

## Concepto de funcionalidad completa de un sistema

Ivar Jacobson – modeló los sistemas pensando en cómo se mueven en el teatro: Actores principales y secundarios, escenarios.

Los **casos de uso** son una técnica para analizar y especificar el comportamiento de un sistema o de una parte de este describiendo qué hace el sistema, pero no cómo lo hace (los **requisitos funcionales**).

Podemos decir entonces que un **caso de uso** es una forma de expresar como “algo o alguien” externo al sistema interactúa con el mismo usando dichas funcionalidades y obteniendo un resultado de interés para sí mismo. Los sistemas no solo son usados por personas, sino también por otros sistemas de hardware y/o software. En cualquier caso, llamaremos “actor” a ese “algo o alguien” que interactúa con el sistema. **Sólo es actor aquel que interactúa con el sistema.**

Los actores siempre son externos al sistema, por lo tanto, al identificarlos se comienza a definir el límite y el alcance de este.

Existe una diferencia entre el actor y el usuario, el actor representa un rol y el usuario es una persona o grupo de personas que cuando usa el sistema asume un rol, por lo tanto, puede ocurrir que una misma persona (usuario) en distintos momentos asuma roles diferentes y utilice el sistema de acuerdo con el rol asumido.

**Actor principal:** Quién inicia el caso de uso

**Actor secundario:** Es aquel actor que si no interviene no permite al actor principal obtener su beneficio.

Hoy se usa como parte del ciclo de vida de desarrollo del sistema como en el modelado ágil.

Diferencia con modelo de dominio: el modelo de dominio es desde el punto de vista del sistema.

CARACTERÍSTICAS
<ul style="list-style-type: none"><li>• Están expresados desde el punto de vista del actor y no del sistema.</li><li>• Se documentan con texto informal, en primera instancia, el documento que detalla la especificación de los casos de uso va a estar dirigido al cliente.</li><li>• Describen tanto lo que hace el sistema como lo que hace el actor cuando interactúa con él.</li><li>• Son iniciados por un único actor a la vez, un mismo caso de uso puede ser iniciado por más de un actor en momentos diferentes, pero no puede haber casos de uso que no sean iniciados por ningún actor</li><li>• Están acotados a una sola funcionalidad completa del sistema, una funcionalidad completa es un caso de uso si se debe indicar explícitamente al sistema que se quiere acceder a dicha funcionalidad.</li><li>• Los casos de uso deben dejarle un beneficio o resultado de interés al actor que lo inició (no siempre es un producto)</li></ul>



Un CDU habitualmente involucra varios requisitos funcionales.

No hay una relación directa que un requisito funcional es un caso de uso y un caso de uso de un requisito funcional. Porque por ahí puede ser un requisito funcional muy complejo que implique ser un caso de uso o puede ser un requisito funcional tan simple como “diga informar cantidad”, pero ese informar cantidad no es un CDU sino un paso.

Los pasos a seguir para obtener el diagrama de CDUs y descripción general de cada uno

1. Identificar los actores
2. Identificar los casos de uso para cada actor
3. Identificar nuevos casos de uso a partir de los existentes:
  - 3.1. Variaciones significativas: casos de uso de igual nombre, que se ejecutan de maneras diferentes (Ej.: comprar por internet o en persona). (Ej.: si hay diferencias entre registrar la venta a un cliente minorista y a uno mayorista, entonces, del caso Registrar Venta se podrían obtener los casos Registrar venta a cliente minorista y Registrar venta a cliente mayorista, si las variaciones del caso de uso lo justifican.)
  - 3.2. Casos de uso opuestos
  - 3.3. Casos de uso que preceden y que suceden
4. Describir los casos de uso con trazo grueso: breve descripción desde la perspectiva del actor. [Es muy importante que el analista de sistemas, junto con su equipo o, revisen bien este modelo antes de presentárselo al cliente]
5. Priorizar los CDU
  - 5.1. Imprescindibles y de prioridad alta: sin ellos el sistema no funciona
  - 5.2. Importantes y de prioridad media: los que se pueden negociar con el cliente
  - 5.3. Deseables o de prioridad baja: posibles de implementar pero que podrían eliminarse

[Una vez categorizados, se selecciona el orden de iteración en general, se incluyen los de prioridad alta se discuten los de prioridad media y se descartan los de prioridad baja cuyo costo sea alto]

6. Describir los CDU con trazo fino: paso a paso desde la vista del actor
7. Identificar relaciones entre CDUs.

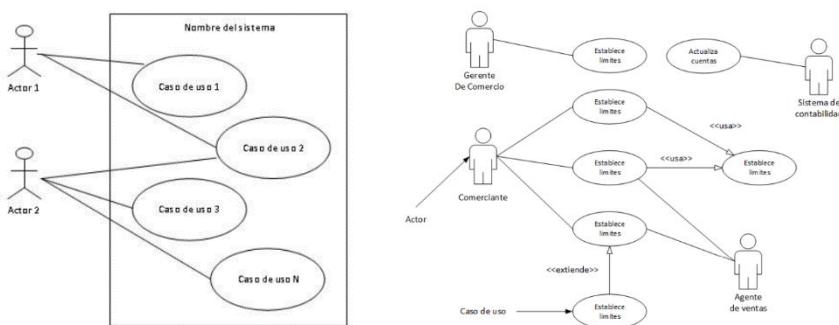
Con los casos de uso estamos viendo qué requieren los stakeholder. Estamos modelando los requisitos en un formato de CASO DE USO.

Esto sirve de forma:

EXTERNA → darle una documentación a quien nos convocó de lo que entendemos.

INTERNO → En etapa de análisis, pasársela al área de diseño para que empiecen a bosquejar las interfaces.

## Diagrama de casos de uso del UML



El rectángulo es el límite del sistema

Los Stickman son los actores (**Vamos a modelar solamente el actor principal en esta cátedra, no vamos a graficar al actor secundario**)

Los óvalos son los nombres de los CDU

## La especificación esencial de casos de uso

Tienen una interfaz tentativa MOCKUP para el CDU para identificar los datos que se reciben. No es la interfaz de SW.

- **Nombre:** verbo en infinitivo + sustantivo (DEVOLVER LIBRO, PRESTAR LIBRO, PAGAR VIAJE EN COLECTIVO)
- **Tipo:**
  - Base: es un CDU base para ejecutar varias tareas
  - Incluido: Ej.: *COMPRO EN INTERNET* → *PAGO CON TARJETA* (*incluido obligatoriamente*)
  - Extendido: cuando es por mi decisión que se accede a otro CDU / le agrega valor al caso base. Ej.: *cuando accedo a las diferentes secciones comprando una entrada*
- **Descripción general:** 3 o 4 renglones donde dice quién es el actor, el beneficio y que otra cosa le permite (Ej.: *sanción si no devuelve el libro*), narración genérica del caso de uso.
- **Actores principales:** Ej.: en una venta, puedo tener al vendedor, jefe de ventas o supervisor, etc. Puede haber más pero siempre lo inicia uno solo.
- **Actores secundarios:** Lo necesito interactuando en el CDU para que el actor principal obtenga el beneficio.
- **Autor**
- **Fecha de creación**
- **Precondición:** incluye aquel o aquellos casos de uso que deban haberse ejecutado previamente, al menos una vez, para que el caso de uso que quiero ejecutar sea exitoso y me permita acceder al beneficio. Puede haber una, varias o ninguna, es muy difícil encontrar un caso de uso sin precondición.
- **Puntos de extensión:** en el caso de que sea extendido, se aclara acá Ej.: “ver ubicaciones de entradas”
- **Descripción detallada:**
  - Flujo Normal: 1 el actor hace, 2 el sistema responde, 3 el actor hace... si en algún momento se incluye un CDU se menciona “Se incluye CDU XXXXX”. Se especifica también “Fin de caso de uso”.

- **Flujo alternativo:** Si se dan situaciones que no permiten que obtengas el beneficio se detallan acá.  
Ej.: no es alumno, o es alumno, pero está sancionado, no hay copia, es el único ejemplar y no se puede retirar. Impiden RETIRAR LIBRO. Se separan con número y nombre.: A0, A1, A2.
- **Postcondición:** Beneficio para el actor principal y descripción del estado del sistema después de la ejecución.  
Ej.: para el caso de uso REGISTRAR VIAJE EN COLECTIVO → VIAJE EN COLECTIVO REGISTRADO.  
Ej.: en INSCRIPCIÓN A MATERIA → EL ALUMNO ESTÁ INSCRIPTO (y queda una vacante menos en el sistema).

## El análisis de los requisitos: La verificación versus la validación: naturaleza y actores.

Calidad: cuando hablamos de validez y verificar hablamos de calidad.

Concepto de calidad: naturaleza del producto.

Satisfacer al cliente. Cuando alcanzamos la calidad alcanzamos lo que el cliente pagó.

### Gestión de calidad:

- **Modelo clásico:** control de calidad -QC- (esfuerzo determinado, hacer la producción y después probarla)

Ej.: fábrica de tornillos. Compro material, contratos obreros, a fin de mes tengo un depósito de un millón de tornillos.

Los analizo (peso, mido, etc.), los que están bien los guardo en una caja, los que están mal los guardo en otra caja. Descarto los no aptos de la muestra. Se puede tomar una muestra.

Ej.: si nos tomasen una sola prueba al final de la universidad, en vez de tomar todas las pruebas posibles entre jardín y universidad, nos ahorramos tiempo.

- **Quality Assurance (QA):** permanentemente monitoreando/verificando fase del proceso tratando de detectar un error.


Cuando hablamos de **ing. De requisitos** hablamos de técnicas de verificación y validación, es decir que hay cosas que nosotros podemos hacer antes de entregar, para verificar si estamos construyendo bien nuestro producto, y segundo es validar con el cliente si está correcto.

**Validación:** si es el sistema correcto → Lo hace el cliente

**Verificación:** si estamos construyendo el sistema correctamente → Lo hace el equipo de desarrollo

Siendo del equipo de desarrollo puede haber un especialista en interpretar al cliente cuando éste no puede darse a entender, son nuevos roles que nacen con las metodologías ágiles (SCRUM).

Técnicas de validación (Sommerville)	
<b>Revisión de requerimientos</b>	Los requerimientos se analizan sistemáticamente usando un equipo de revisores que verifican errores e inconsistencias.
<b>Creación de prototipos</b>	En esta aproximación se muestra un modelo ejecutable del sistema en cuestión a los usuarios finales y clientes. Así ellos podrán experimentar con este modelo para constatar si cubre necesidades reales.
<b>Generación de casos de prueba (los crea el cliente)</b>	<ul style="list-style-type: none"> <li>• Los req. Deben ser comprobables.</li> <li>• Si las pruebas para los req. Se diseñan como parte de la validación, esto revela con frecuencia problemas en los requerimientos.</li> <li>• Si una prueba es difícil o imposible de diseñar, esto generalmente significa que los requerimientos serán difíciles de implementar, por lo que deberían reconsiderarse.</li> <li>• El desarrollo de pruebas a partir de los req. Del usuario antes de escribir cualquier código es una pieza integral de la programación extrema.</li> </ul>



**Lista de verificación para validar requerimientos**

Con frecuencia es útil analizar cada requerimiento en comparación con preguntas de verificación. A continuación se presentan algunas:

- ¿Los requerimientos están enunciados con claridad? ¿Podrían interpretarse mal?
- ¿Está identificada la fuente del requerimiento (por ejemplo, una persona, reglamento o documento)? ¿Se ha estudiado el planteamiento final del requerimiento en comparación con la fuente original?
- ¿El requerimiento está acotado en términos cuantitativos?
- ¿Qué otros requerimientos se relacionan con éste? ¿Están comparados con claridad por medio de una matriz de referencia cruzada u otro mecanismo?

**INFORMACIÓN**

- ¿El requerimiento viola algunas restricciones del dominio?
- ¿Puede someterse a prueba el requerimiento? Si es así, ¿es posible especificar las pruebas (en ocasiones se denominan criterios de validación) para ensayar el requerimiento?
- ¿Puede rastrearse el requerimiento hasta cualquier modelo del sistema que se haya creado?
- ¿Es posible seguir el requerimiento hasta los objetivos del sistema o producto?
- ¿La especificación está estructurada en forma que lleva a entenderlo con facilidad, con referencias y traducción fáciles a productos del trabajo más técnicos?
- ¿Se ha creado un índice para la especificación?
- ¿Están enunciadas con claridad las asociaciones de los requerimientos con las características de rendimiento, comportamiento y operación? ¿Cuáles requerimientos parecen ser implícitos?

Comentarios sobre los puntos de la imagen:

- Al riesgo que algo se malinterprete se lo llama ambigüedad.
- Tenemos un requisito ¿Tenemos la posibilidad de saber cómo se generó ese requisito?
- El requisito está acotado en términos cuantitativos significa: que sea medible.

- A las Relaciones entre requerimientos los vemos en el diagrama de casos de uso.
- Violación del dominio: si se sale del dominio el cliente no puede especificar los casos de prueba.
- Se debe definir cuál va a ser la forma de medirlo.
- ¿La especificación del caso de uso coincide con el diagrama?

**Negociación:** normalmente se va consensuando con el cliente el grado de satisfacción.

## Historias de Usuario

Es una corta declaración de intención que describe algo que el sistema necesita hacer para el usuario. Se realizan sobre funcionalidades parciales. Y se usan cuando hay requisitos cambiantes. Se utilizan en el desarrollo ágil.

INVEST
Independiente: no requiere de otra
Negociable: se puede reemplazar por otra de diferente prioridad
Valor: que sea necesaria y de valor para el proyecto
Estimable: que el equipo se sienta tranquilo y seguro estimándola.
pequeñas : que no sean grandes, funcionalidades pequeñas
Verificable: que se le puedan realizar pruebas. O que sea comprobable.

Las historias de usuario **no son requisitos**, son más bien una carta de intención de lo que queremos que haga el sistema, son recordatorios para conversaciones que tendremos más adelante

*Como <rol> quiero/deseo/necesito <funcionalidad> para <beneficio de negocio>*

SMART	
SPECIFIC	específicas
MESURABLE	medibles
ACHIEVABLE	alcanzables
RELEVANT	relevantes
TIME-BOUNDED	limitadas en el tiempo

Tienen un **CONTEXTO**, que es el que te da información de las condiciones que desencadenan al escenario.

Tienen **CRITERIOS DE ACEPTACION (o CONFIRMACIÓN)**, es decir, la validación de requisitos del cliente con su equipo de análisis. Es a través de los criterios de aceptación que establecemos los límites de la historia de usuario. En parte, los usamos para confirmar la completitud y la precisión o conformidad de la historia con lo que quieren los usuarios.

Se escriben en un lenguaje simple.

En el fondo se trata de una lista de condiciones de satisfacción que todos los involucrados tendremos en cuenta al momento de diseñar, construir, probar y entregar el producto resultante. Los criterios de aceptación nos permiten reconocer cuándo la historia está Terminada (o criterio de Done de Scrum).

Representan las condiciones de satisfacción que se aplicarán para determinar si la historia cumple o no la intención, así como los requisitos más detallados. Es precisamente la Prueba de Aceptación del usuario, que muestra cómo el usuario confirmará que la historia se ha implementado a su entera satisfacción.

Los flujos alternativos en la actividad, los límites de aceptación y otras clarificaciones deberían capturarse junto con la historia.

Muchos de estos se pueden convertir en casos de pruebas de aceptación u otros casos de pruebas funcionales, para la historia.

LAS 3 C
CARD: Debe entrar en una CARD
CONVERSACIÓN: Conversación: conjunto de aclaraciones realizada por PROD OWNER
CONFIRMACIÓN: Criterios de Aceptación

Si quedó un escenario por contemplar entra como una nueva HDU al backlog.

<b>Si una historia de usuario esta "DONE" significa que:</b>
Fue probada y certificada por el equipo
Fue aceptada por el Dueño de Producto
Si llegase a existir un "BUG", con plena seguridad afirmo que fue un escenario no identificado y por lo tanto una historia nueva debe implementarse.

Las historias de usuario tienen dentro de sus **objetivos**:

>>Sincronizar las expectativas del Dueño de Producto o usuario con el equipo respecto a una funcionalidad

>>Servir como elemento que dirigirá la construcción del producto de software

#### DADO - Y - CUANDO - ENTONCES

Criterios de aceptación (se pueden manejar dos opciones) Criterios de aceptación en Prosa	Criterios de Aceptación en Formato BDD GIVEN – DADO WHEN – CUANDO THEN – ENTONCES (Este forma ayuda a que no queden cabos sueltos en la historia)
Que solicite lo datos de la empresa	<b>Escenario 1: Solicitar Datos de la empresa</b> <b>DADO</b> que me encuentro en la página de cliente <b>Y</b> se haya registrado al cliente como empleado <b>CUANDO</b> se soliciten los datos de la empresa <b>ENTONCES</b> se pedirá el nombre de la empresa.
Que solicite el NIT (Número de Identificación Tributaria) para las empresas) y lo valide	<b>Escenario 2: Solicitud de NIT</b> <b>DADO</b> que me encuentro en la página de cliente <b>Y</b> se haya registrado al cliente como empleado <b>CUANDO</b> solicite la información de NIT <b>ENTONCES</b> se verificará el número del NIT que su estructura sea correcta.

La historia de usuario debe ser de un tamaño tal que a lo sumo tome 3 días una persona construirla, probarla (quedando con cero errores) y desplegarla en el ambiente de pruebas (o el ambiente indicado), de manera que el equipo de trabajo pueda decir tranquilamente y sin ningún temor "¡ESTA LISTA PARA PRODUCCIÓN!"

-----

#### Características de una buena historia de usuario

Luego de hacer la CCC (Carta, Conversación, Confirmación), definitivamente el criterio más importante que usamos es este de INVEST. Los ingredientes clave de una historia de usuario son: quién es el usuario, qué quiere hacer el usuario y por qué lo necesita.

Si queda muy grande se pueden tener en cuenta varios patrones para realizar un **splitting** de la misma.

<b>Spike</b>	Experimento o una prueba de concepto técnico que nos sirve para encontrar alternativas de solución a un problema específico o a una situación de la cual no estamos seguros en el desarrollo del producto. Normalmente, representa incertidumbre y el objetivo es resolverla lo más pronto posible. Se establece un bloque de tiempo (time-box) para un spike de no más de un sprint, en lo posible de mucho menos. Nunca prometer la implementación de una historia y la realización de un spike del que la historia es dependiente, en el mismo sprint.
<b>Variaciones en la interfaz</b>	Hacer la interfaz simple y luego complejizar.
<b>Esfuerzo importante</b>	Por cuales empezar (priorizar según objetivos y complejidad)
<b>Simple-compleja</b>	Primero hacerlas simples y complejizar.

<b>Retrasar el rendimiento</b>	no retrasar req no funcionales, no dividir demasiado pronto, no dividir de más, no dividir por componente, no olvidar las pruebas de la H
<b>Variaciones por tipos de usuario</b>	¿Podrías dividir la historia para el usuario que permita agregar más valor, obtener mayor aprendizaje o retroalimentación efectiva, o aun para el de mejor riesgo?
<b>Variaciones por interesado</b>	¿Podrías dividir la historia para el interesado que aporta mayor valor y luego mejorar con los demás?
<b>Variaciones por navegador</b>	¿Tiene la historia el mismo comportamiento para varios navegadores de Internet?
<b>Variaciones por plataforma</b>	¿Podrías empezar construyendo la historia para la plataforma que proporciona mayor valor y luego implementarla para las restantes?
<b>Servicios externos</b>	¿Podrías empezar haciendo un “mock” del servicio para aislar la dependencia y simular su comportamiento? Luego podrías implementar la funcionalidad completa en otra Historia.
<b>Región geográfica</b>	¿Podrías empezar construyendo la historia para una región geográfica que te aporte más/mucho valor y luego extenderla al resto de regiones?
<b>Retrasa los comportamientos opcionales</b>	¿La historia incluye mucho comportamiento opcional (por ejemplo, distintas formas de lograr la misma meta)?
<b>Retrasa el comportamiento por condiciones de error</b>	¿Puedes hacer que este comportamiento por errores se implemente en historias subsiguientes luego de que el comportamiento “normal” se implemente?
<b>El mayor valor</b>	Pareto 80 valor por sobre 20% de la HDU
<b>Hasta acá llegamos</b>	Historia muy grande, backlog casi terminado

#### Esfuerzos sugeridos de HDU según duración de sprint

- Análisis
- Diseño
- Implementación
- Revisión Par (esta es una buena práctica)
- Pruebas
- Despliegue
- Corrección
- Actualización de documentación relevante para el equipo.

<b>PUNTOS CLAVE</b>
1. identificar y entender a los consumidores y usuarios
2. historia de usuario: identificar la carta, recordatorio de conversaciones y respetar la forma.
3. criterios de aceptación: establecemos limites, confirmamos completitud, precisión o conformidad. Se escriben en lenguaje simple.
4. Contexto: entender el entorno en el que existe la hdu.
5. definición de preparado
6. definición de terminado
7. resultado esperado
8. métricas: conocer urgencia nos habla de la prioridad de la misma
9. feedback