aceptación y los mismos casos de prueba también nos van a servir para chequear si hemos descripto correctamente los requisitos.
- Por supuesto que las mismas técnicas las vamos a poder utilizar para la **validación**. ¿Entendimos correctamente los requisitos? Esto es claramente una actividad que vamos a realizar junto con los usuarios, con los interesados. Si le podemos mostrar un prototipo, de alguna manera vamos a estar validando que entendimos correctamente sus necesidades, que entendimos correctamente los requisitos.

Ingeniería de requisitos en el ciclo de desarrollo

Ahora bien, ¿qué es lo que pasa con el proceso de ingeniería de requisitos en el contexto del ciclo de desarrollo? Bueno hay varias alternativas, hay varios ejemplos que podemos tomar. En el pasado, estamos hablando hace 40, 50 años atrás, algunos sostenían la visión de que el software había que desarrollarlo de manera similar a como se desarrollaba el hardware. Entonces sería importante tener la definición completa de lo que el producto tenía que hacer antes de proceder al desarrollo. Entonces usualmente con ese tipo de ciclo de vida, con ese tipo de ciclo de desarrollo, todas las actividades de ingeniería de requisitos estaban concentradas al inicio de los proyectos, al inicio del ciclo de desarrollo. En los ciclos de vida un poco más modernos, las actividades de ingeniería de requisitos, como decíamos al principio, van sucediendo en forma iterativa e incremental a lo largo de todo el ciclo de vida.

Un ejemplo: Scrum



[2] Análisis y especificación usualmente se realiza en simultáneo.

Un ejemplo de ciclo de vida iterativo e incremental es Scrum, que ya hemos mencionado en un momento con anterioridad. Scrum divide al ciclo de vida del producto en etapas, en releases. Cada release es una versión que sale del producto. Para preparar una versión, se utiliza a su vez una serie de interacciones llamadas Sprints. Un Sprint es una iteración que dura entre 2 y 4 semanas. Durante ese sprint el equipo está abocado a refinar la definición del producto y a desarrollar y probar el producto. Al final de cada sprint el equipo entrega una versión potencialmente operativa del producto final. No necesariamente la versión que se vaya a entregar, pero tiene que ser una versión que esté probada y que esté estable, no necesariamente que sea completa. Entonces el proceso arranca con una definición de la visión del producto, esto puede estar plasmado en un documento, en una presentación. Esa visión del producto, la establece el product owner. El product owner es, como su nombre lo indica, el dueño del producto. Es un representante del usuario que trabaja junto al equipo de desarrollo y que es el encargado de definir qué es lo que el sistema tiene que hacer. Ese documento de visiones, esa definición de visión es una descripción de muy alto nivel de cuáles son los problemas que busca resolver el producto a desarrollar, cuáles son sus objetivos, cuáles son potencialmente sus usuarios y en todo caso, grandes funcionalidades que tiene que incluir. Esa definición de visión permite establecer el Backlog del producto. ¿Qué es el Backlog del producto? Una lista priorizada de las funcionalidades que hay que desarrollar. No es una lista estática, sino que se va a ir refinando a medida que vayamos avanzando en el desarrollo. Entonces, el equipo de trabajo, al comienzo de cada sprint, se reúne con el product owner, discute cuáles son las features, las funcionalidades prioritarias a desarrollar, decide, de común acuerdo, cuáles son las features, las funcionalidades que se pueden tomar para desarrollar en el sprint, esto en función de la capacidad operativa del equipo, previamente, ha habido una estimación de esfuerzo, duración, costo de desarrollar cada una de esas funcionalidades, esas funciones, en definitiva, son el equivalente a nuestros requisitos. Y con esa información, se prepara la planificación del sprint, en definitiva, el Sprint Backlog es otro Backlog que lo que tiene es una definición de las actividades que hay que llevar a cabo para implementar cada una de las funcionalidades que se han tomado para este Sprint. El equipo inicia su trabajo, hay una dinámica diaria, hay una reunión que se llama Daily Stand up Meeting, en donde el equipo en su totalidad se reúne, revisa qué fue lo que hizo el día anterior, revisa o define qué es lo que va a hacer durante el día vigente, el día de la fecha, se plantean algunos obstáculos, se plantean problemas, no se discuten soluciones, simplemente se define el plan de acción a seguir. Entonces tenemos como una mini iteración, que es la iteración diaria, una iteración más grande que es la iteración del sprint y una todavía más grande que es la del release. En definitiva, son todos los loops que estamos viendo en el diagrama. Entonces, al final de cada sprint, lo que vamos a tener es una versión del producto probada, estable, que cumple con los requisitos, con las funcionalidades comprometidas, y que potencialmente, se podría entregar si fuera el caso, por supuesto, que poder arribar a una definición completa del producto, va a llevar varios Sprints. Fíjese que al final de cada sprint lo que tenemos es una retrospectiva. En esa retrospectiva, lo que realiza el equipo es qué funcionó en el equipo, qué es lo que no funcionó, qué obstáculos tuvieron. De alguna manera, estamos implementando ahí un feedback que nos permite refinar la planificación y nos permite a su vez mejorar el proceso de desarrollo. También, al final de cada sprint, estamos en condiciones de refinar el product Backlog. Una de las cosas que hay que hacer con el product Backlog, es decidir si la lista de funcionalidades, en definitiva, la lista de requisitos que tenemos ahí cambia de prioridad, sigue siendo la misma o si hay que hacer algo sobre esa lista.

[**V** / F] El product owner es un representante del usuario que trabaja junto al equipo de desarrollo. Se encarga de definir qué es el producto o servicio a desarrollar.

Scrum no necesariamente utiliza una técnica llamada User Stories. User Stories es una manera de describir qué es lo que el sistema necesita con respecto al sistema. Qué es lo que necesita que el sistema le ayuda a resolver. No es un requisito en el sentido absolutamente formal del término, pero sí es un puntero a una charla que hay que tener con el especialista, en este caso el product owner. Por ejemplo, como feature yo puedo tener una definición de una historia de usuario que nos diga como alumno quiero poder inscribirme en un curso de una materia, esa puede ser una definición de una user story. Claramente, para que sea un requisito en el sentido clásico que hemos descripto antes, hace falta alguna información adicional. En el caso de la User Stories, esa información adicional lo dan los criterios de aceptación asociados a esa historia de usuario. Vamos a ver un poco más de eso más adelante. Pero bien, este es un ejemplo de qué es lo que pasa con los requisitos a lo largo de un ciclo de desarrollo que sigue un

modelo de proceso iterativo e incremental. Vamos a profundizar en próximos vídeos, cada una de las técnicas que hemos presentado y vamos a profundizar qué lo que pasa con cada una de ellas en función de la metodología o estilo de desarrollo elegido.

Disciplinas relacionadas

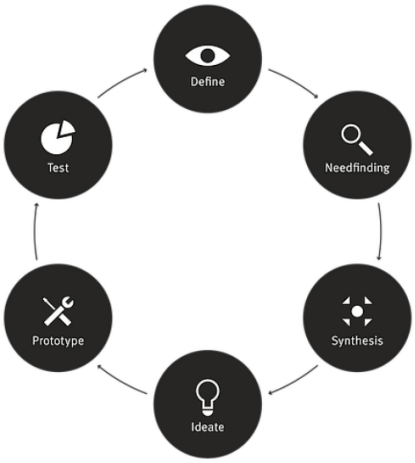
Hay 2 disciplinas relacionadas con la ingeniería de requisitos que es importante mencionar:

Business Analysis

Una es el business analysis. Esta disciplina busca aplicar conocimiento, técnicas y herramientas para identificar problemas, determinar necesidades, y formular y recomendar soluciones para satisfacer dichas necesidad. No es necesariamente la solución es software, sino que puede implicar cambiar un proceso de negocio, transformar una estructura organizacional, cambiar el rumbo estático de la compañía. Es una disciplina si se quiere más general y que está muy alineada con la ingeniería de requisitos porque siempre terminamos identificando problemas que no necesariamente se resuelven con software. Y de hecho, si recordamos lo que hemos dicho con anterioridad, el software siempre forma parte de un sistema, con lo cual, es entendible que cuando estemos analizando los requisitos de una pieza de software de una futura aplicación nos encontramos con que hay otros problemas para mejorarlo.

Design Thinking

La otra disciplina relacionada es Design Thinking. Design Thinking viene del lado del diseño de producto, no necesariamente del diseño de productos de software. Es un método centrado en las personas que tienen como propósito resolver problemas difíciles y mal definidos. Sigue un ciclo iterativo e incremental. Se fusiona muy bien con Scrum, con las metodologías ágiles, es perfectamente aplicable en dichos contextos. De hecho, en la Universidad de San Gallen hay un cuerpo de docentes que está trabajando, investigadores que vienen trabajando desde hace tiempo en la fusión entre ambas disciplinas.



[V / **F**] Dentro de la ingeniería de requisitos es poco de importante contar con habilidades sociales ya que existen herramientas que simplifican el trabajo.[3]

[3] Más allá de las habilidades técnicas, tiene que tener muchas habilidades blandas.