

ESPECIFICACIÓN DE REQUERIMIENTOS II



Especificar requisitos correctamente.

Refinar el Diagrama de Casos de Uso.

Especificar requerimientos funcionales.

Especificar requerimientos no funcionales.

Especificar historias de usuario.

01

Especificación de Casos de Uso

02

Especificación de requerimientos funcionales

03

Especificación de requerimientos no funcionales

04

Historias de usuario



¡La única manera de hacer con exactitud lo que se nos pide es que se nos indique con exactitud qué es lo que se desea que hagamos!

La intención de lo que se desarrolla a continuación es actualizarnos en cómo expresar con exactitud qué es lo que se desea **construir**, siguiendo las dos rutas donde bien podemos "**refinar**", es decir, ir a un mayor detalle de nuestro diagrama de casos de uso o ir hacia nuestras historias de usuario.

En ambos casos siempre será necesario precisar mejor los **requisitos** de calidad. Estos tres aspectos son los pivotes de este tema.



01 Especificación de Casos de Uso

El Modelo de Casos de Uso está conformado por tres elementos:



Diagrama de casos de uso.



Especificación de los casos de uso.



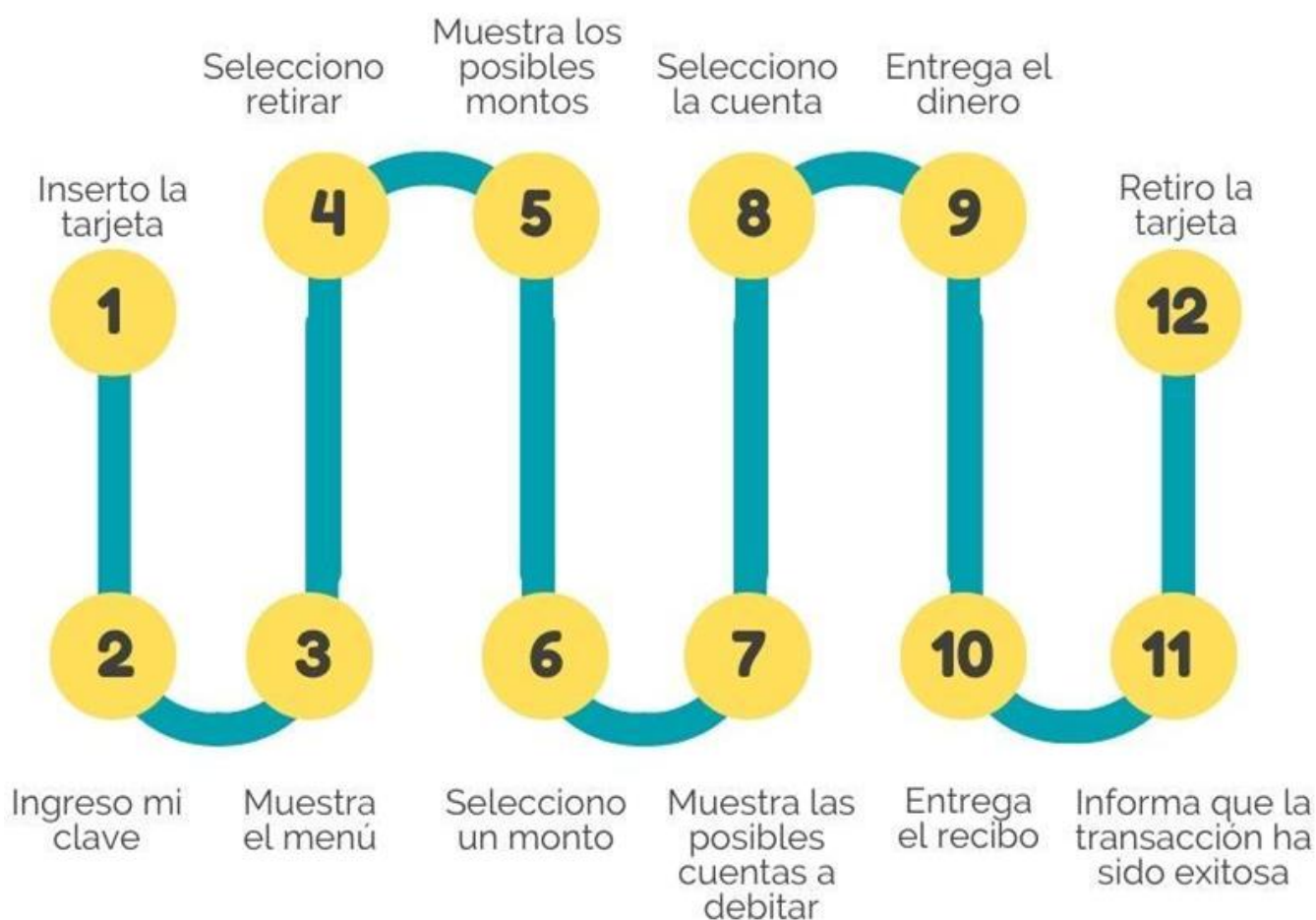
Lista de casos de uso vs actores.

La meta, entonces, es construir el segundo elemento de nuestro modelo de casos de uso; pero tal como están los casos de uso del diagrama de casos de uso inicial, esta especificación no es posible. Para ello primero debemos "refinar" este diagrama.

Comenzamos por dar la definición de caso de uso:

Un caso de uso (refinado) define una secuencia de **acciones** ejecutadas por un sistema que producen un resultado **observable** de valor para un actor.

Por ejemplo, esta es la secuencia que todos seguimos, en general, para retirar un monto de dinero en un cajero automático:

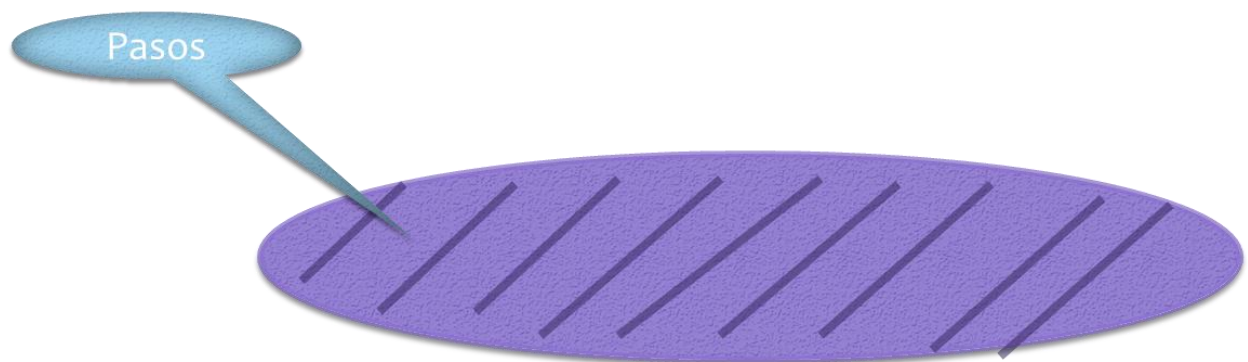


Refinarlo significa que tiene la cantidad de pasos necesarios, ni más ni menos. En este ejemplo faltan los pasos ejecutados por el cajero automático; esto también hay que precisarlo.

¿Qué podemos hacer para simplificar este número de pasos? Se recurre a la propuesta de relaciones tipo **Extend** o **Include** entre los Casos de Uso.

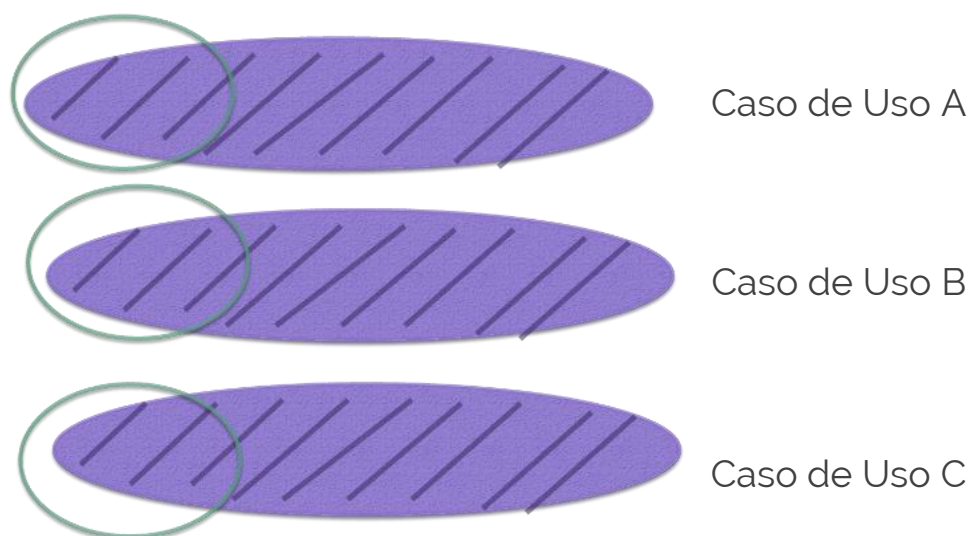
Veamos cómo las identificamos:

Relaciones Include: este es un Caso de Uso no refinado (figura 1):



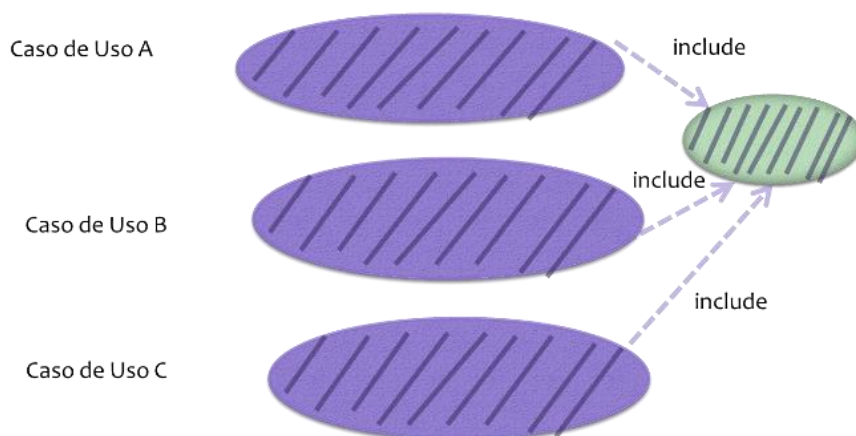
Fuente: elaboración propia

Y observamos que hay tres Casos de Uso en mi diagrama de Casos de Uso inicial, en los que un subconjunto de pasos se repiten en los tres (figura 2):



Fuente: elaboración propia

Sacamos este subconjunto de pasos y establecemos una relación de **Include** entre este subconjunto de pasos y los otros tres Casos de Uso. Tal como se observa en la figura 3:

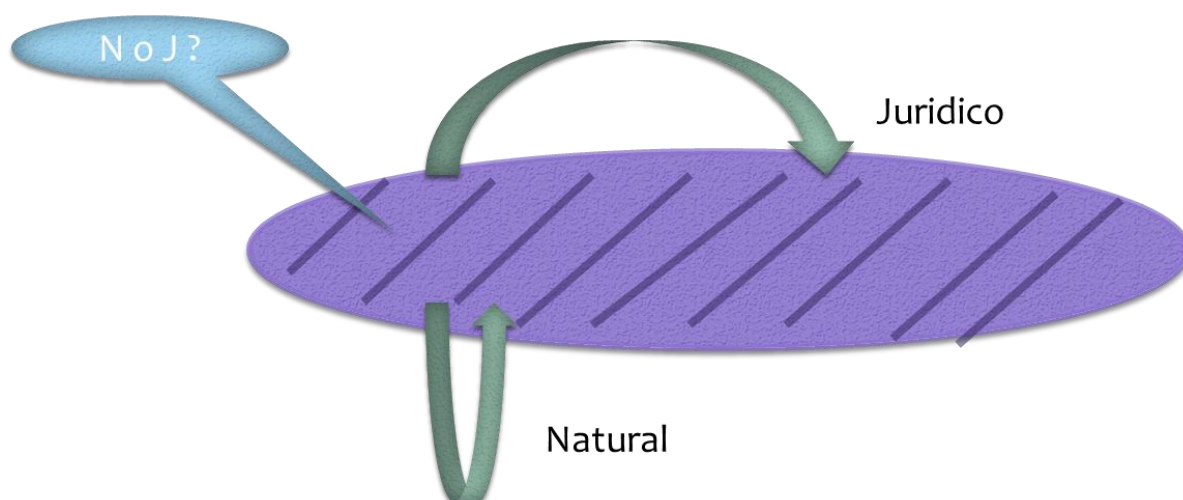


Fuente: elaboración propia

De esta manera, si bien ahora tengo un Caso de uso adicional, todos los casos de uso contienen **solo los pasos que necesitan**. Esto conduce a diseños modulares, de bajo acoplamiento y con componentes reutilizables.

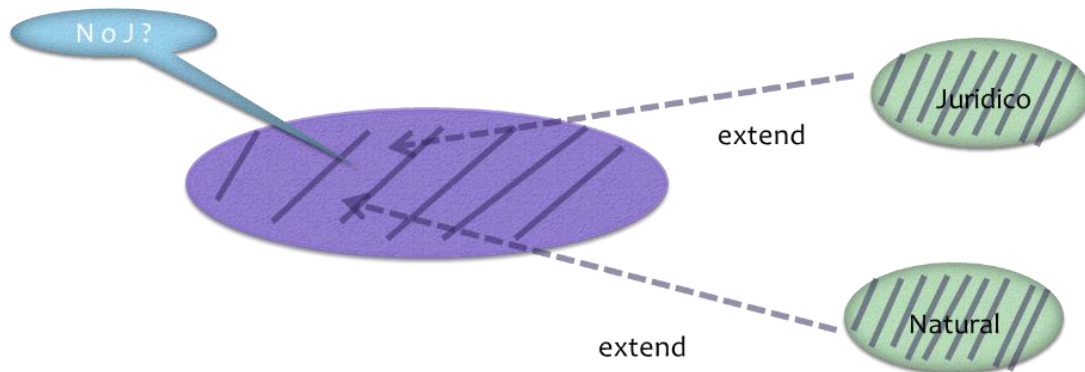
Veamos ahora otra situación.

Relaciones Extend: en este Caso de Uso se sigue un subconjunto de pasos si y solo si se da la condición de Persona Natural. Si es Persona Jurídica sigue otro subconjunto de pasos (figura 4):



Fuente: elaboración propia

A este caso de uso le aplicamos una relación de **Extend**. Sacando de él aquel subconjunto de pasos que solo corresponden a la Persona jurídica y el subconjunto de pasos que solo corresponden a la Persona natural (figura 5):

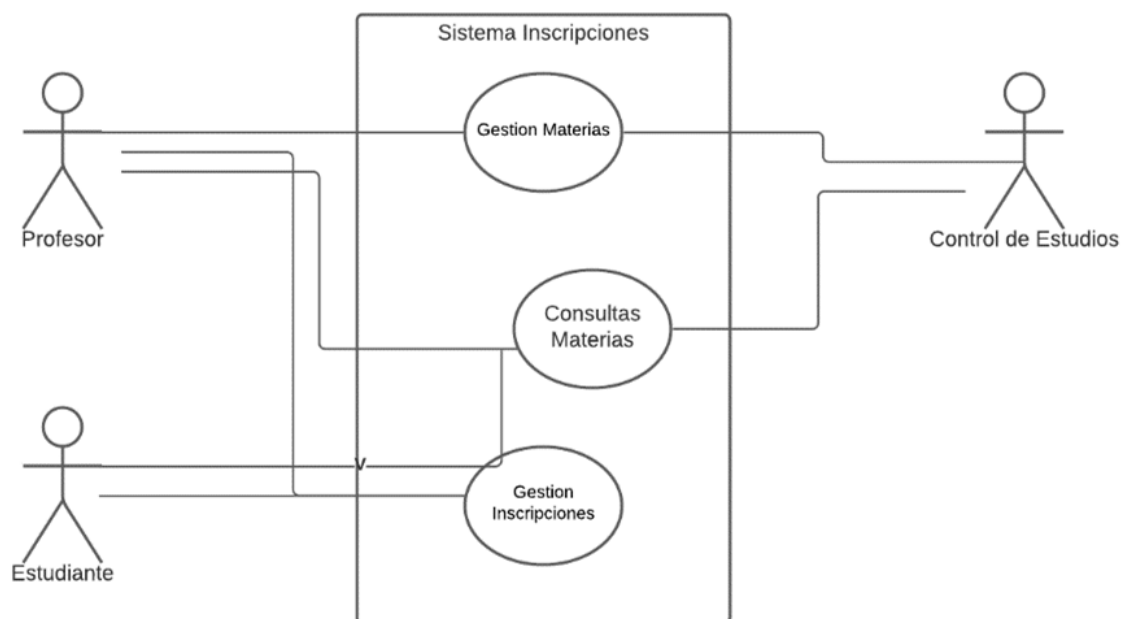


Fuente: elaboración propia

Nuevamente, la relación de *Extend* propicia modularidad y reutilización.

Prestemos atención a la sintaxis utilizada en ambos diagramas para representar las relaciones de *Incluye* y de *Extend*. Cuando se refina un diagrama de Caso de Uso también se revisan los **actores**.

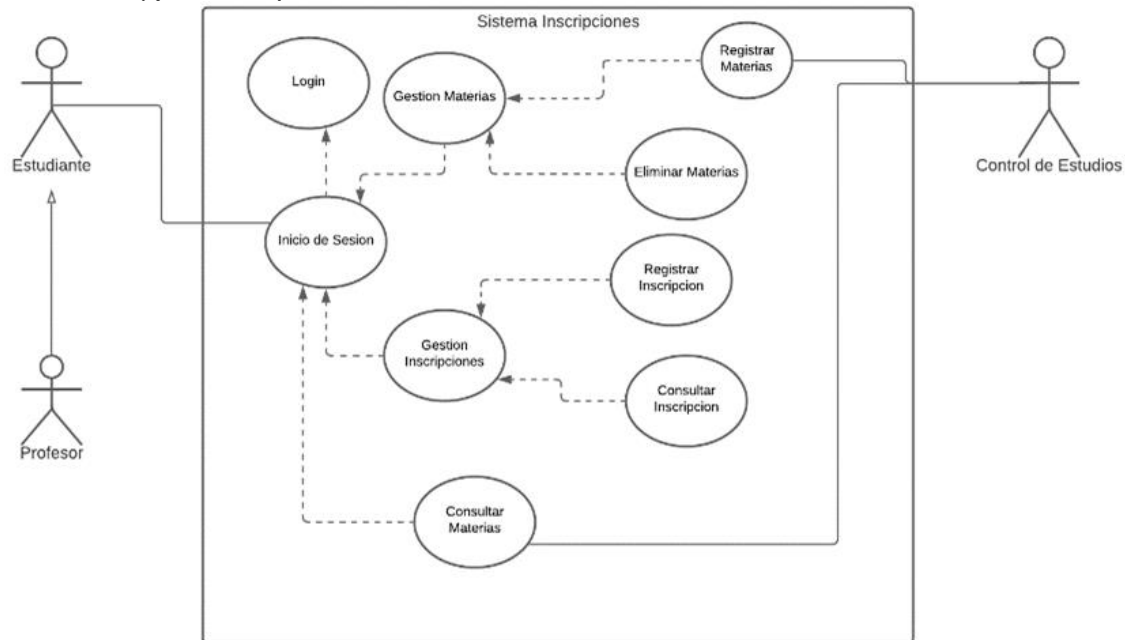
En el ejemplo del profesor se visualiza que puede “hacer” más cosas que el estudiante:



Fuente: elaboración propia

A fin de simplificar el diagrama se establecen relaciones de **generalización** entre los actores, colocando en el tope al actor (usuario) que menos cosas puede hacer con la app, y al final al **más poderoso**.

Nuestro diagrama quedaría así:



Fuente: elaboración propia

Nótese que de esta manera el diagrama queda más "limpio", pero se pierde información. Ahora no sabemos lo que el profesor puede hacer; por ello se hace la lista de actores vs. Casos de Uso, que es el tercer elemento de nuestro Modelo de Casos de Uso (tabla 1):

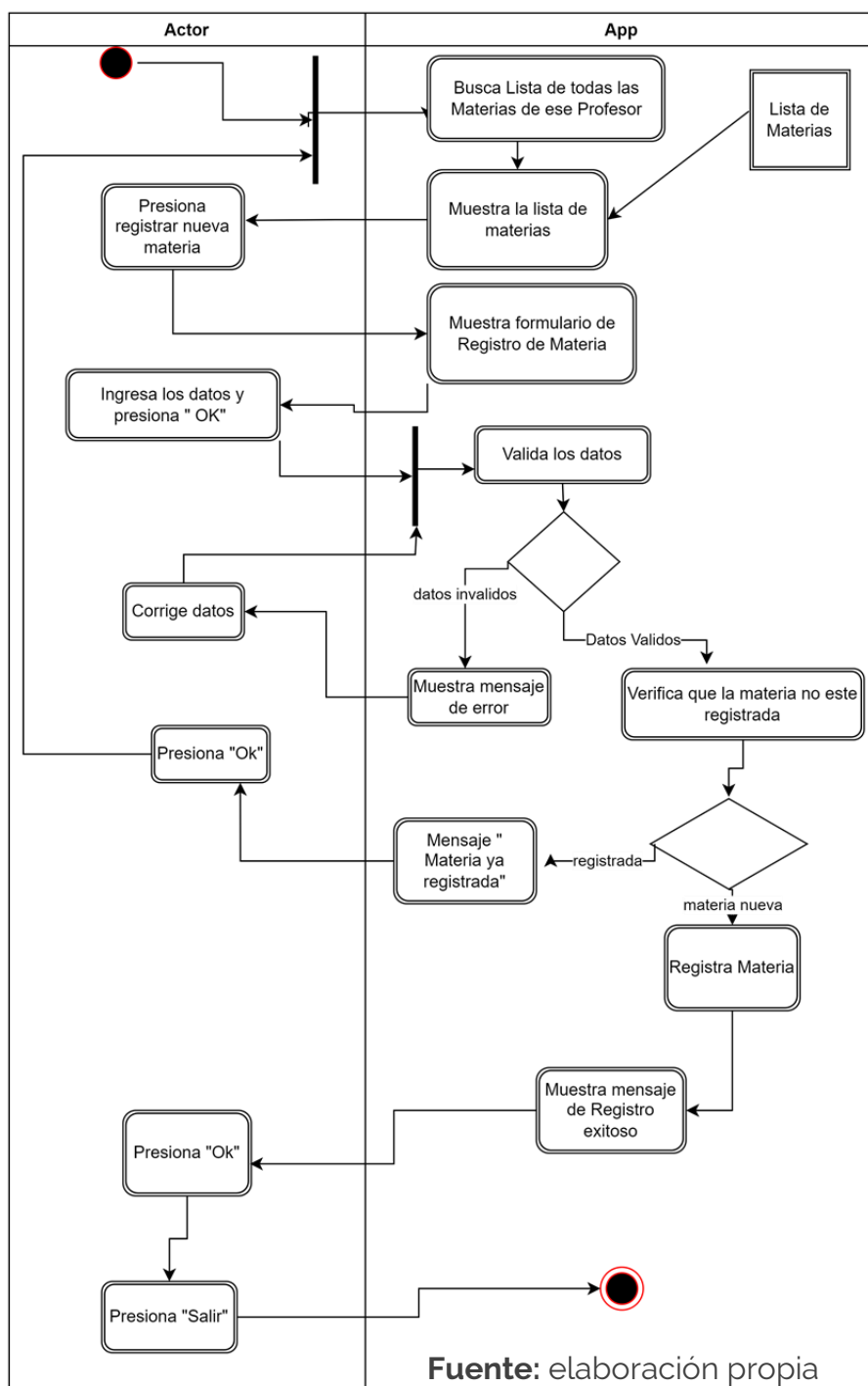
Actor	Caso de Uso
Estudiante	<ul style="list-style-type: none"> • Consultar materia. • Registrar inscripción. • Consultar inscripción. • Loguearse
Profesor	<ul style="list-style-type: none"> • Consultar materia. • Registrar materia. • Eliminar inscripción. • Consultar inscripción. • Loguearse.

Fuente: elaboración propia

Luego de que se han identificado todas las posibles relaciones de *Include* y *Extend* en nuestro Diagrama de Casos de Uso inicial, estamos en condiciones de “especificar” todos y cada uno de los Casos de Uso refinados. Ahora se llama Diagrama de Casos de Uso o Diagrama de Casos de Uso refinado.

Para ello hacemos uso del **Diagrama de Actividades**, usando la notación que propone **UML** para su elaboración. A manera de ejemplo se muestra el Diagrama de Actividades del Caso de Uso refinado:

Registro de materia:



Fuente: elaboración propia

Nota:

- Siempre se inicia el diagrama que representa al Caso de Uso refinado del lado del actor.
- Siempre se cierra del lado del sistema.
- Las "opciones" de decisión deben estar especificadas sobre los arcos de salida y no dentro del "rombo" (nodo de control tipo decisión).
- Debe contemplar todos los escenarios de uso del Caso de Uso, es decir, los pasos que se seguirán si todo va bien y los que no si ocurre algún inconveniente (se les conoce como flujos alternos). En el ejemplo se consideraron dos flujos "alternos": 1) cuando no se introducen los datos correctamente y 2) cuando se intenta registrar una materia que ya está registrada.
- Note también que se indica lo que debe hacer la app.

El enfoque clásico para especificar requisitos es mucho más exigente en su formulación: no se queda solo con concretar los pasos del Caso de Uso (su especificación), es decir, con el Diagrama de Actividades por cada Caso de Uso refinado, sino que exige todavía más detalles. Por ello, en este punto vamos a aprender a especificar **requisitos funcionales** según este enfoque clásico.



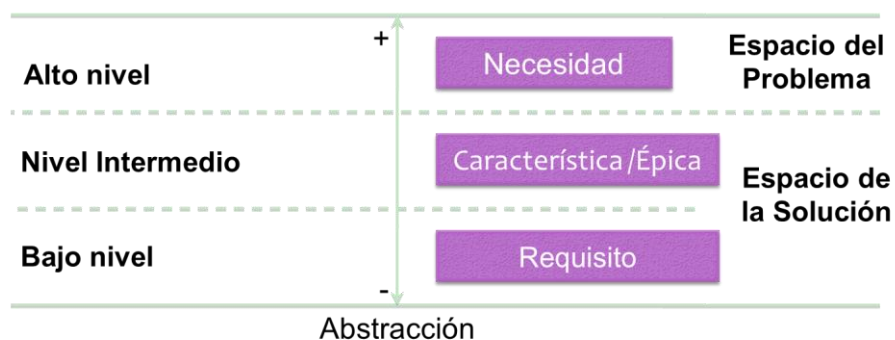
Comenzamos con la definición de "requisito":

Un requisito de *software* es la especificación detallada de algo que debe hacer el sistema para cumplir con una capacidad del sistema (característica/épica, o feature en inglés). Es una capacidad (funcional o no) que el sistema debe tener detallada y está enlazada con al menos una **característica/épica**.

Por ejemplo:

- Insertar datos de un producto
- Eliminar datos de un producto
- Modificar datos de un producto
- Consultar datos de un producto
- En ambiente web, con los navegadores Chrome, Firefox, Safari.
- Tiempo de respuesta de 20 segundos, etc.

Representan un menor nivel de abstracción (tabla 2):



Fuente: elaboración propia

Ahora bien, hay dos tipos de requisitos:

- Funcionales.
- No funcionales.



Requisitos funcionales:

- Definen las funciones que el sistema será capaz de realizar.
- Describen las **transformaciones** que el sistema realiza sobre las entradas para producir salidas.
- Definen qué hace el sistema (describen todas las entradas y salidas).



Requisitos no funcionales:

- Tienen que ver con características que de una u otra forma pueden **limitar el sistema** como, por ejemplo, el rendimiento (en tiempo y espacio), interfaces de usuario, fiabilidad (robustez del sistema, disponibilidad de equipo), mantenimiento, seguridad, portabilidad, estándares, etc.
- Definen los atributos que le indican al sistema cómo realizar su trabajo en términos de eficiencia, *hardware*, *software*, *interface*, usabilidad, etc.; es el cómo y cuándo.
- Propiciar calidad.

¡Marcan la diferencia!

La especificación de un requisito maduro cumple con un conjunto de atributos:

- **Necesario:** es necesario si su omisión provoca una deficiencia en el sistema a construir, y además su capacidad, características físicas o factor de calidad, no pueden ser reemplazados por otras capacidades del producto o del proceso.
- **Conciso:** es conciso si es fácil de leer y entender. Su redacción debe ser simple y clara para aquellos que vayan a desarrollar o a consultarlo en un futuro.
- **Consistente:** es consistente si no es contradictorio con otro requisito.
- **No ambiguo:** no es ambiguo cuando tiene una sola interpretación. El lenguaje usado en su definición no debe causar confusiones al lector.
- **Verificable:** es verificable cuando puede ser probado de manera que permita hacer uso de los siguientes métodos de verificación: inspección, análisis, demostración o pruebas.



Para cumplir con todos estos atributos, al especificar un requisito funcional debemos precisar tres aspectos de él:

- 1 **Elaborar una frase declarativa:** esta debe ser sencilla y precisa sobre lo que debe hacer el sistema, mas no cómo debe hacerlo. Ejemplo: el sistema debe registrar los datos del producto.
- 2 **Identificar las reglas de negocio asociadas al requisito:** ejemplo: el titular de la cuenta debe ser mayor de edad. La dirección de envío puede ser diferente a la dirección de facturación. El correo debe ser el institucional
- 3 **Interfaces hombre-máquina:** cómo quiere el usuario que se “vea” el requisito.

Fuente: elaboración propia

Estructura de datos (tabla 3):

Nombre	Descripción	T	L	RVP	VP D	I/ O	R/ O	Validaciones

Diagram illustrating the data structure (tabla 3) with annotations:

- Nombre del Campo:** Points to the 'Nombre' column header.
- Tipo:** Points to the 'T' column header.
- Rango de Valores:** Points to the 'VP D' column header.
- Entrada/Salida:** Points to the 'I/O' column header.
- Breve descripción:** Points to the 'Descripción' column header.
- Longitud:** Points to the 'L' column header.
- Valor por Default:** Points to the 'RVP' column header.
- Requerido/Opcional:** Points to the 'R/O' column header.

Fuente: elaboración propia

La especificación de requisitos no funcionales, de la que hablaremos aquí, se utiliza en los dos enfoques: clásico y ágil. Esto se hará nuevamente con una **frase declarativa**, pero más extensa.

Primero vamos a precisar mejor qué es un requisito no funcional.

Algunos de ellos son:



Requisitos de calidad:

- Usabilidad
- Interoperabilidad
- Confiabilidad
- Eficiencia
- Seguridad.



Requisitos legales y regulatorios: Por ejemplo, toda transacción bancaria debe registrarse por el Código de Comercio.



Limitaciones al diseño:

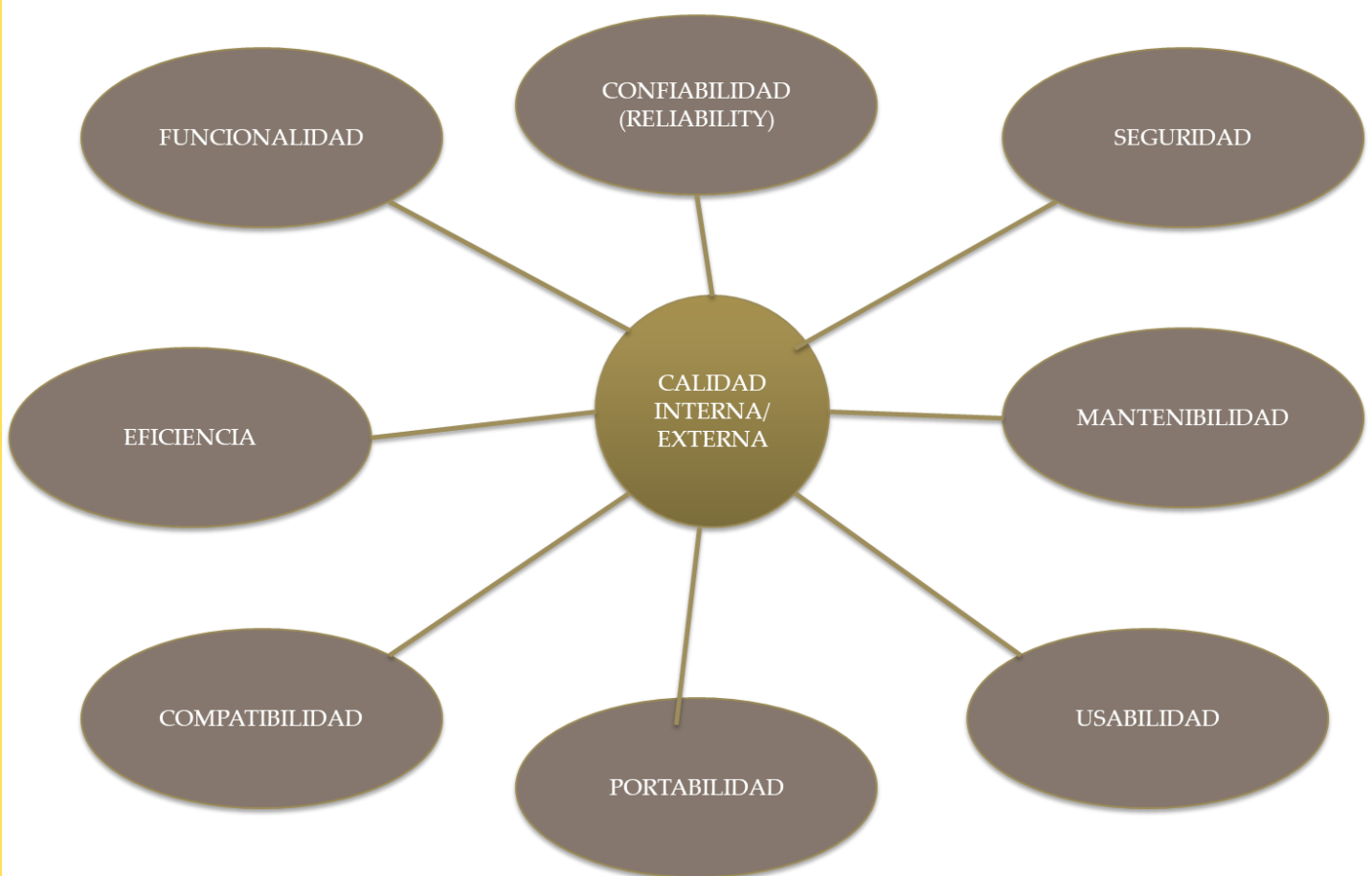
- Sistemas operativos
- Ambientes
- Compatibilidad
- Estándares aplicativos.

Algunos autores también llaman a estos requisitos como "suplementarios".

Con los llamados de **requisitos de calidad**:

- Para evitar confusiones, es preferible sustentarse con el estándar **ISO25010**.
- Este estándar define todas las características y subcaracterísticas que debe cumplir un software de calidad y da las definiciones de estas características.
- ¡Como es un estándar, nadie las discute!

Las características de calidad, según este estándar, son:



Fuente: elaboración propia

Sus definiciones se consiguen fácilmente en internet. Estos **requisitos no funcionales** son muy difíciles de identificar porque por regla general las personas del área del negocio no los conocen bajo dicha terminología.

Comunican:

- ¡Tenemos poco dinero y poco tiempo!
- ¡Este producto va a ser una ventaja competitiva!
- ¡Hay que entregarlo ya!
- ¡Debe ser muy simple!
- Este producto es o se va a fusionar con...
- ¡Vamos a extendernos a nuevas sucursales!
- ¡Tenemos temporadas pico!
- ¡Se necesitaba para ayer!
- ¡Debe ser ultrarrápido!
- ¡Tiempo de inactividad cero!

La gravedad de estas afirmaciones viene al generar un impacto muy fuerte en el **diseño**, pues si no las identificamos a tiempo, el *software* que se genere tendrá muchos problemas de seguridad, rendimiento, escalabilidad, etc.



Estos requisitos se transformarán en lo que se conoce como cualidades de la arquitectura.

La siguiente tabla (4) nos orienta a cómo precisarlos un poco mejor:

Comentario	Caso de Uso
Ventaja competitiva	Agilidad, capacidad de prueba, implementabilidad, escalabilidad, disponibilidad, tolerancia a fallas.
Fusión y nuevas sucursales	Interoperabilidad, escalabilidad, adaptabilidad, extensibilidad.
Time to Market	Agilidad, fácil de probar y rápidamente desplegable.
Poco tiempo y/o presupuesto	Sencillez, viabilidad.
Satisfacción del usuario	Rendimiento, disponibilidad, tolerancia a fallas, capacidad de prueba, capacidad de implementación, agilidad, seguridad.

Fuente: Cómo precisar cualidades de la arquitectura. Traducido de Richard y Ford (2020)

- Es imposible identificarlos si el equipo **no trabaja colaborativamente**.
- Por ello, existen técnicas tales como Kata (<http://nealford.com/katas/>).
- Las especificaciones de los requisitos no funcionales o suplementarios las registraremos en el **ERS**.

04 Historias de usuario

Para cerrar el tema, falta especificar los requisitos funcionales según el enfoque ágil.

En el *BRIEF*, además de precisar el mapa de impacto, elaboraste con tu equipo el mapa de historias de usuario. Ahora debemos ser más específicos al determinar de qué tratan cada una de las historias de usuario.

- Una historia de usuario se comienza a especificar cuándo comienza el Sprint.
- Nuevamente, al igual que en el caso de los Casos de Uso, se hace colaborativamente.
- Se propone precisar las "3 C" de la historia.

Card

Una descripción muy breve como para que quepa en una tarjeta. Con la siguiente estructura:

- **Como:** <Persona que expresa una necesidad>
- **Quiero/necesito:** la necesidad requerida
- **Para:** valor obtenido, contexto de la necesidad.

Ejemplos:

- Yo como **estudiante** quiero **inscribirme** en las materias que me corresponden para cursarlas en **este semestre**.
- Yo como **cliente** quiero **conocer** la descripción del producto para decidirme a **comprarlo**.

Conversación

Es una oportunidad para el diálogo entre quien expresa la necesidad y quien la satisface.

- Esta "C" requiere de un importante nivel de detalle para ser implementada.

Criterios de aceptación

La historia debe contener un conjunto de elementos necesarios para determinar que se ha entregado lo solicitado.

Como la conversación requiere más detalles, utilizaremos otra plantilla para su realización, propuesta por Salazar (www.gazafatonarioit.com):

The User Story Conversation Canvas

Lucho Salazar (@luchosalazarc)


2. Historia de Usuario	1. Personas	4. Contexto	
3. Criterios de Aceptación		5. Definición de Preparado	6. Definición de Terminado
7. Resultado Esperado		8. Métricas	9. Retroalimentación

Fuente: The User Story Conversation Canvas. Salazar (2017)

Aquí tenemos un ejemplo:

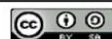
The User Story Conversation Canvas

[Lucho Salazar \(@luchosalazarc\)](#)

2. Historia de Usuario Como Suscriptora Premium Quiero Tener la oportunidad de acceder primero a los libros de mi interés Para Ser una de las primeras lectoras y escribir y publicar sobre el libro.	1. Personas <div style="text-align: center;">  Sandra Dee </div> <p>Edad: 54 Lugar: Cartagena Educación: Medica – Pediatra Trabajo: varias clínicas y hospitales Familia: vive con su hijo</p> <p>Metas Descubrir nuevos autores y libros Encontrar historias únicas Catalogar su colección de libros Publicar un libro de ensayos</p> <p>Frustraciones Mantenerse al día con las series Olvidar fechas de lanzamiento Falta de tiempo para buscar y leer</p> <p>Hábitos En el día pasa mucho en el transporte Gusta de cajas de colección Siempre termina un libro Lee e-libros pero prefiere físicos</p> <p><i>¡Quédate en casa, salva vidas!</i></p>	4. Contexto <ul style="list-style-type: none"> Sandra tiene poco tiempo para buscar nuevos libros En un dispositivo móvil, mientras espera transporte, tarde en la noche, mientras hace pausa en el trabajo Alcance: solo primeras ediciones Solo de proveedores registrados Sería bueno conocer mejores precios de venta y costos de envío desde distintas fuentes
3. Criterios de Aceptación <ul style="list-style-type: none"> <input type="checkbox"/> Recibir alertas frecuentes sobre nuevas publicaciones con una semana de anticipación <input type="checkbox"/> Quiero decidir sobre la frecuencia de alertas <input type="checkbox"/> Incluir los detalles del libro: autor, tema, bio, enlaces a notas de prensa, expectativa mundial, Editor(ial), si habrá o no evento de lanzamiento. <input type="checkbox"/> Quiero saber si estará disponible en mi ciudad <input type="checkbox"/> Saber dónde puedo adquirirlo 		5. Definición de Preparado Se debe establecer si se quiere información específica del editor o solo de la editorial. ¿Cómo clasificar la confiabilidad de las fuentes de información? Tener definidos los accesos a la información vía Google.
7. Resultado Esperado Tener acceso a información relevante y privilegiada sobre el lanzamiento del libro o colección. Información de contacto con el autor o editor. Formatos disponibles. Posibilidad de hacer la compra con anticipación. Información de contacto de otros interesados en la publicación.	8. Métricas <ul style="list-style-type: none"> # de solicitudes de preventa Calificación anticipada de la publicación (Ranking de interés) 	6. Definición de Terminado Verificar la información de la publicación con al menos tres editoriales. Aprobada la verificación de accesibilidad de Google.
		9. Retroalimentación Nivel o Grado de Satisfacción de las personas Nivel de Felicidad del usuario Impacto generado

www.gazafatonarioit.com
 Versión 1.0.5.20170505

This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](#)



Fuente: Ejemplo: The User Story Conversation Canvas. Salazar, L. (2017)

Veamos punto por punto.

Persona: obviamente aquí va la "persona" afectada por la historia o su protagonista. Por ejemplo: estudiante. Como vieron en la lámina anterior, se puede escribir mucho detalle sobre él. Yo propongo solo lo relevante para efectos del desarrollo de la historia.

Historia: aquí se escribe la primera "C". En nuestro ejemplo:

- Yo como **estudiante**
- quiero **inscribirme** en las materias que me corresponden
- Para cursarlas en **este semestre**.

Criterios de aceptación: ¿cuáles características debe tener la historia construida para que el usuario y el equipo de desarrollo dé su ok? En nuestro ejemplo:

- Queda registrada la inscripción del estudiante correctamente.
- El estudiante puede imprimir o tiene en pdf su comprobante de inscripción.
- Se registraron los cambios en los cupos de las materias correspondientes, etc.

Contexto: describe el entorno en el cual se ejecuta la historia. En nuestro ejemplo:

- El estudiante está en su casa o desde su móvil
- Con conexión de internet muy débil
- Puede no querer imprimir el comprobante
- Debe ser muy amigable la interfaz de selección de usuarios
- Solo deben aparecer las materias que él puede cursar (es decir, que él haya cumplido con los prerrequisitos)
- Si no hay cupo en un horario seleccionado, debe mostrarle las otras opciones
- Es parte de la Épica/Entregable **Proceso de Inscripción**
- El estudiante se debió loguear.

Definición de preparado: el equipo necesita saber que la historia está lista para ser desarrollada en el sprint correspondiente. Debe cumplir con las características antes descritas: independiente, negociable, valiosa, estimable, pequeña y probable (testeable). En nuestro ejemplo:

- **Independiente:** lo será solo cuando en la base de datos (BD) se hayan registrado los prerequisites cubiertos por el estudiante para cada materia.
- **Negociable:** ya descrita en el Card (punto 2).
- **Valiosa:** la universidad requiere que el estudiante no necesite trasladarse para inscribirse.
- **Estimable:** el equipo se siente seguro de que esta historia es solo de dos puntos de esfuerzo.
- **Pequeña:** se puede desarrollar en tres semanas.
- **Probable:** a partir de ella se pueden formular casos de prueba que verifiquen su funcionamiento correcto.

Definición de terminado: el equipo debe estar de acuerdo en cuándo se puede considerar que ya se terminó de construir la historia. En nuestro ejemplo:

- Casos de prueba, todo ok.
- Casos de prueba de aceptación, ok.
- Documentación completa.

Resultado esperado: establecer explícitamente lo que se quiere. En nuestro ejemplo:

- La inscripción quedó registrada correctamente en la base de datos.
- El estudiante puede imprimir o tiene en pdf su comprobante de inscripción.
- Puede hacer la inscripción desde su PC, laptop o móvil.

Métricas: la medición ayuda a gestionar y a optimizar el trabajo. En nuestro ejemplo:

- Número de casos de prueba exitosos/fallidos.
- Tiempo estimado versus tiempo ejecutado.
- Punto de historia estimado versus puntos de historia ejecutados.

Retroalimentación: las organizaciones inteligentes son las que aprenden. Hay que gestionar el conocimiento, es decir, registrar las lecciones aprendidas. En nuestro ejemplo:

- ¿Por qué nos fue bien?
- ¿Por qué nos fue mal?
- ¿Cuáles fueron los casos de prueba fallidos? ¿Por qué fallaron?
- ¿A qué se debió el retraso?
- ¿Qué ocurrió que no estaba previsto? ¿Por qué?

A pesar de toda esta precisión, puede darse el caso de que necesitemos ser más concretos en la descripción de este diálogo (la conversación). Sobre todo si hay que aplicar muchas reglas de negocio.

Una herramienta muy poderosa para ello es, de nuevo, el **Diagrama de Actividades**.

En algunas ocasiones sugiero “enlazar” al punto 4 “Contexto” con un Diagrama de Actividades que explique mejor cómo queremos que fluya la historia; sobre todo es clave identificar esos flujos alternos.

Las especificaciones de las historia de usuario las registraremos en el ERS.



Este tema es intenso, pues es la concreción de lo que se va a construir y estamos aprendiendo bajo dos enfoques.

Clásico:

- El diagrama de Casos de Uso inicial no nos da esta concreción; para ello lo refinamos, usando relaciones de *Include* y de *Extend*. Por cada Caso de Uso refinado elaboramos su especificación, apoyándonos en un Diagrama de Actividades.
- Cada requisito funcional lo especificamos usando cuatro elementos: frase declarativa, reglas de negocio, estructura de datos y sus interfaces hombre-máquina.

Ágil:

- Especificamos las historias de usuario, identificadas en el BRIEF y en el Mapa de historias de usuario, utilizando la plantilla de Salazar (2017).
- En ambos se dieron algunos lineamientos de cómo especificar los requisitos no funcionales o suplementarios.

La especificación de los requisitos funcionales bajo un enfoque ágil (historias de usuario) y de los no funcionales de nuestro proyecto, los registraremos en el ERS.



Referencias de las imágenes

Richard, M. y Ford, N. (2020). Cómo precisar cualidades de la arquitectura [Imagen]. Recuperado de *Fundamentals of Software Architecture: An Engineering Approach*. O'Reilly Inc.

Salazar, L. (12 de mayo 2017). Ejemplo: The User Story Conversation Canvas [Imagen]. Recuperado de *The User Story Conversation Canvas*. Gazafatonario IT. <http://www.gazafatonarioit.com/2017/05/the-user-story-conversation-canvas.html>

Salazar, L. (12 de mayo 2017). The User Story Conversation Canvas [Imagen]. Recuperado de *The User Story Conversation Canvas*. Gazafatonario IT. <http://www.gazafatonarioit.com/2017/05/the-user-story-conversation-canvas.html>

Ponsot, E. (s.f.). Ejemplo de requisito funcional [Imagen]. Recuperado de <https://www.researchgate.net/profile/Ernesto-Ponsot-Balaguer>

Has culminado la revisión del tema