Requerimientos de Sistemas

Resumen
Cursada 2023

Parcial 1

Qué nos pueden pedir
En base a los casos trabajados

Requerimientos del Proyecto y Requerimientos del Producto

POR QUÉ se encara el proyecto, identificando el problema o problemas a resolver.
PARA QUÉ se encara el proyecto, identificando las intenciones del cliente.
QUIÉN es el cliente del producto.
QUÉ producto será el resultado del proyecto.

Por qué → el problema
Para qué → la intención *(principal)*
Quién → el que paga: *Cliente*
Qué → producto final

Requerimientos que impactan en el éxito del proyecto → derivan en ***premisas***

Requerimientos del Negocio

- Cuestionario para encuentro aclaratorio *(que permita especificar algo como objetivos del negocio, resolver inconsistencias, etc)*
  ¿Qué espera obtener con el sistema?
  ¿Cuánto espera achicar _un_valor_?
  ¿Por qué se produce actualmente?
  ¿Cómo puede ayudar el sistema?

- Antecedentes
  - Quiénes y cuántos trabajan/participan actualmente
  - Problema *(lo que se hace actualmente que da problemas)*
  - Procesos que se realizan actualmente
  - Cómo se realizan esos procesos actualmente
  - Formularios/planillas
  - Sistemas que se usan
- Oportunidades, necesidades e ideas
  - Va todo junto en un mismo texto
  - Necesidades → problema que quieren solucionar
  - Idea del producto que tiene el *cliente* / cómo solucionar problema
- Objetivos del negocio
  - Se tienen que poder medir

  Reducir/Mejorar/Lograr en/que un **X%** *_objetivo_* **luego de Y meses de la puesta en marcha del sistema**
  Ej:

- Reducir el número de vehículos, que se aceptaron y no pudieron ser atendidos, en un X%, en Y meses después de la puesta en marcha del sistema.
- Mejorar las estimaciones que hacen los jefes del taller mecánico para el ingreso de nuevos vehículos, en un X%, luego de Y meses de la puesta en marcha del sistema.
- Reducir en un 50% el tiempo invertido en la coordinación general de cada concurso, luego de Y meses de puesta en marcha el sistema.

Alcanzar/ un *_objetivo_* **de U unidades *(total/mensual)* luego de Y meses de la puesta en marcha del sistema**
Certificar *_objetivo_* **a partir del próximo período contable**

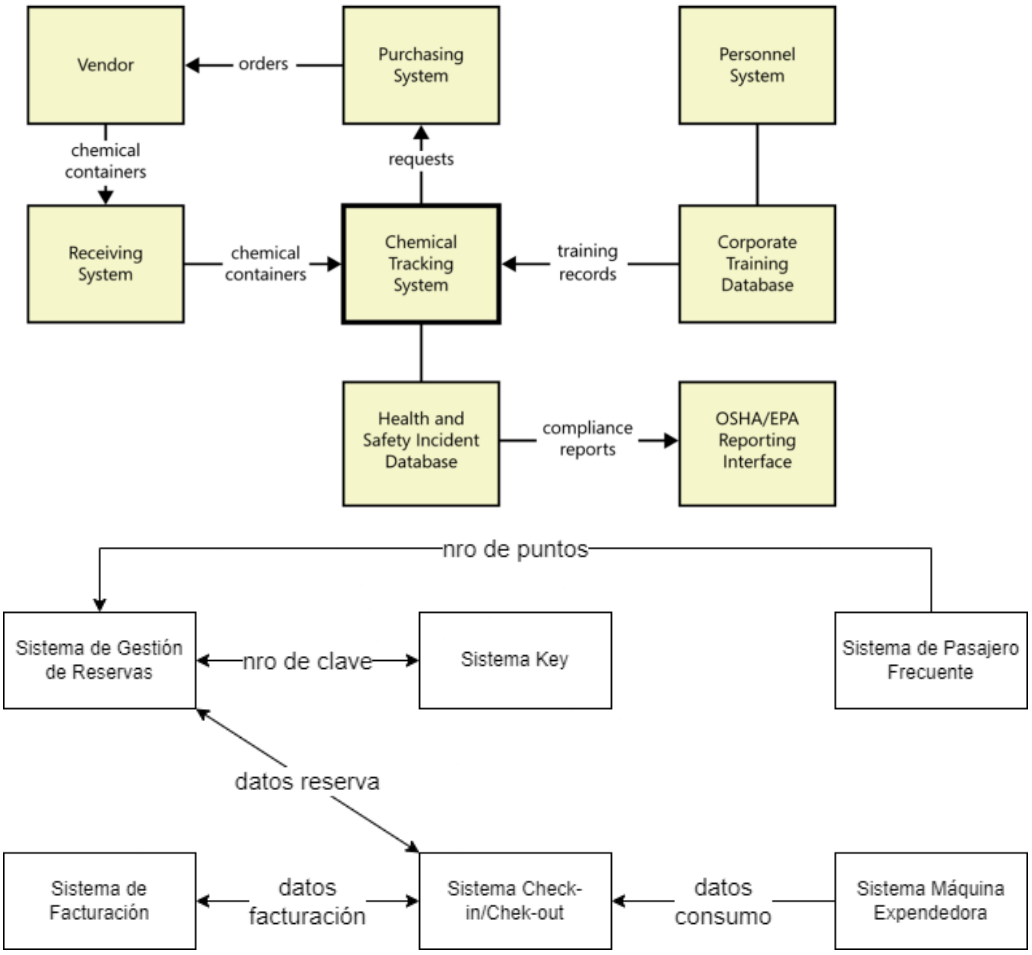
- Visión del producto

PARA *[cliente que demanda el producto]*
QUIEN *[necesidad, oportunidad o idea]*
EL *[nombre del producto]*
ES *[categoría del producto o tipo de proyecto]*
QUE *[características y cualidades distintivas]*
EN LUGAR DE *[sistema o proceso actual o producto de la competencia]*
EL NUEVO PRODUCTO *[ventajas competitivas o principales beneficios]*

- Cliente es quién paga
- *EL* → se puede reemplazar por una breve descripción del producto
- *ES* → automatización, de gestión, embebido, etc
- *QUE* → detalles del sistema
  |→ notificaciones, visualizar *_datos_que_permite_visualizar_*
- *EN LUGAR DE* → antecedentes: lo que se hace actualmente
  - ej. Concurso de proyectos de extensión:
    EN LUGAR DE recibir solicitudes vía mail y tener que realizar controles manuales
- *EL NUEVO PRODUCTO* → relacionado con los objetivos y los beneficios, encarar la descripción por ese lado
  Tiene que ser detallada la descripción

- Mapa de ecosistema

Conexiones con otros sistemas *(directas e indirectas)* y qué datos se pasan
(nombres abstractos → *sustantivos*)



Caso: Check-in - Check-out

- Lista de participantes que son fuente de requerimientos *(identificar requerimientos: RN-RU-RF)*
- Documentos para analizar para la captura *(explicar por qué también)*
  Ej. → Reglamentaciones.
  → Reportes, comprobantes, planillas/formularios.
  → Manuales de procesos, distribución de responsabilidades y tareas.

- Lista de características

  Primer nivel siempre van a estar:

  Gestión de usuarios (en caso que haya usuarios)
  Iniciar sesión
  Cerrar sesión
  Editar datos de usuario

  Gestión de reportes (generalmente está en caso que se quiera tener control de algo)

  Atributos de calidad
  (más comunes)
  Usabilidad
  Seguridad
  Privacidad

Portabilidad

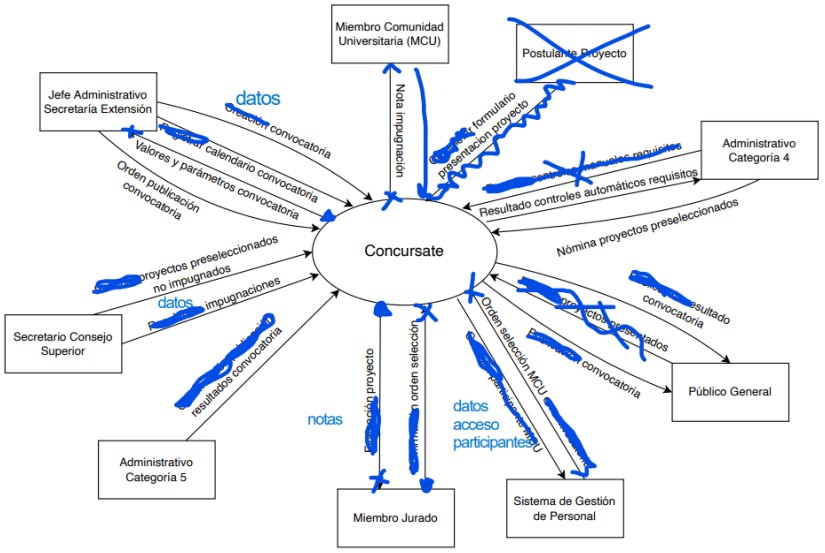
|→ Si está portabilidad, no hay como característica de 1er nivel: *"Interacción con otros sistemas"*

- Limitaciones del producto

  Cosas que el producto **no** va a asegurar
  Ej → El sistema no va a decidir/ El sistema no va a asegurar/ …

- Premisas y riesgos del proyecto
  - Premisas → cosas que se asumen //a validar c/ cliente igual
  - Riesgos → Si lo pongo en premisa,no pongo la negación en riesgos
    - Cosas que influyen en el éxito del proyecto *(negativamente)*
  - Si pongo algo en las oportunidades no lo pongo en las premisas
  - Ej: Caso Taller TOBAS → Comprar tablets, si lo pongo como una oportunidad no lo pongo como premisa
- Diagrama de contexto

  (Diagrama de flujo de datos nivel 0)
  Solo actores que interactúan con el sistema
  En las flechas van datos, nombres abstractos
  Tiene que ser consistente con la lista de características
  Ej. Concurso de proyectos de extensión



w

Reglas del Negocio

- Hechos |→ Frase que es verdadera *(fact)*
  Ej. Cada paquete tiene un costo de envío
- Restricciones |→ Frase que ***limita*** las operaciones/funcionalidades del sistema
  |→ Limita acciones del usuario, por ejemplo restricciones de acceso
  Ej. El usuario *x* puede hacer tal cosa
  Ej. Cliente no puede solicitar más de 3 préstamos
- Disparador de acción |→ Inicia actividad específica bajo determinadas condiciones

|→ tiene forma de inferencia a veces pero no es, porque desemboca en una actividad o algo que debe hacer el sistema.
Ej. Si es el último día del mes, generar un reporte general

- Inferencias |→ Hecho que deriva de otro hecho
  |→ Suelen tener la misma forma que los *DA*
  Ej. Si un cliente realizó un pedido en los últimos 30 días se lo considera activo
- Regla de cómputo |→ Indica cómo transformar un dato dado en un nuevo dato usando una fórmula
  Ej. El sueldo neto de un empleado es el sueldo brou, menos los aportes del empleado a la OS y a la caja jubilatoria, menos el impuesto a las ganancias, más la antigüedad, las asignaciones complementarias y el salario familiar.

Extras
Requerimiento **Inconsistente** → dos o más requerimientos no pueden satisfacerse al mismo tiempo
Requerimiento **Impreciso** → si no puede interpretarse
Requerimiento **Ambiguo** → si puede interpretarse de múltiples formas.

Parcial 2

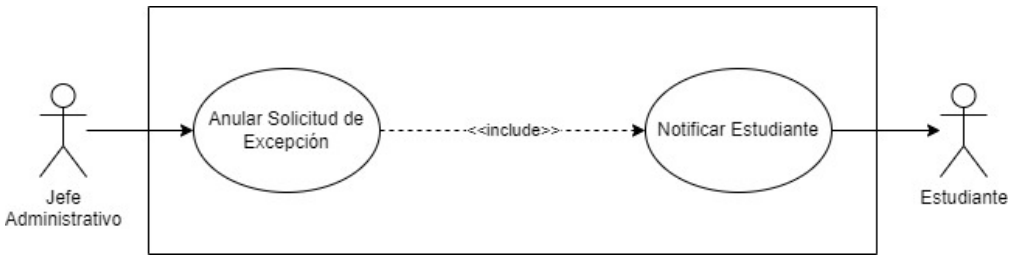
Qué nos pueden pedir
En base a los casos trabajados

Requerimientos de Usuarios

Diagrama de Casos de Uso

- Identifique los actores.
- Identifique los casos de uso.
- Elabore el Diagrama de Casos de Uso para el Sistema de Gestión de Solicitudes de Excepción del Departamento de Ciencias e Ingeniería de la Computación de la Universidad Nacional del Sur.

- Generalmente, si el sistema ***notifica*** a alguien, es un caso de uso a parte que es <<include>> (si siempre se notifica) de otro.
  - Ej. Caso: Solicitudes de Excepción
    - Si se *anula una solicitud* se notifica al estudiante



- Siempre es mejor poner un caso de uso a no ponerlo si no estamos seguros, porque si lo pusiste podes justificar por qué pero si no está, no tenés forma de constatar por qué no está (los ayudantes no lo ven jej)
- Si hay un caso donde se puede aprobar y rechazar algo, y se sigue un circuito diferente para alguno de los dos, entonces son casos de uso separado
- Siempre hay que mirar qué cosas hay que hacer porque puede haber casos que se repiten porque tienen el mismo circuito, y entonces se "unen".
- Fijarse cuando se pueden reutilizar casos. Volviendo al punto anterior, hay casos que pueden ser *inclusión* o *extensión* de varios casos. Como los adjuntar, en este caso, se va a tener un adjuntar sólo si es relevante en el sistema.
- Si hay alguna tarea (*caso de uso)* que realice el sistema tiene que afectar a un actor (*secundario)* para que se ponga en el DCU
    - La tarea que realice el sistema tiene que ser programada, después de X dias o Y horas se realiza la tarea
    - Ej. Después de 48 hs de realizar la solicitud se envía notifica al afiliado del estado de la solicitud (CU: Notificar Afiliado; Actor principal: Timer (sistema); Actor Secundario: Afiliado)

Matriz de Roles y Permisos

Van todos los CU y los Actores del DCU

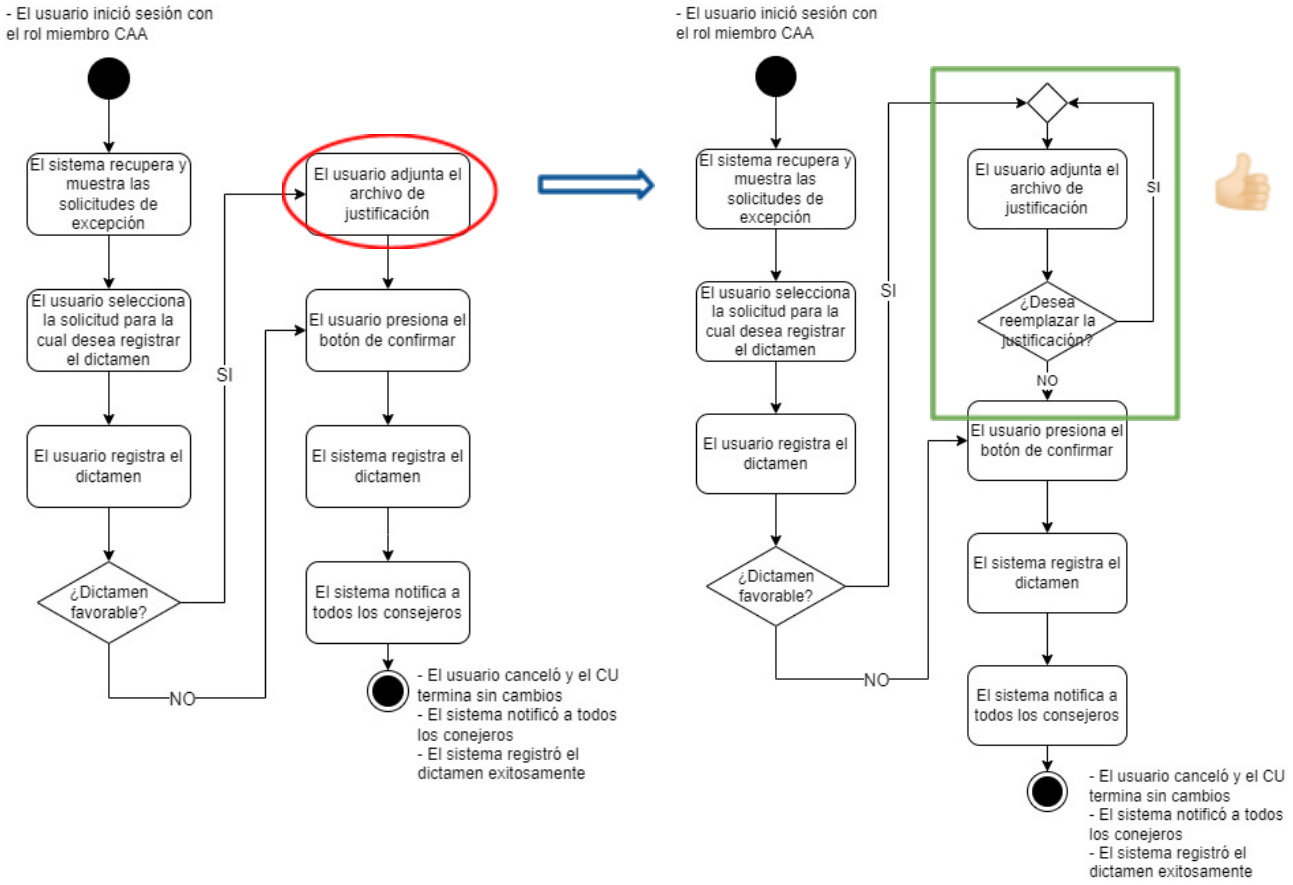
Requerimientos funcionales

Diagrama de Actividad

- Modela secuencia de pasos o acciones
- Se utiliza para modelar el detalle de una tarea
- Tiene pre y poscondiciones
- Con la especificación detallada sale fácil
- Primer paso, si no te dan la GUI, hacerla así te das una idea de cómo va a ser el circuito.
- **Precondiciones**: Siempre va a estar "el usuario inició sesión con el rol X"
- **Poscondiciones**: Si en la tarea tiene que confirmarse, por ejemplo si el usuario tiene que completar algo y dar ok, entonces va a estar "el usuario canceló y el CU terminó sin cambios".
    - Siempre van cosas que podes asegurar cuando termina el caso de uso, ejemplo: el sistema registró/almacenó X cosa; el sistema notificó a _*actor*_; el sistema no encontró X cosa; etc.
- Siempre que el usuario ingresa un dato (*tarea1*: El usuario ingreso X dato) y el sistema habilita algún botón a partir de completar ese campo (*tarea2*: El sistema

habilita el botón Y), cuando el usuario presiona el botón habilitado (*tarea3*: El usuario presiona el botón Y) el sistema **intenta** recuperar y (probablemente) muestra datos en función al dato ingresado, por lo que la siguiente tarea es (*tarea4*) “El sistema intenta recupera y muestra Z”

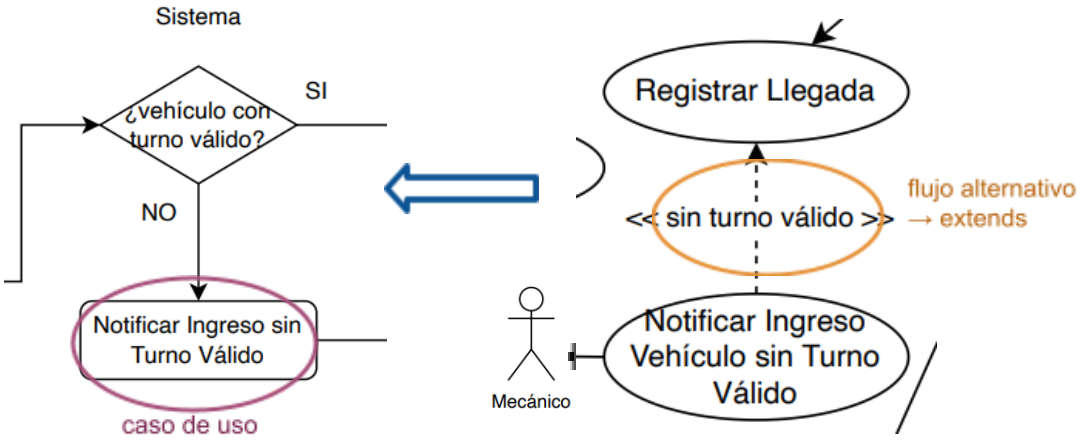
- Está bueno plasmar reemplazar archivo de justificación ej. me equivoque de archivo y quiero subir el que está bien



- En las tareas plasmar bien:
  - Las GUI, cuándo se muestran y qué.
  - Los mensajes de éxito MSG o de error ERR
  - Las reglas del negocio o artículos que afectan a la tarea

Diagrama de Carriles

- Modela el flujo que involucra tareas realizadas por distintos actores.
  - Se consideran casos de uso que son ***extensión*** de otros.
  - No se ponen los ***include*** porque no generan cambios en el flujo.
  - Si hay un caso de uso que es <<*notificar …*>> (solo si el “notificar…” ***no es*** include de otro) hay un carril para el sistema.



- Mirar siempre bien las reglas del negocio o artículos que haya en el caso para poder encontrar el orden correcto de los casos de uso
- Solo van casos de uso, con el nombre tal cual

Especificación Detallada

- Precondiciones y poscondiciones son iguales a las del Diagrama de Actividad.
- Para el flujo normal se considera el paso a paso minucioso cuando se hace el caso de uso que se detalla
- Siempre:
  - Para el flujo normal:
    - "..el sistema muestra la interfaz GUI-x"
    - *(1)* "..el sistema ***recupera*** y ***muestra*** X" según los datos ingresados previamente
    - "..el sistema habilita X botón" antes de que el usuario presione ese botón
    - Las reglas del negocio que afectan en c/ paso
    - "..el sistema ***almacena*** X"
    - "..el sistema muestra el mensaje de éxito MSG-x"
  - Para el flujo alternativo:
    - "El usuario puede cancelar entre x.0 y z.0"
    - *(1)* "i.1.1 el sistema no logró recuperar X " ; "i.1.2 el sistema muestra el mensaje de error ERR-x" ; "i.1.3 Volver a x.o" x.0 es el paso del flujo normal donde se ingresó algo que está asociado (según alguna RN) a lo que el sistema quiere recuperar, es decir, se ingresa algo que le permite al sistema recuperar datos. Ej. Se ingresa el LU de un alumno para recuperar su nombre y apellido, si el LU está mal ingresado, el sistema no puede recuperar el nombre y apellido del alumno y se genera un flujo alternativo que denota un error.
  - Para el flujo excepcional
    - Cosas que el sistema no puede controlar, ej. no hay internet, se corta la luz, etc.
- En la sección de reglas del negocio, van todas las RN y artículos que impactan directa e indirectamente en el flujo normal de la tarea. Además, si hay reglas del negocio que impactan directamente una tarea, ej. El sistema genera el número de prescripción según **DA-1**, se ponen también en el paso del flujo normal.

- Presunciones por lo general (nunca en los casos que nos dan en la materia) no hay
- La *descripción* en la especificación detallada es contar en criollo lo que se hace en la tarea en base al flujo normal. Lo recomendable es hacerla al final.
- **Plantilla:**

ID:	**Nombre:**	
Autores:	**Creado:**	**Última Modificación:**
Actores:		
Descripción:		
Precondiciones:		
Poscondiciones:		
Presunciones: ----		
Flujo Normal		
Flujos Alternativos		
Flujos Excepcionales		
Frecuencia de Uso:	**Prioridad:**	
Reglas del Negocio:		

Especificación Breve

- Una *especificación breve* puede ser un párrafo que describe la tarea con el nivel de detalle que resulte suficiente para que el usuario valide el caso de uso, el desarrollador lo implemente y el responsable de testing defina los casos de prueba.
- Cada especificación puede estar acompañada por las etiquetas o rótulos que identifican a las reglas del negocio vinculadas al caso de uso.
- Los casos de uso consultar son los más fáciles porque tienen menos pasos y nop hay que almacenar nada
- Ej. Caso: Solicitudes de Excepción
  CU: Consultar pedidos de documentación adicional
  El sistema recupera y muestra los pedidos de documentación adicional disponibles. El jefe administrativo selecciona un pedido de documentación adicional. (se abren los detalles del pedido) El jefe administrativo puede salir en cualquier momento. El CU termina sin cambios.

Parcial 3

Qué nos pueden pedir

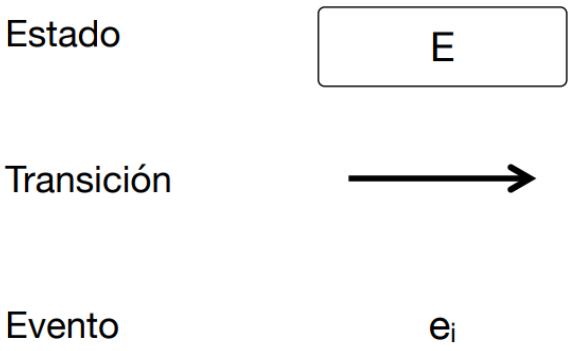
En base a los casos trabajados

Requerimientos para los Datos

Lo ideal es, a medida que vamos leyendo el caso dado marcar todo lo necesario posible para el desarrollo del diagrama de estados y el diagrama de clases. Identificar clases, atributos, asociaciones, quién se asocia a qué y cuándo es conveniente separar en clases distintas, estados, sus transiciones, casos puntuales de transiciones, etc.

Diagrama de transición de Estados

- Tiene estados y transiciones dadas por **casos de uso** (eventos que generan las transiciones)

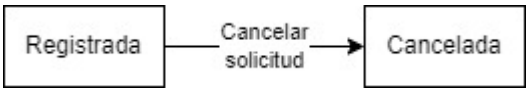


- Primero hay que identificar todos los estados posibles para la situación planteada. Ej. estados de una *solicitud.*
- Mirar bien a lo largo de todo el caso los estados distintos que se pueden dar y observar detalladamente las reglas del negocio.

- Si encuentro una RN que me indica la existencia de un estado y hacia donde va, anoto el estado y al lado la RN así no me olvido donde está y no pierdo tiempo (yo lo hago en una hoja aparte donde anoto todo lo que voy a usar del caso)
  Ej. *Solicitudes de Mantenimiento y Reparación*

  RN-20 Una solicitud solo puede ser cancelada si su estado actual es "registrada".

  |→ Nos da 2 cosas, existe un estado Cancelada para la solicitud y un estado Registrada, y además la transición de Registrada hacia Cancelada. Luego, voy al Diagrama de Casos de Uso y miro qué caso de uso me permite cancelar una solicitud *(Cancelar solicitud)* , ese va a ser el evento de la transición.

- Si hay una RN que dice que para "estado_1,....,estado_x" se considera *pendiente*, eso nos ayuda a saber qué clases y asociaciones vamos a poner en la matriz cuando tengamos que recuperar algo que esté "pendiente" y nos va a dar estado posibles.
- Nunca (generalmente) se pone el estado pendiente en el diagrama de estados.
  Ej. *Solicitudes de Mantenimiento y Reparación*

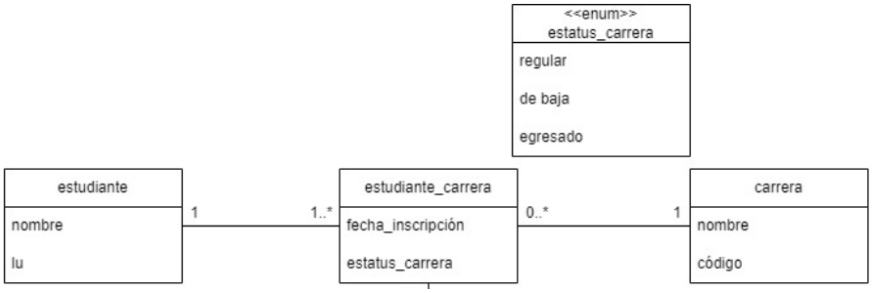
  RN-19 Una solicitud cuyo estado actual es "registrada", "evaluada", "asignada", "iniciada" o "finalizada" se considera pendiente.

  Existen los estados (para el diagrama de estados):
  - Registrada;
  - Evaluada;
  - Asignada;
  - Iniciada;
  - Finalizada.
- Si está el diagrama de carriles sale fácil. El flujo de casos de uso dado por el diagrama de carriles, te da el flujo y los casos de uso que permiten transicionar entre estados en el diagrama de estados.
- Si hay un ***caso de uso*** desde un estado específico que es transición hacia más de un estado, se pone un mini texto que nos indica en qué situación se da esa transición, es como un <<*extends*>>

Diagrama de Clases

- Se modelan entidades, sus atributos y las relaciones entre ellas, como modelo de análisis.
- Es muy parecido al diagrama de E/R
- En realidad se llama Diagrama de Conceptos del Negocio
- La asociación es una relación entre clases.
- Como se utiliza como modelo de análisis ***no*** se indican nombres en las relaciones ni flechas que establezcan sentido de navegación entre las clases.
- Aridad → Cantidad de clases que participan de una relación.
- Multiplicidad → Cuántas instancias de una clase estarán vinculadas a una instancia de la clase relacionada.
  - Multiplicidades: 1 ; 1..* ; 0..1 ; 0..* ; m..n (rango específico)
- Atributos → Cualidad que caracteriza a las instancias de una clase (no se indica el tipo)

- Herencia → mecanismo que establece una relación de generalización-especialización entre clases.
- Clase Asociación → clase que modela la relación entre clases y tiene atributos propios.
    - Como alternativa se puede modelar una clase directamente, que se relacione con las otras dos haciendo de nexo entre ellas.
    - Ej. *Solicitudes de Excepción*



→ La clase estudiante_carrera hace de nexo entre estudiante y carrera para poder modelar los atributos de la relación entre estudiante y carrera.

- En las consultas y reportes, y en los formularios, hay indicios de qué clases se modelan.

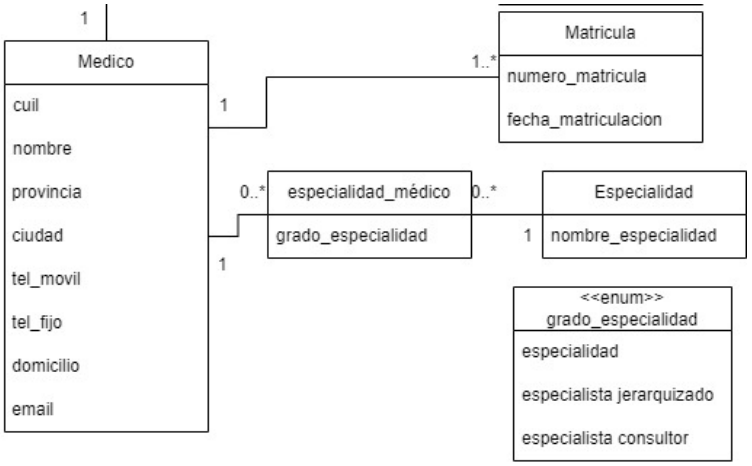
CUIL	27-17169073-4	**Nombre**	Ana	**Apellido**	López
Matrícula	5257	**Fecha Matriculación**		01/10/1980	
Domicilio	Terrada 1234	**Ciudad**	Bahía Blanca	**Provincia**	Buenos Aires
E-mail	ana.lopez@gmail.com	**Teléfono Fijo**		**Teléfono Móvil**	0291-154230022
Especialidad				**Grado**	
Ginecología				Especialista Consultor	
Obstetricia				Especialista Jerarquizado	

Observación: el grado es "especialista", "especialista jerarquizado" o "especialista consultor".

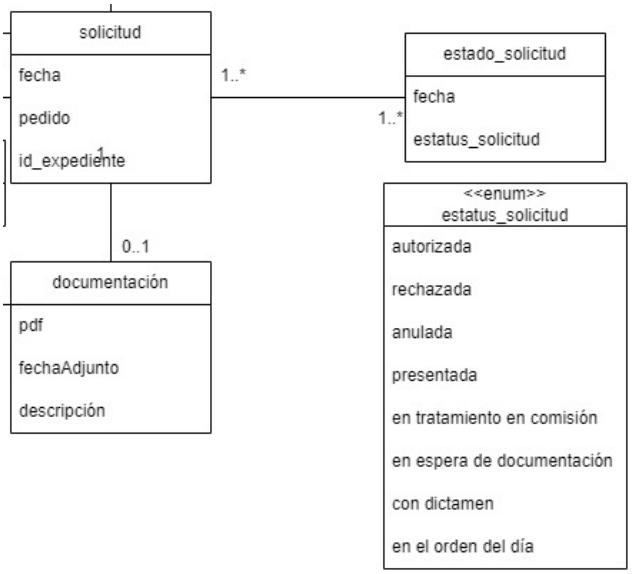
Por ejemplo, en el caso de *Circuito de Estudios de Diagnóstico por Imágenes*, si observamos el formulario de los Médicos Prestadores de la OS podemos deducir:

1) Al haber un formulario para médico, hay que mantener los datos de los médicos por lo tanto va a haber una clase que sea Médico.
2) Los datos de la clase Médico son los que se mantienen en la planilla, como el *nombre, apellido, matrícula, fecha de matriculación, domicilio, ciudad, cuil, teléfono, especialidad y grado.*
3) El problema es que muchos médicos pueden tener la misma especialidad, entonces eso nos da la pauta de que va a haber que modelarla aparte, en otra clase.
4) Además el grado de la especialidad ya sabemos que tiene valores definidos (*por default)* por lo que, probablemente, haya un enumerado de por medio.
5) Por otro lado, el grado de la especialidad no es ni de la especialidad, ni del médico, es de la relación entre el médico y la especialidad porque el médico tiene un grado para determinada especialidad que tenga por lo que tenemos 2 opciones. Modelarlo como una clase asociación entre médico y especialidad o como una clase intermedia

entre estas dos. Esa clase va a tener como atributo el grado que va a estar asociado a un enumerado.



- Si hay que realizar un seguimiento histórico de los estados, como por ejemplo poder ver todos los estados por lo que pasó un paquete en el correo, en vez de modelarlo como un atributo únicamente, se modela como una clase a parte asociada a la clase que tiene ese estado, esto es para que los estados no se “pisen”. Además se ponen como atributos el estado (que va tomar los valores del enumerado) y la fecha y hora. La multiplicidad de la relación es 1..*
  Ej. *Solicitud Excepción*
  La solicitud tiene estados por los que va pasando y en los antecedentes se menciona que el estudiante puede realizar el seguimiento de las solicitudes propias. En este caso, la solicitud va a tener varios estados asociados con fecha y hora.



- Los estados del diagrama de estados van a formar parte de un enumerado <<enum>> que contenga los estados asociados en un atributo de una clase.
- A partir de las reglas del negocio podemos obtener data de cuándo se modelan datos o no y qué datos se van a modelar y cómo se realizan las relaciones y asociaciones entre clases.

Ej. *Solicitud de Reparación y Mantenimiento*

RN-1 Cada dependencia de la municipalidad posee un nombre y una sigla que la identifica.

Esto nos indica que va a haber una clase Dependencia que va a tener como atributos un nombre y una sigla identificatoria

RN-9 Cuando se registra una solicitud, el sistema le asigna un rótulo unívoco que la identifica.

La solicitud se registra en el sistema, esto es se almacena y se almacena su estado, por lo que de antemano sabemos que vamos a tener una clase que modele esa solicitud. La RN-9 nos da la pauta de que va a tener un atributo rótulo para identificarla. Como un id en una base de datos.

- Observar que también las interfaces gráficas nos dan la pauta de qué datos se modelan y qué datos no, como atributos de una clase.

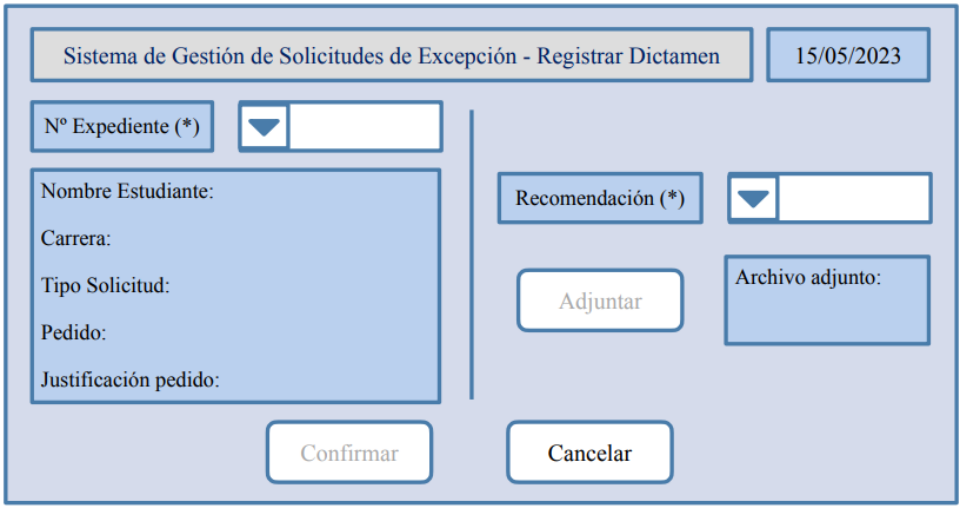
Matriz de Validación

Es una herramienta que nos ayuda para saber si falta modelar datos en el diagrama de clases. Para la plantilla que se adopta en la materia, en la primera columna contiene los CU y en la primera fila *crea, modifica, elimina* y *usa/lee/consulta*, que denotan las acciones que se generan mediante esos CU en cuanto a las clases y asociaciones.

	Crea	Modifica	Elimina	Usa/Lee/Consulta
CU_1	Cla_x *(clase)* Cla_y - Cla_z *(asociación)* …	…	…	…
…				
CU_n				

En las columnas se ponen las clases y/o asociaciones que se ven afectadas por los distintos casos de uso de la primera columna.

- Las clases se ponen con su nombre
- Las asociaciones se ponen con el nombre de la clase 1 que se asocia a la clase 2, separadas por un **-** *(guión)*.
- Para saber que clases o asociaciones se ven afectadas por ese caso de uso es útil ver las consultas y reportes, planillas o formularios, GUIs y las descripciones detalladas o breves.
  Ej. *Solicitudes de Excepción*



GUI-101

Para la GUI (gui-101) presentada para registrar un dictamen, se solicita el número de expediente, el cual es un atributo de la solicitud por lo que para Registrar Dictamen se ***Usa/Lee/Consulta*** la clase "Solicitud". Luego para recuperar el nombre del estudiante, la carrera en la que está inscripto, etc. Se necesita consultar a las asociaciones de "Solicitud" con "estudiante_carrera", y de "estudiante_carrera" con "carrera" y por otro lado con "estudiante".

Coloquio

Enfoque centrado en el usuario

Prototipos

El uso de prototipos de software hace que los requerimientos sean más tangibles, visibilizando los casos de uso y achicando la brecha entre el entendimiento o concepción del sistema por parte de clientes/usuarios y el analista.
Ayudan a revelar y resolver ambigüedades y la incompletitud de los requerimientos
→ El **prototipado** permite evitar que se inviertan más recursos de los necesarios para el desarrollo del producto
Un prototipo puede ser:

- Experimento o simulación del producto final
- Porción o modelo del sistema real, que incluso puede no proveer funcionalidades
- El feedback temprano sobre los prototipos ayuda a establecer el acuerdo base, reduciendo así el riesgo de no satisfacer las expectativas de clientes y usuarios

Un prototipo que ayuda a los usuarios y desarrolladores a visualizar cómo los requerimientos deben ser implementados puede revelar deficiencias en los requerimientos

Propósito

Aclarar, completar y validar requerimientos → reducen la incertidumbre, permiten establecer acuerdos, identificar malentendidos, imprecisiones o ambigüedades, encontrar errores y omisiones, evaluar la calidad y validar los requerimientos a un bajo costo
Explorar alternativas de diseño → permiten explorar distintas técnicas de interacción con los usuarios, evaluar y optimizar la usabilidad y eficiencia, explorar el uso de distintas tecnologías, y confirmar la visión del producto antes de construir la solución
Crear un sub-producto que evolucione hacia el producto final → permite crear un subconjunto de funcionalidades no entregable y hacerlo evolucionar a través de ciclos cortos hasta alcanzar un producto entregable

Atributos o Descriptores

Alcance → maqueta o prueba de concepto
Uso futuro → descartable o evolutivo
Formato → papel o electrónico
Cada prototipo posee una combinación específica de valores para estos atributos
Ej.:
→ Maqueta descartable en papel
→ Prueba de concepto evolutiva electrónica : Pieza de sw funcional que ilustra alguna capacidad técnica deseada, que puede evolucionar hacia un producto entregable

Alcance

Maqueta (*Mock-up*)

- Hace **foco** en la **experiencia del usuario**
- Se centra en una porción de la interfaz de usuario
- **No** hace hincapié en la arquitectura ni en funcionalidad detallada
- Muestra la interfaz del usuario, pantallas y su navegación, con muy poca (o ninguna) funcionalidad
- Navegación limitada → efecto nulo o el efecto es un msj probablemente cte
- **Usuario puede juzgar si podrá hacer su trabajo con el sistema y refinar sus requerimientos**. No se muestra el trabajo que hará con el sistema

Prueba de concepto (*Proof-of-concept*)

- Explora la factibilidad del enfoque propuesto
- **Implementa una porción de las funcionalidades del sistema**, atravesando todas las capas técnicas, desde la interfaz hasta la implementación efectiva del software
- Funciona como va a funcionar el sistema real
- Útil cuando se desea evaluar si la **arquitectura propuesta** es sensata y factible, cuando se desea optimizar algoritmos, evaluar un esquema de base de datos, confirmar la solidez de una solución existente en la nube, o testear requisitos críticos de sincronización

Uso futuro

Descartable (*Throwaway prototype*)

- Se descarta luego de ser utilizado para generar feedback
- Hace enfásis en la implementación rápida y la modificación sobre la ***robustez***, ***confiabilidad***, ***performance***, y ***mantenibilidad a largo plazo***

- Debe estar **enfocado en una porción del sistema** sobre la que hay un alto nivel de *incertidumbre, ambigüedad* o un alto *riesgo* de que se hayan *omitido reqtos.*
- Debe construirse lo más rápido posible e invirtiendo la menor cantidad de recursos y tiempo
- No debería evolucionar para transformarse en el producto final, porque probablemente no se consideraron, entre otras cosas, atributos de calidad como robustez, confiabilidad, rendimiento o mantenibilidad.
- Es más apropiado cuando el equipo enfrenta incertidumbre, ambigüedad, incompletitud o imprecisión en los requerimientos
- **Wireframe** → es un tipo de prototipo descartable
  |→ comúnmente usado para el diseño de la interfaz personalizada de usuario y el
  | diseño web
  |→ útiles para ayudar a entender a los usuarios los tipos de actividades que pueden querer hacer en la pantalla

Evolutivo (*Evolutionary prototype*)

- Brinda una base arquitectónica sólida para construir el producto de forma ***incremental*** a medida que los requerimientos se vuelven más claros con el tiempo
- Crece hasta producto final a través de una serie de iteraciones
- Debe construirse con buenas prácticas, código robusto y de calidad desde el comienzo, y diseñarse para su fácil y frecuente crecimiento y mejora
- La **primera iteración** de un prototipo evolutivo puede pensarse como una **entrega piloto** que implementa una porción inicial de los requerimientos del sistema
- El desarrollo ágil constituye un ejemplo de prototipado evolutivo. El equipo construye el producto a través de una serie de iteraciones, utilizando el feedback de las iteraciones anteriores para ajustar la dirección de los siguientes ciclos de desarrollo
- Toma más tiempo que un prototipo descartable

Caracterización mediante atributos

	Descartable	Evolutivo
Maqueta	• Aclarar y refinar requerimientos de usuarios y funcionales. • Identificar funcionalidades ausentes. • Explorar interfaces de usuario alternativas.	• Implementar requerimientos de usuarios centrales. • Implementar requerimientos de usuarios adicionales, en base a prioridades. • Implementar y refinar sitios web. • Adaptar un sistema ante cambios frecuentes y rápidos en las necesidades del negocio.
Prueba de Concepto	• Verificar factibilidad técnica. • Evaluar rendimiento. • Adquirir conocimiento para mejorar las estimaciones en los recursos.	• Implementar e incrementar la funcionalidad principal de varios niveles y sus capas de comunicación. • Implementar y optimizar algoritmos centrales. • Testear y ajustar rendimiento.

Formato

Papel (*Paper prototype*)

- Dibujo en papel, pizarrón o en una herramienta de dibujo
- Ayudan a probar si los usuarios y desarrolladores comparten la misma idea de los requerimientos
- Forma barata, rápida y sin requisitos de tecnología para explorar cómo podría verse una porción del sistema
- Permiten dar pasos tentativos y poco riesgosos en dirección a una solución antes de pasar a la etapa de implementación
- El diseñadores de las interfaces sketches ideas de pantalla sin preocuparse por la posición y apariencia exacta de los componentes
- Útiles para **explorar funcionalidades** y el **flujo del sistema**
- Ayudan a que las iteraciones sean más rápidas

Electrónico (*Electronic prototype*)

- Software funcional para solo una parte de la solución
- Existe una **gran variedad de herramientas para construir prototipos electrónicos**, en particular para el maquetado de interfaces gráficas
- Las herramientas permiten implementar y modificar los componentes de las interfaces fácilmente, independientemente de la eficiencia del código
- Iterar sobre simulaciones, ensamblando pantallas, estableciendo el flujo de navegación, e implementando controles de interfaz de usuario provee un mecanismo valioso para aclarar y validar requerimientos de usuarios y evaluar la solución propuesta

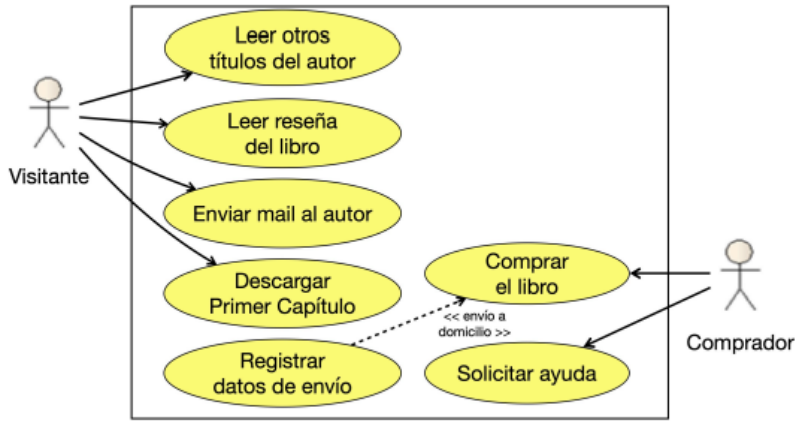
Mapa de diálogo

→ Modela el diseño de una interfaz de usuario a un nivel alto de abstracción
→ Muestra los elementos asociados a las tareas de usuario y los links de navegación entre ellos, sin contemplar detalles del diseño de pantallas

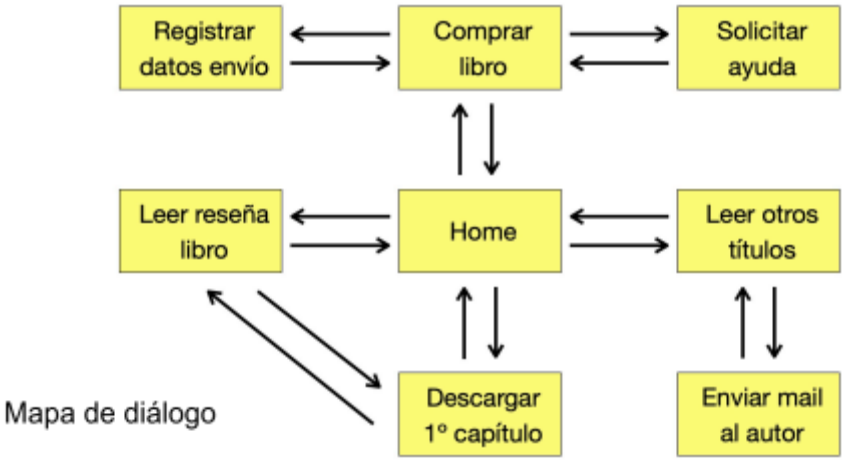
Componentes

- **Caja/Rectángulo**: página que contribuye a proveer los servicios identificados en los casos de uso
- **Flecha**: link que habilita la navegación entre páginas

Ej.: Sitio web para la publicidad y venta de un libro



DCU reducido



Uso de prototipos

Luego de que los requerimientos son refinados y las pantallas son diseñadas, cada elemento de las interfaces de usuario pueden ser optimizadas para la usabilidad

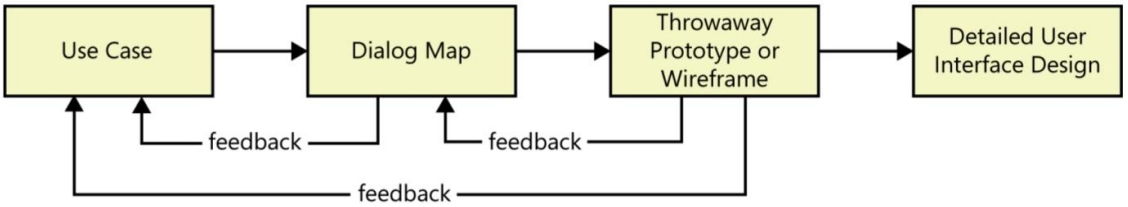
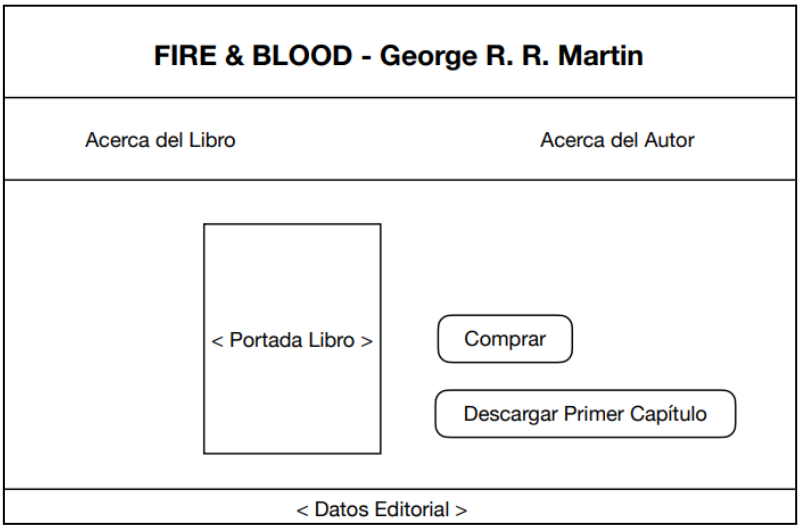


FIGURE 15-2 Activity sequence from use cases to user interface design using a throwaway prototype.

Ej.: Sitio web para la publicidad y venta de un libro - ***Prototipo***



Evaluación – Consideraciones importantes

- Observar a los usuarios interactuar con el prototipo y evaluar la usabilidad
- Elegir usuarios representativos por su nivel de experiencia, perspectiva o rol
- Crear guías para que los usuarios realicen una serie de tareas u operaciones específicas derivadas de los casos de uso o características abordadas en el prototipo

- Elaborar encuestas que los usuarios pueden completar inmediatamente después de usar el prototipo
- Preguntas generales:
  • ¿El prototipo implementa la funcionalidad como vos esperabas?
  • ¿Qué funcionalidad le falta al prototipo?
  • ¿Se te ocurre alguna condición de error que el prototipo no contemple?
  • ¿Hay alguna función innecesaria?
  • ¿Hay alguna forma de simplificar alguna de las tareas que requieren muchos pasos?
  • ¿Alguna vez dudaste de que hacer después?

Riesgos

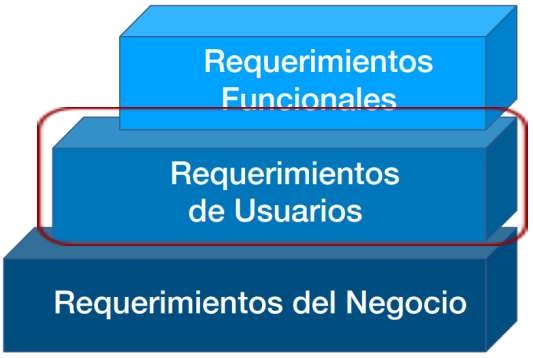
- Inversión de recursos en forma excesiva
- Presión para lanzar el producto
- Generación de falsas expectativas de performance
- Distracción en detalles irrelevantes
- El prototipo reemplaza a la especificación de requerimientos ← *¡nunca debe pasar!*

Factores de éxito

- Incluir tareas de prototipado en el plan del proyecto
- Indicar el propósito de cada prototipo antes de construirlo y explicar qué pasará con el resultado
- Planificar el desarrollo de varios prototipos
- Crear prototipos descartables mientras más rápido y barato sea
- Invertir la menor cantidad de esfuerzo posible para responder preguntas y/o resolver incertidumbre en los requerimientos
- No prototipar requerimientos que ya son comprendidos, excepto para explorar alternativas de diseño
- Usar datos creíbles en los prototipos de pantallas y reportes
- No esperar que un prototipo reemplace requerimientos escritos

Atributos de calidad

Comportamiento y cualidades del producto



Desde los primeros encuentros el analista captura las necesidades de clientes y usuarios.

Qué: Requerimientos funcionales especifican el comportamiento del sistema

Cómo: Los ***atributos de calidad*** establecen las cualidades del comportamiento.
Ejemplo → *"Necesito que de lunes a sábado el sistema esté disponible las 24hs del día"*

Importancia

● Influyen en el nivel de satisfacción que brinda el producto al cliente.
- Pueden determinar el éxito del proyecto

● Originan *requerimientos funcionales*, y marcan *decisiones de diseño* y arquitectónicas.
● Es más complicado rediseñar un sistema para que cumpla el estándar de calidad deseado que hacerlo desde un principio
● Deben abordarse las expectativas de calidad de clientes y usuarios durante la captura de requerimientos.
● Ignorar las expectativas de calidad de clientes y usuarios puede conllevar demoras en el lanzamiento del producto.

Clasificación

Externos

- Son percibidos y valorados por el usuario
- Describen características observables cuando el software está ejecutándose
- Influyen en la satisfacción de clientes y usuarios.

➔ Disponibilidad
La medida en que los servicios del sistema están disponibles cuándo y dónde se los necesita.
Ej. Sistema disponible las 24hs del día
- ¿Qué porcentaje de tiempo durante el día debe estar disponible el sistema?
- ¿En qué horario el sistema tiene que estar disponible el 100% del tiempo?
- ¿Todos los días?¿Todo el año?
- ¿Todas las funcionalidades?
- ¿Qué funcionalidades son más críticas y requieren mayor disponibilidad?
- ¿Qué consecuencias pueden provocar que la disponibilidad del sistema no satisfaga los requerimientos especificados?
- ¿En qué horarios y días resulta más adecuado planificar las tareas de mantenimiento que afecten a la disponibilidad del sistema?¿Cuál es la duración máxima tolerable?
- ¿Cuál es la duración máxima tolerable para un mantenimiento no programado?¿Cómo se van a manejar los intentos de acceso de los usuarios durante los períodos de mantenimiento programado y no programado?
- ¿Cómo se notifica a un usuario que está usando el sistema, que dejó de estar disponible por completo o algunas funciones?
- ¿Qué dependencias existen sobre las funcionalidades que afectan a la disponibilidad?
- Si una tarea de mantenimiento puede realizarse con el sistema funcionando, ¿Qué impacto será tolerable sobre la confiabilidad y la performance?¿Cómo podrías reducirse ese impacto?

➔ Instalabilidad
Cuán fácil es instalar, desinstalar y reinstalar correctamente el sistema
- ¿Cuántos minutos demande instalar el sistema?
- ¿Cuántos minutos es tolerable que demande instalar el sistema completo a un usuario capacitado?
- ¿Qué operaciones de instalación pueden realizarse sin afectar el trabajo de los usuarios?
- ¿Qué operaciones de instalación requieren reiniciar el dispositivo?

- ¿Qué controles pueden realizarse para verificar el éxito de la instalación?
- ¿Qué usuarios están autorizados para realizar cada una de las operaciones vinculadas con la instalación?

➔ Integridad

La medida en que el sistema protege contra la inexactitud y la pérdida de datos

Causas y consecuencias

- Daños físicos en los dispositivos de almacenamiento
- El usuario sobrescribe un archivo sobre otro
- Una falla del software provoca que se pierdan las referencias
- Una falla en los dispositivos de lectura provoca que el formato de los datos se corrompa

➔ Interoperabilidad

Con qué facilidad el sistema puede interconectarse e intercambiar datos con otros sistemas o componentes

- ¿Con qué sistemas externos interactúa el producto?
- ¿Qué estructuras de datos y qué servicios se intercambiarán?
- ¿En qué formato deben realizarse los intercambios?
- ¿Qué dispositivos de hardware deben interconectarse?
- ¿Qué protocolo de comunicación se utilizará en cada caso?
- ¿Qué otros requerimientos impone el sistema con el que se realiza la comunicación?

➔ Rendimiento

Cuán rápidas y predecibles son las respuestas del sistema a las entradas del usuario u otros eventos

- ¿Cuál es el tiempo de respuesta que usted considera **aceptable** para que el sistema responda la consulta *q1*?
- ¿Cuál es el tiempo de respuesta que usted considera **inaceptable** para que el sistema responda la consulta *q1*?
- ¿Cuál es el tiempo de respuesta que usted considera **adecuado** para que el sistema responda la consulta *q1*?
- ¿Cuántos usuarios en promedio podrían estar usando simultáneamente el sistema?
- ¿Cuál es el número máximo de usuarios que el sistema debería poder llegar a atender simultáneamente?
- ¿Cuál es el horario del día, el día de la semana o la época del año en la que se espera su uso más intenso?

➔ Confiabilidad

Cuánto tiempo funciona el sistema antes de experimentar una falla

Ej. cajero no te lee la tarjeta, entonces no es confiable

- ¿Cuánto tiempo debería transcurrir como mínimo entre fallas?
- ¿Qué tarea estaría dispuesto a rehacer el usuario ante una falla?
- ¿Qué tareas no pueden repetirse si se produce una falla?
- ¿Qué funcionalidades del sistema debería de ser superconfiables?
- ¿Cuánto tiempo puede estar caído el sistema sin afectar significativamente el negocio?

➔ Robustez

Cuán bien responde el sistema a condiciones de operación inesperadas

➔ Safety

Cuán bien protege el sistema contra lesiones o daños físicos a las personas

➔ Seguridad
Cuán bien protege el sistema contra el acceso no autorizado a la aplicación y sus datos
- ¿Necesita que el sistema sea seguro?¿Cuán seguro debería de ser el sistema?
- ¿Cuáles son los datos “sensibles”?
- ¿Quiénes están autorizados a ver, copiar, modificar, borrar datos sensibles?
- ¿Qué funcionalidades del sistema quedan restringidas a ciertas clases de usuarios?
- ¿Qué chequeos se establecen para asegurar que el usuario está trabajando en un entorno seguro?
- ¿Qué datos son confidenciales?
- ¿Sobre qué funcionalidades es necesario restringir el acceso?
- ¿Para qué tipo de usuarios?
- ¿Es posible definir roles?¿La estructura de roles es jerárquica?
- ¿Qué mecanismo es adecuado para restringir el acceso?¿Contraseñas?¿Credencial digital?¿Firma digital?
- ¿Qué ítems, cuando son eliminados, se mantienen almacenados en un tarro de basura?¿Qué usuarios pueden acceder al tarro de basura?¿Cuánto tiempo?¿Pueden restablecerse?¿Qué usuarios pueden restablecerlos?
- ¿Con qué frecuencia se ejecuta el software para detectar virus?

➔ Usabilidad
Cuán fácil es para las personas aprender, recordar y usar el sistema
- ¿Cuál es el tiempo promedio para que un usuario entrenado complete una tarea?
- ¿Cuál es el tiempo promedio para que un usuario complete el entrenamiento que le permita realizar sus tareas básicas?
- ¿Cuántas transacciones puede completar un usuario en un tiempo dado?
- ¿Cuántas interacciones *(clicks, teclas, toques)* se requieren para realizar una tarea?
- ¿Cuánto tiempo transcurre entre dos solicitudes de ayuda de un usuario?
- ¿Qué porcentaje de tareas puede realizar el usuario sin pedir ayuda?
- ¿Cuántos mensajes de error recibe el usuario mientras realiza su tarea?
- ¿Cuántas pantallas visita el usuario hasta que encuentra lo que busca?
- (En todos los casos no se está evaluando al usuario sino al sistema)

Internos
- No son observables durante la ejecución del sistema
- Describen propiedades que se perciben durante el desarrollo, verificación o mantenimiento
- Afectan indirectamente a clientes y usuarios

➔ Eficiencia
Cuán eficientemente el sistema utiliza los recursos informáticos
- ¿Cuál es el máximo número de usuarios simultáneos que tendrá el sistema inicialmente?

    - ¿Cuánto puede llegar a crecer ese número?
    - En cada uno de los indicadores de rendimiento, ¿a partir de qué valor se comprometen los objetivos del negocio?

➔ Modificabilidad
Cuán fácil es mantener, cambiar, mejorar y reestructurar el sistema

➔ Portabilidad
Con qué facilidad se puede hacer que el sistema funcione en otros entornos operativos
    - ¿Sobre qué plataformas se ejecutará el software?
    - ¿Qué partes del software tienen mayores requerimientos de portabilidad?
    - ¿Qué datos o componentes deben tener la capacidad de migrar con o sin intervención del usuario?

➔ Reusabilidad
Hasta qué punto los componentes del sistema se pueden utilizar en otros sistemas

➔ Escalabilidad
Con qué facilidad el sistema puede crecer para manejar más usuarios, transacciones, servidores u otras extensiones
    - ¿Cuánto puede llegar a crecer el número de usuarios que acceden simultáneamente al sistema en los próximos meses o años?
    - ¿A cuántos usuarios debería adaptarse el sistema sin afectar la performance y sin cambios de software o de hardware?
    - ¿Cuánto puede llegar a crecer el volumen de datos en los próximos meses o años?

➔ Verificabilidad
Con qué facilidad los desarrolladores y testers pueden evaluar y confirmar que el software se implementó correctamente exhibiendo el comportamiento esperado
    - ¿Cómo se puede confirmar que un algoritmo produce el resultado esperado?
    - ¿Existen algoritmos no determinísticos que aumentan la dificultad de confirmar el resultado?
    - ¿Es posible definir un conjunto de datos que tenga alta probabilidad de detectar errores, si existen?

Tipos de proyecto

● Sistemas Embebidos
- Rendimiento
- Eficiencia
- Confiabilidad
- Robustez
- Safety
- Seguridad
- Usabilidad

● Aplicaciones Web o Corporativas
- Disponibilidad
- Integridad
- Interoperabilidad

- Performance
- Escalabilidad
- Seguridad
- Usabilidad

● Aplicaciones de Escritorio o Móviles
- Performance
- Seguridad
- Usabilidad

Exploración y captura

❖ Ciertos atributos se contraponen lo que hace imposible que se puedan dar todos en simultáneo, por lo que hay que buscar balance y determinar cuáles son más importantes para el **éxito del proyecto**

❖ Los **objetivos de calidad** son definidos en términos de los atributos esenciales, para que los diseñadores puedan tomar las decisiones adecuadas

❖ El sistema completo debe analizarse desde la perspectiva de cada factor de calidad seleccionado

Captura

● No adecuadas
- ¿ Usted considera que el sistema tiene que ser seguro?
- ¿Necesita que sea eficiente?
- ¿Cuáles son los requerimientos de interoperabilidad?
- ¿Qué nivel de seguridad debe garantizar el sistema?
- ¿Qué tan eficiente debe ser el software?

● Pasos a seguir
1) Comenzar con un listado amplio de atributos de calidad y examinar la lista con usuarios representativos
2) Reducir la lista y seleccionar los atributos que son importantes para el proyecto
3) Priorizar cada atributo de calidad, respecto a los restantes
4) Capturar expectativas específicas para cada atributo y realizar preguntas sobre el comportamiento aceptable e inaceptable del sistema

Priorización

Attribute	Score	Availability	Integrity	Performance	Reliability	Robustness	Security	Usability	Verifiability
Availability	2		^	^	^	<	^	^	<
Integrity	6			<	<	<	^	<	<
Performance	4				<	<	^	^	<
Reliability	2					<	^	^	^
Robustness	1						^	^	<
Security	7							<	<
Usability	5								<
Verifiability	1								

- Si tiene que elegir entre *disponibilidad* e *integridad*, ¿cuál elige? Integridad
- **<** indica que el factor de la fila es el más importante // □ indica que el factor de la columna es el más importante (son flechas)
- El factor más importante de la tabla es seguridad porque tiene 7 ptos
- Es importante verificar inconsistencias: Si A > B y B > C, no puede darse A < C

Trade-offs

	Availability	Efficiency	Installability	Integrity	Interoperability	Modifiability	Performance	Portability	Reliability	Reusability	Robustness	Safety	Scalability	Security	Usability	Verifiability
Availability								+		+						
Efficiency	+				−	−	+	−		−		+		−		
Installability	+							+					+			
Integrity			−		−		−		−		+		+	−	−	
Interoperability	+		−	−			−	+	+		+	−		−		
Modifiability	+		−				−		+	+			+			+
Performance		+			−	−		−			−		−		−	
Portability		−			+	−	−			+				−	−	+
Reliability	+	−		+		+	−				+	+		+	+	+
Reusability		−		−	+	+	−	+						−		+
Robustness	+	−	+	+	+		−		+			+	+	+	+	
Safety		−		+	+		−			+				+	−	−
Scalability	+	+		+			+	+	+		+					
Security	+			+	+		−	−	+		+	+			−	−
Usability		−	+				−	−	+		+	+				−
Verifiability	+		+	+		+			+	+	+	+		+	+	

- El signo **+** en una celda indicar que incrementar el atributo de la fila usualmente tiene un efecto positivo en el de la columna
- El signo **-** en una celda indica que incrementar el atributo de la fila generalmente afecta negativamente en el atributo de la columna
- Una **celda vacía** que el atributo de la fila tiene poco o ningún efecto en el atributo de la columna

Trade-offs y priorización

● Para alcanzar el balance óptimo en las características del producto es necesario identificar, especificar y priorizar los *atributos de calidad* pertinentes[1] durante el desarrollo de software.

● Al definir los **atributos de calidad** que son importantes para el proyecto, tener en cuenta el *trade-off* entre los atributos para evitar comprometerse a metas conflictivas

[1] *Que es adecuado u oportuno en un momento o una ocasión determinados.*

Métricas

Las **metas** o **requerimientos asociados** a atributos de calidad del producto deben ser medibles

- Medibles Directamente: pueden cuantificarse. Es posible establecer una escala con **valores mínimos y máximos**
- Medibles Indirectamente: no se les puede asociar valores concretos pero sí **indicadores que permitan decidir** si se cumplieron o no

Restricciones

- Una restricción es un **requerimiento que limita** las alternativas de decisión del diseñador y el resto del equipo de desarrollo
- Pueden ser impuestas por participantes externos, por otros sistemas que interactúan con el que se está construyendo o por otras etapas del ciclo de vida del sistema.

  Captura

  - ¿Por qué se impone la restricción de usar un software de código abierto? Quizás el verdadero requerimiento es mejorar la *extensibilidad*.
  - ¿Por qué se impone usar *.NET*? Quizá el verdadero requerimiento es lograr *portabilidad*.

Calidad de los Requerimientos

Características individuales

Aplican a cada requerimiento en forma individual, sin considerar los requerimientos restantes.

- Completo
  - Un requerimiento es **completo** si tiene toda la información necesaria para que el lector pueda comprenderlo
    Para ***requerimientos funcionales***:
    - El desarrollador puede implementarlo correctamente
    - El responsable de testing puede escribir un test de validación
  - Está vinculado con la ***correctitud***
- Correcto
  - Un requerimiento es **correcto** si describe exactamente una característica o funcionalidad del sistema que permite satisfacer las necesidades de alguna la parte interesada
    Para ***requerimientos funcionales***:
    - Describe claramente la funcionalidad a ser implementada
  - Para evaluar la correctitud es necesario recurrir a la *fuente del requerimiento*
  - Está vinculado a la ***trazabilidad***

- Factible
  - Un requerimiento es **factible** si puede implementarse en el entorno operativo del sistema y con los recursos establecidos en el proyecto (tiempo, presupuesto, personal, etc.)

    - La participación de desarrolladores en algunos encuentros de captura es útil para chequear si en la práctica se van a poder implementar las necesidades que se van planteando, en términos de tiempo, costo, y esfuerzo.
    - Cuando se descartan requerimientos por razones de *factibilidad* es necesario evaluar el impacto en la **visión del producto** y el **alcance del proyecto**.
- Necesario
    - Un requerimiento es **necesario** si corresponde a una característica, tarea, funcionalidad o atributo valioso para un participante identificado como *fuente de requerimientos*.
        - Cada requerimiento debe poder rastrearse al usuario, clase de usuarios o cliente que le dio origen.
        - Cada requerimiento debe estar vinculado a un *objetivo del negocio* que establece por qué es necesario.
    - Está vinculado a la ***trazabilidad***
- Priorizado
    - Un requerimiento está **priorizado** si se ha establecido un valor que indica cuán esencial es para el producto.
    - La *asignación de prioridades* requiere que participantes de diferentes clases expongan su perspectiva y negocien un acuerdo.
- No ambiguo
    - Un requerimiento es **no ambiguo** si solo tiene una interpretación posible o todas las interpretaciones posibles son equivalentes para clientes o usuarios.
        - Una misma persona puede interpretar un mismo requerimiento de múltiples formas.
        - Cada persona asigna una sola interpretación para el requerimiento, pero distintas personas lo interpretan de diferente manera.
    - Está vinculado a la ***completitud***
- Verificable
    - Un requerimiento es **verificable** si es posible establecer un criterio objetivo para decidir si su implementación es correcta.
    - Puede estar acompañado de un conjunto de casos de prueba que permita verificarlo.
    - Está vinculado a la ***correctitud*** y ***completitud***

Características globales

Aplican al conjunto completo de requerimientos.

- Completo
    - Un conjunto de requerimientos es **completo** si abarca todas las necesidades de las partes interesadas y todas las características del producto definidas en el alcance de la iteración actual.
- Consistente
    - Un conjunto de requerimientos es **consistente** si no hay conflicto entre los requerimientos del mismo nivel o de diferentes niveles.
    - Está vinculado a la ***correctitud*** *(individual)* y a la ***trazabilidad***
- Modificable
    - Un conjunto de requerimientos es **modificable** si es posible reconocer e identificar con precisión cada requerimiento del conjunto

    - La identidad evita la redundancia, favorece el reuso y permite mantener la historia de cada requerimiento
    - Está vinculado a la ***trazabilidad***
- Trazable
    - Un conjunto de requerimientos es **trazable** si cada requerimiento puede vincularse hacia atrás hasta su origen y hacia adelante con los requerimientos derivados de él, su diseño, implementación y mecanismos de testing
    - Cada ***requerimiento funcional*** debe poder rastrearsae a la **regla del negocio**, el **requerimiento de usuario**, **requerimiento no funcional** o **requerimiento del negocio** que le dio origen.

Recomendaciones para la Especificación de Requerimientos

➢ Al especificar requerimientos es recomendable tener presentes las siguientes metas:
    - Toda persona que lee el requerimiento lo interpreta de la misma manera que cualquier otro lector
    - La interpretación de los lectores coincide con la interpretación de la persona que especificó el requerimiento

➢ Adoptar y mantener un **estilo de escritura** uniforme
    - Brindar especificaciones breves y concisas siempre que sea posible
    - No ligar la prioridad de los requerimientos a la forma en la que están escritos.
      (Ej: el sistema “hará”/“debe hacer”/“puede hacer” X cosa)
    - No usar debería, deberá, para distinguir lo indispensable de lo deseable
    - Evitar el uso de sinónimos para referenciar a un mismo término
    - Usar voz activa cuando sea posible, para distinguir el sujeto que lleva a cabo una acción
    - No combinar múltiples requerimientos en una misma especificación.
      El uso de conectores como “y”, “o”, “además”, “también”, “salvo que”, etc. puede ser un indicio de combinación de múltiples requerimientos.

➢ Los requerimientos deben ser especificados con un **nivel de detalle** que provea a los desarrolladores y testers la información que necesitan
    - Mayor nivel de detalle: necesario cuando el producto es para un cliente externo, cuando se tercerizan etapas del proceso de desarrollo, o cuando el equipo de trabajo se encuentra disperso geográficamente, entre otros.
    - Menor nivel de detalle: puede adoptarse cuando el producto corresponde a un desarrollo interno, cuando los clientes y usuarios están altamente involucrados. o cuando los desarrolladores y testers son muy experimentados, entre otros.

➢ Evitar introducir **ambigüedades** en las especificaciones
    - Utilizar términos de manera consistente, siguiendo las definiciones del glosario
    - Evitar el uso de términos ambiguos o cuya interpretación es subjetiva
    - Evitar el uso de expresiones del tipo “A/B”
    - Considerar adecuadamente los rangos de valores, incluyendo los valores límite

    - Evitar el uso de negación.
- ➢ Evitar la **incompletitud** en las especificaciones
    - Analizar operaciones simétricas
    - Analizar expresiones con lógica compleja o múltiples combinaciones de valores
    - Analizar situaciones excepcionales

Priorización de Requerimientos

La **prioridad** puede pensarse como un atributo que establece una relación de orden entre los requerimientos de acuerdo a su importancia o urgencia

→ La **priorización** de requerimientos es especialmente importante cuando
- Los recursos son escasos
- Los requerimientos son complejos
- Existen múltiples criterios, e intereses diversos o contrapuestos

Captura

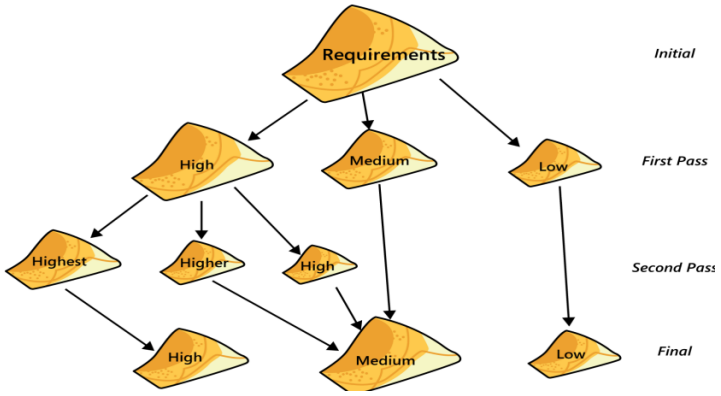
- *¿Hay alguna otra forma de satisfacer la necesidad que aborda este requerimiento?*
- *¿Cuáles serían las consecuencias de omitir o postergar este requerimiento?*
- *¿Cómo impacta a los objetivos del negocio si este requerimiento no se implementa por muchos meses?*
- *¿Por qué la postergación en la implementación de este requerimiento afectaría a algún cliente?*
- *¿La implementación de este requerimiento en esta iteración amerita la postergación de todos los demás requerimientos que poseen la misma prioridad?*

Técnicas

La elección de las técnicas de priorización muchas veces depende del entorno u organización para la cual se está desarrollando el producto.

- In/Out
  Es la técnica más simple, en la que **se solicita** a las partes interesadas que tomen una lista de requerimientos y tomen una decisión binaria
    - ¿El requerimiento queda dentro o fuera de la iteración actual?
- Comparación de a Pares y Asignación de Ranking
  Similar a la técnica de priorización de atributos de calidad mediante la construcción de una tabla de doble entrada
  Desventajas
    - Realizar las comparaciones es tedioso y hasta inviable cuando se tiene una gran cantidad de requerimientos
    - Asignar un ranking individual a los reque. no tiene mucho sentido. Alternativa, agrupar requerimientos y priorizar los grupos por iteración

- <u>Escala de 3 Niveles</u>
  Se definen 3 categorías o niveles a ser considerados por las partes interesadas: alta, media, baja



- <u>Doble Escala: Importancia y Urgencia</u>
  Cada requerimiento puede considerarse **importante** o no tan importante para alcanzar los objetivos del negocio, y como **urgente** o no tan urgente. Estas características son las que determinan la prioridad

	Important	*Not So Important*
Urgent	High Priority	Don't Do These!
Not So Urgent	Medium Priority	Low Priority

- <u>MoSCoW</u>
  Se consideran 4 categorías para la clasificación de requerimientos
  Must: el requerimiento es determinante para el éxito del proyecto. **Debe** satisfacerse
  Should: el requerimiento es importante para el éxito del proyecto, pero no determinante. **Debería** satisfacerse, de ser posible
  Could: el requerimiento es una característica deseable, pero **puede ser** postergado o eliminado. Implementarlo solo si los recursos lo permiten
  Won't: El requerimiento **no será** implementado en la iteración actual, pero puede incluirse a futuro
- <u>$100</u>
  Se distribuyen $100 imaginarios a cada miembro del equipo de priorización. Cada miembro del equipo "invierte" su dinero en los requerimientos que considera más importantes para ser incluidos en la iteración actual. La prioridad queda determinada por la inversión total,

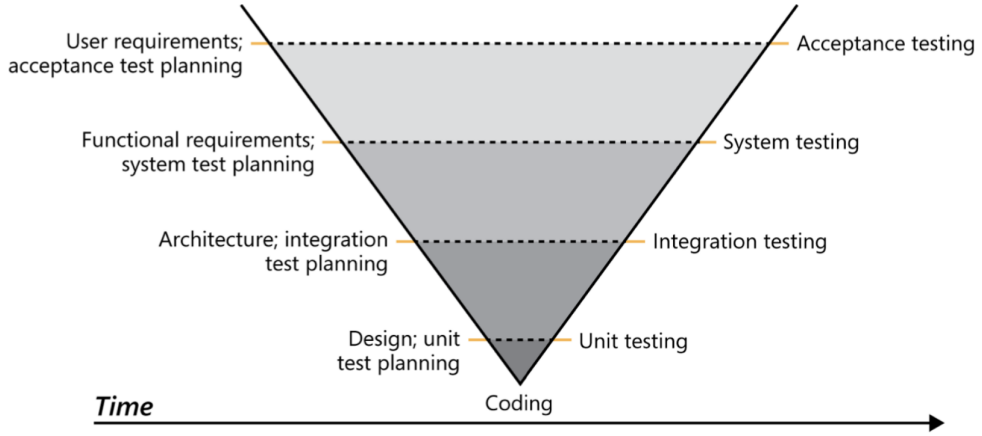
Validación de Requerimientos

Objetivos

- Los requerimientos son *necesarios, identificables, correctos, completos, factibles y verificables → individuales*
- El conjunto de requerimientos es *consistente, completo y trazable → globales*

Los requerimientos proveen las bases para el diseño e implementación del sistema

Modelo de Desarrollo de SW en V



Técnicas

→ Prototipos

- Son una herramienta de validación que ayuda a hacer más tangibles los requerimientos
- Permiten evaluar la **completitud** y **factibilidad** de los requerimientos
- Permiten validar que todos los participantes comparten una misma interpretación para los requerimientos especificados
- Permiten favorecer el diseño de casos de prueba para los *requerimientos funcionales*

→ Test Conceptual

- Pueden construirse a partir de los *requerimientos funcionales, modelos de análisis (diagrama de clases, diagrama de estados, diccionario de datos..)* y *prototipos*
- Son independientes de la implementación, deben cubrir el *flujo normal* y los *flujos alternativos y excepcionales* de los CU. Y deben cubrir todos los flujos del proceso y todos los caminos de decisión

- Ej. Sistema de Compra Online - CU: *"Ver Orden Realizada"*
  El usuario ingresa el número de orden, la orden existe y la realizó el usuario.
  Resultado esperado: mostrar detalles de la orden.

  El usuario ingresa el número de orden, la orden no existe.
  Resultado esperado: Mostrar mensaje "No existe una orden con ese número"

→ Revisión Informal

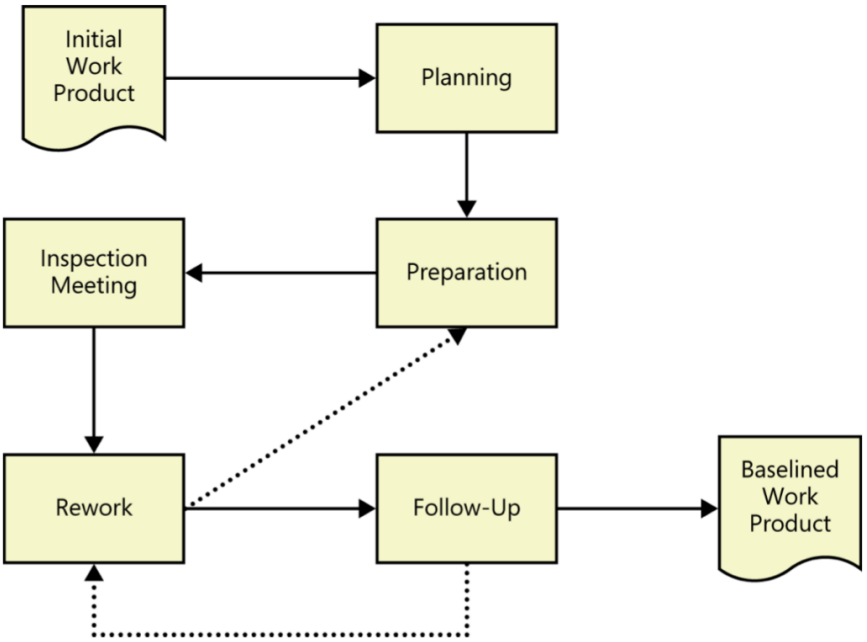
- Reuniones de revisión informal son útiles para informar a otros miembros del equipo sobre el producto y recolectar feedback des-estructurado
- Permiten identificar grandes errores, inconsistencias y ausencias
- Tipos:
  - Revisión de escritorio → el *analista* solicita a un colega que revise a su trabajo
  - Vistazo rápido → el analista invita a varios colegas en forma simultánea para que examinen en forma conjunta su trabajo
  - Presentación → el analista organiza una presentación estructurada, sin llegar a ser rigurosa, destinada a que un grupo de colegas la revisen y posteriormente formulen comentarios

→ Revisión Formal

- Tiene una estructura más rígida y sigue un procedimiento sistemático
- Se genera un reporte que incluye una *descripción del material examinado,* la *nómina de revisores,* y la *opinión de los revisores* sobre la aceptación o rechazo de cada requerimiento
- Todos los miembros del equipo de revisión formal son responsables por la calidad de la revisión
- *Inspección*
  - Es una técnica que brinda muchos beneficios
  - Favorece al desarrollo de software de calidad
  - Puede aplicarse a requerimientos, documentos de diseño, código fuente y planes de proyectos
  - Tiene un costo alto en recursos y tiempo
  - **Participantes**:
    → Autor de las especificaciones de requerimientos bajo inspección
    → Fuentes de los requerimientos bajo inspección
    → Personas que tomarán como insumo los requerimientos bajo inspección
    → Responsables de los sistemas con los que interactúa el producto en desarrollo
  - **Roles**:
    → Autor
    → Moderador
    → Lector
    → Escriba
  - **Prerrequisitos** ⇨ documento los debe cumplir

→ Documento sigue la plantilla adoptada por el equipo de trabajo y no tiene errores evidentes de ortografía, gramaticales o de formateo
→ Los números de línea y otros identificadores están impresos en el documento para facilitar las referencias a secciones específicas
→ Todos los aspectos no resueltos están marcados como TBD o similar
→ El moderador no encontró > 3 errores graves en una breve lectura de una porción representativa del documento

- **Proceso**



→ **Planificación** → realizada por el autor y el moderador
  - Se determinan participantes y roles, documentos a inspeccionar, material necesario previo a la inspección, tiempo necesario para la inspección, fecha de la inspección

→ **Preparación**
  - El autor comparte con los inspectores información de background para que entiendan el contexto de la información bajo inspección
  - Cada inspector analiza los documentos utilizando una *checklist* en búsqueda de errores comunes u otras técnicas
  - El 75% de las anomalías se detectan durante esta etapa
  - Planificar invertir en la preparación al menos la mitad de tiempo previsto para la inspección
    Checklist de defectos preguntas

Completeness

- ❑ Do the requirements address all known customer or system needs?
- ❑ Is any needed information missing? If so, is it identified as TBD?
- ❑ Have algorithms intrinsic to the functional requirements been defined?
- ❑ Are all external hardware, software, and communication interfaces defined?
- ❑ Is the expected behavior documented for all anticipated error conditions?
- ❑ Do the requirements provide an adequate basis for design and test?
- ❑ Is the implementation priority of each requirement included?
- ❑ Is each requirement in scope for the project, release, or iteration?

Correctness

- ❑ Do any requirements conflict with or duplicate other requirements?
- ❑ Is each requirement written in clear, concise, unambiguous, grammatically correct language?
- ❑ Is each requirement verifiable by testing, demonstration, review, or analysis?
- ❑ Are any specified error messages clear and meaningful?
- ❑ Are all requirements actually requirements, not solutions or constraints?
- ❑ Are the requirements technically feasible and implementable within known constraints?

Quality Attributes

- ❑ Are all usability, performance, security, and safety objectives properly specified?
- ❑ Are other quality attributes documented and quantified, with the acceptable trade-offs specified?
- ❑ Are the time-critical functions identified and timing criteria specified for them?
- ❑ Have internationalization and localization issues been adequately addressed?
- ❑ Are all of the quality requirements measurable?

Organization and Traceability

- ❑ Are the requirements organized in a logical and accessible way?
- ❑ Are all cross-references to other requirements and documents correct?
- ❑ Are all requirements written at a consistent and appropriate level of detail?
- ❑ Is each requirement uniquely and correctly labeled?
- ❑ Is each functional requirement traced back to its origin (e.g., system requirement, business rule)?

Other Issues

- ❑ Are any use cases or process flows missing?
- ❑ Are any alternative flows, exceptions, or other information missing from use cases?
- ❑ Are all of the business rules identified?
- ❑ Are there any missing visual models that would provide clarity or completeness?
- ❑ Are all necessary report specifications present and complete?

→ **Reunión**

- El lector describe cada requerimiento a partir de la interpretación que hace cada ítem del documento
- El escriba registra la interpretación y los comentarios de los demás inspectores
- Luego de examinar la documentación, los inspectores pueden acordar que se acepta el documento tal como está, que se necesitan realizar cambios menores, o que se necesitan cambios mayores
- La reunión no debería de durar más de dos horas. Si se necesita más tiempo programar otra reunión

→ **Rehacer**

- Todo control de calidad va a revelar anomalías
- El analista debe estar preparado para rehacer parte de los requerimientos luego de una reunión de inspección
- Defectos no corregidos implicarán mayores costos a futuro, con lo cual es el momento para resolver ambigûedades, eliminar la imprecisión, y sentar bases para un desarrollo exitoso

→ **Seguimiento**

- El moderador u otro participante designado trabaja con el autor para asegurar que todos los aspectos a resolver fueron abordados y que se corrigieron todos los errores marcados

- Esta etapa corresponde al cierre del proceso y permite que el moderador determine si se cumplió el criterio de terminación
- Como resultado de esta etapa pueden requerirse nuevas modificaciones (cambios parciales o incorrectos)

→ **Criterio de terminación** → moderador verifica los siguientes aspectos, antes de dar por finalizado el proceso de inspección

- Ningún documento tiene errores gramaticales o de ortografía
- Todos los aspectos planificados han sido resueltos
- Todos los cambios propuestos para los requerimientos han sido realizados y documentados
- Todos los documentos satisfacen la estructura de las plantillas
- El moderador no encuentra anomalías evidentes luego de examinar documentos

Recomendaciones

- Planificar la inspección, enfocándose en primera instancia en las secciones del documento que resulten relevantes. El documento no necesariamente tiene que leerse en forma secuencial.
- Priorizar la revisión de requerimientos funcionales que se usarán con frecuencia o provocan riesgos.
- En las reuniones sucesivas asegurarse de revisar con mayor atención las secciones nuevas o que se han ampliado desde la revisión anterior.
- Comenzar las inspecciones en forma temprana, no cuando se está cercano a la finalización del desarrollo de requerimientos.
- Asignar suficiente cantidad de días para las reuniones de inspección y suficiente cantidad de tiempo para cada reunión.
- Considerar que la mayoría de los participantes tienen otras responsabilidades.
- Verificar que cada inspector tiene suficiente información acerca del proyecto y su contexto.
- Limitar el número de revisiones de un mismo documento a un máximo de 3 por inspector

→ Test de Aceptación a partir de Criterios de Aceptación

- Permiten validar no solo los requerimientos individuales sino aspectos generales de la solución propuesta
- Definen las condiciones mínimas para que el producto se considere listo para salir al mercado
- ¿Cómo evaluarías que el producto cumple tus necesidades?
- Los criterios deben especificarse de manera tal que distintas personas objetivas lleguen a la misma conclusión sobre si se cumplieron o no
- Abordan los siguientes aspectos:

    - El producto ofrece todas las funcionalidades de alta prioridad
    - El producto satisface características no funcionales y las métricas de calidad esenciales
    - No quedan errores o ausencias por resolver o modificar, vinculados a los requerimientos prioritarios
    - El producto cumple todas las condiciones que imponen las leyes y regulaciones
    - Se tiene soporte para los requerimientos de transición e infraestructura
- Los **test de aceptación** se definen en base a los criterios de aceptación
- Validan que el producto o solución propuesta hace lo que debe
- Debe repetirse cada vez que se agrega funcionalidad y se completa una iteración en el proceso de desarrollo de requerimientos
- Los **test de aceptación** deben cubrir tanto *requerimientos funcionales* como *aspectos no funcionales* del producto
- Tests sobre aspectos funcionales deben quitar relevancia a los flujos alternativos menos frecuentes
- -para aspectos no funcionales, los tests deben asegurar que las metas de performance son satisfechas, que el sistema verifica los estándares de usabilidad establecidos, y que las medidas de seguridad adoptadas son adecuadas

Reuso de requerimientos

- Aprovechar el trabajo realizado previamente *(en el mismo proyecto o en otros)*
- Brinda beneficios que permiten avanzar hacia la solución más rápidamente

Beneficios

- Mejoras en la productividad del equipo
- Aceleración de los ciclos de revisión, prueba y aprobación, conduciendo a entregas más rápidas
- Mayor consistencia y uniformidad dentro de la aplicación y entre aplicaciones o sistemas relacionados o similares
- Reducción de costos
- Reducción de errores

- Puede provocar *riesgos*

Dimensiones del Reuso

→ **Grado de Reuso:** Nivel en el que se reutilizan los componentes
- Cantidad y tipo de material a ser reutilizado

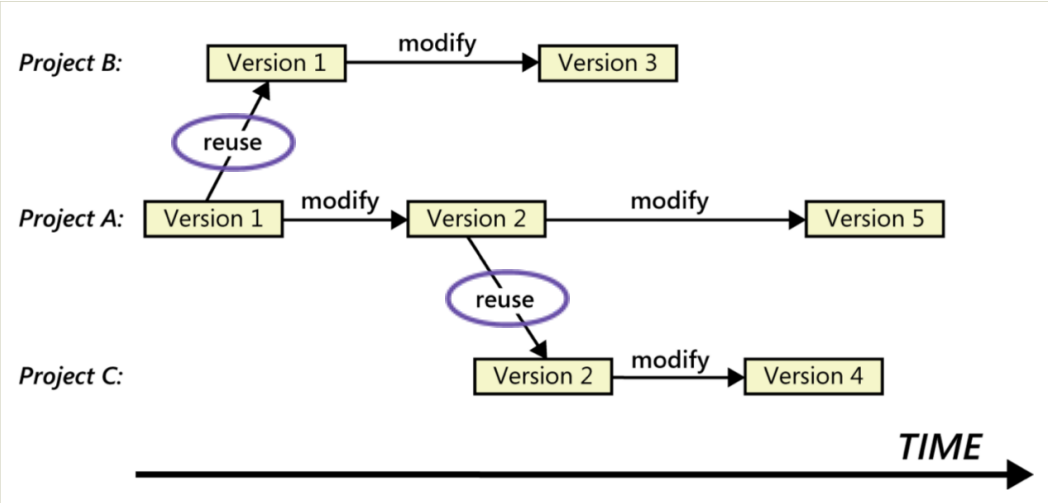
- Requerimiento individual.
- Requerimiento y sus atributos (ej: prioridad, fuente).
- Requerimiento, sus atributos e información asociada (ej: definiciones del glosario, tests de aceptación, reglas del negocio).
- Conjunto de requerimientos relacionados.
- Conjunto de requerimientos relacionados y piezas de diseño.
- Conjunto de requerimientos relacionados, piezas de diseño, implementación y tests.

→ **Grado de Modificación:** Cuánto debe modificarse el componente para ser reutilizado en un nuevo entorno

- Cantidad de cambios que es necesario realizar para poder reutilizar los requerimientos
  - Sin cambios.
  - Modificaciones en los atributos o los valores de los atributos.
  - Modificaciones en la especificación.
  - Modificaciones en la información asociada.

→ **Mecanismo de Reuso:** Metodología empleada para efectivizar el reuso

- Forma en la que simplemente el reuso de requerimientos
  - Copiar y pegar desde otra especificación del mismo proyecto.
  - Copiar y pegar de un repositorio de requerimientos reusables.
  - Utilizar referencias: única copia referenciada, o referenciar especificaciones de otros proyectos.
    - Utilitarios de propósito general.
    - Herramientas de administración de requerimientos.
- Historial de Versiones



- **Escenarios donde es posible identificar oportunidades de reuso de requerimientos**
    - Dentro de un mismo producto
    - Dentro de una línea de productos relacionados
    - Dentro de una organización
    - Dentro de dominio de aplicación
    - Dentro de un entorno operativo
    - Al desarrollar sistemas de mejora o reemplazo

Factores de éxito

- Uso de repositorios o herramientas de administración de requerimientos
- Especificar requerimientos de calidad y mejorar los requerimientos reutilizados cuando sea posible
- Mantener accesibles los links de trazabilidad de los requerimientos
- Uso de glosarios y terminología común a todos sus proyectos de una misma organización
- Fomentar cultura de reuso → especificar requerimientos de calidad con potencial de reuso, y reuso de requerimientos
- Uso de *patrones* o *plantillas,* y selección de un nivel adecuado de requerimientos

Barreras u obstáculos

- Requerimientos no documentados, o su documentación no está disponible, está incompleta o desactualizada
- Resistencia a usar requerimientos genéricos o escritos por otras organizaciones
- Resistencia a usar requerimientos reutilizables argumentando que no aplican al proyecto u organización actual
- Diferencias en el estilo de escritura, técnicas de representación utilizadas, convenciones adoptadas, vocabulario
- Diferencias en la categorización y organización de requerimientos
- Requerimientos muy específicos para un tipo de proyecto o entorno
- Los requerimientos son propiedad del cliente que encomendó el producto inicial y no pueden ser reusados

Metodologías ágiles

Existe una profunda relación entre los modelos de **ciclo de vida** y el **desarrollo y administración de requerimientos**

Modelos predictivos

(son la contra de las metodologías ágiles)
→ Conducidos por planes, intentan anticipar los cambios para minimizar los riesgos del proyecto

- La especificación de requerimientos es exhaustiva, y se completa antes de comenzar con la implementación o incluso antes de la etapa de diseño
- El equipo de desarrollo se prepara para administrar los cambios diseñando una **arquitectura robusta y flexible** que permita adaptarse con facilidad

Modelos Adaptativos

→ Conducidos por los cambios, intentan reducir el impacto de los cambios
- Buscan administrar los cambios durante el *desarrollo del producto*
- Los cambios pueden producirse en el entorno externo, interno a la organización o en el entorno operativo
- Son adecuados en ambientes dinámicos en los que los requerimientos son volátiles
- Son adecuados cuando el sistema se desarrolla a partir de una idea de producto para el cual los requerimientos son inciertos

Las ***metodologías ágiles*** se basan en los modelos iterativos incrementales pero evolucionan hacia un **modelo adaptativo**

- Se caracterizan por **iteraciones breves** con entregas rápidas
- Pequeños incrementos, cada uno de los cuales incluye unas pocas características prioritarias para un grupo de usuarios

Clientes y usuarios

→ Tienen un rol protagónico en todo el proceso porque cada iteración es muy breve
→ Se requiere una selección rigurosa para garantizar usuarios representativos, donde se integran al **equipo de desarrollo** por lo que se requiere un entrenamiento mayor

Historias de usuarios

Narración breve, realizada normalmente por un usuario (directo o indirecto), que describe una característica del nuevo sistema que desde su perspectiva es valiosa.
→ Escrita desde la perspectiva del usuario
→ Debe incluir su rol (el del usuario), la necesidad y su finalidad
→ Los *requerimientos* se especifican a través de historias de usuarios

Como [Rol], **necesito** [Descripción de la característica], **con la finalidad de** [Descripción de la consecuencia]

Ej.: **Como** operador hotelero, **necesito** conocer las tarifas de los principales competidores **con la finalidad de** establecer tarifas que me permitan mantener la ocupación de las habitaciones por encima de la media de ocupación de la competencia

Características

Breves y simples → no deben ser una composición de historias de usuario
Independientes → de modo que puedan desarrollarse por separado
Negociables → no son contratos, son *flexibles*
Valorables → el valor lo ponen los usuarios o compradores del software, nunca los desarrolladores
Estimables → se debería de poder estimar bien el tiempo de producción
Testeables → las historias se deben poder verificar

Superposición

→ Características comunes deben quedar capturadas en una única historia de usuario
Ej.:

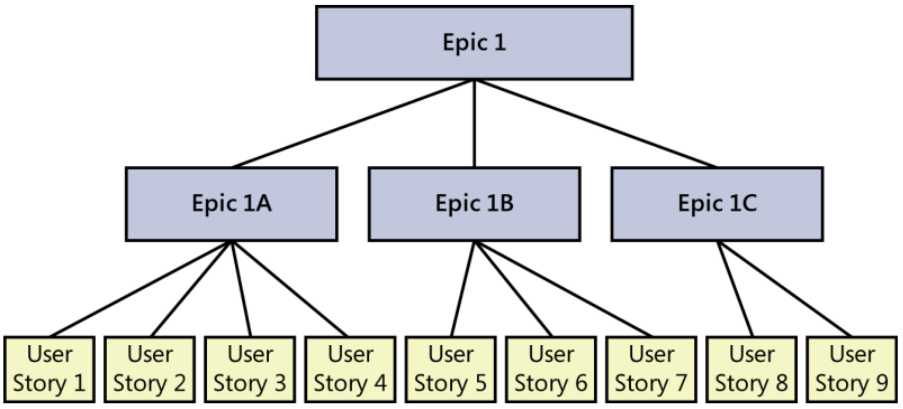
1) **Como** lector de un blog **necesito** poder responder a un comentario y recibir un alerta ante un comentario de otro lector **con la finalidad de** comunicarme con el administrador del blog y otros lectores
2) **Como** lector de un blog **necesito** poder agregar un comentario a una entrada y responder a un comentario de otro lector **con la finalidad de** comunicarme con el administrador del blog y otros lectores.

→ Se *"desglosan"* en:

1) **Como** lector de un blog **necesito** poder responder a un comentario de otro lector **con la finalidad de** comunicarme con el administrador del blog y otros lectores.
2) **Como** lector de un blog **necesito** poder agregar un comentario a una entrada **con la finalidad de** comunicarme con el administrador del blog y otros lectores.
3) **Como** lector de un blog **necesito** recibir una alerta ante un comentario de otro lector **con la finalidad de** conocer los comentarios de otros usuarios tan pronto los realizan.

Historias épicas

Historia de usuario que es **demasiado grande** para implementarla completamente en una sola iteración
→ Deben dividirse en otras más pequeñas que puedan desarrollarse en una iteración

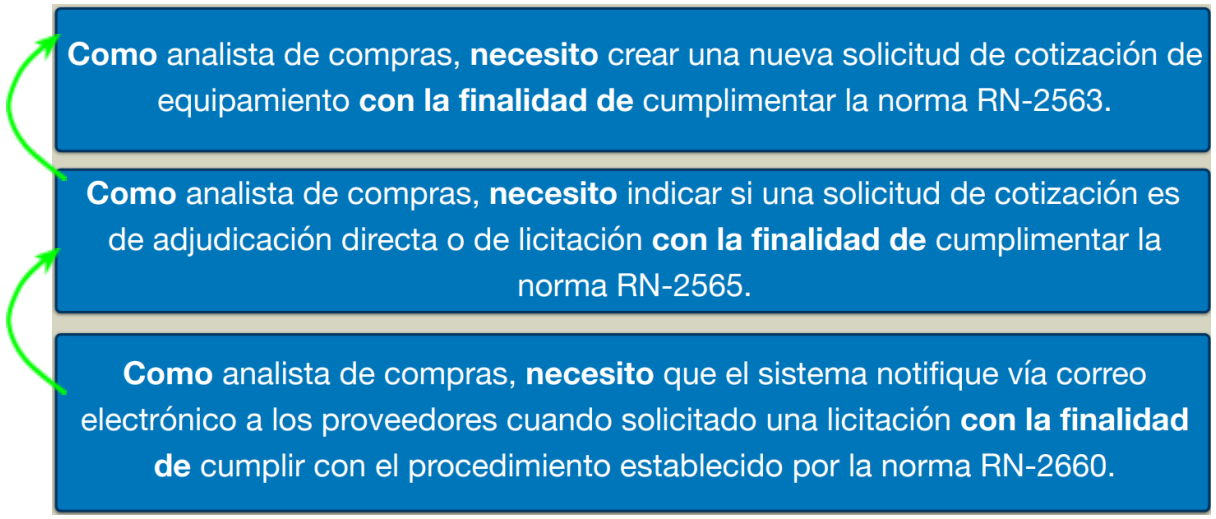


Prioridades

A partir del conjunto de **historias de usuario** se establecen ***prioridades***
→ Las historias con **menor nivel** de prioridad deben estar alineadas con los **objetivos del negocio** y resultar valiosas a los usuarios
→ Se seleccionan las historias con **mayor prioridad** para la próxima iteración y las demás conforman la **lista de pendientes**

Dependencias de orden

- A veces la **prioridad** entre las historias de usuario está **determinada** por **dependencias** de orden entre ellas
- Ej.:



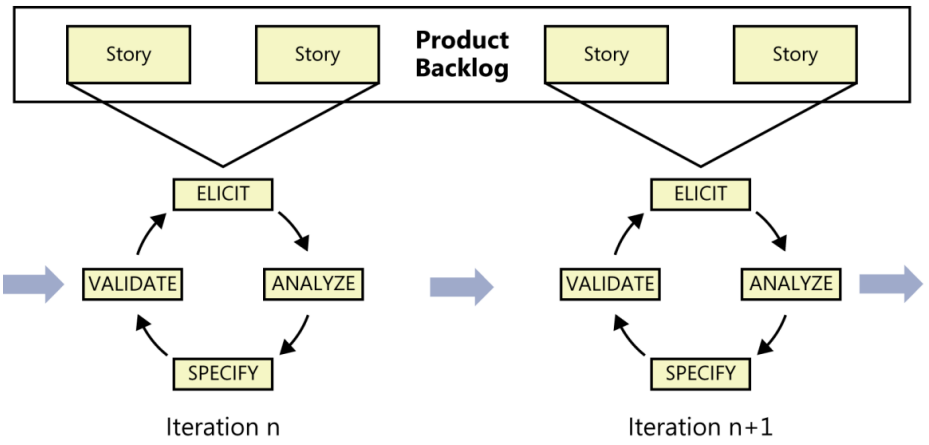
Lista de pendientes

- Está compuesto fundamentalmente de *historias de usuarios*
- Cuando termina una iteración el equipo retoma la lista de pendientes y la actualiza
- Cada proyecto debería **mantener** una **única lista de pendientes**
- Se revisan las prioridades, se seleccionan las historias con mayor prioridad y así siguiendo

Alcance del proyecto

El alcance de cada iteración queda definido por un conjunto de historias de usuarios seleccionadas del backlog, en base a las prioridades y la capacidad de entrega del equipo de desarrollo

- En cada iteración probablemente se identifiquen nuevos requerimientos que se seleccionarán para esa iteración o se incorporarán a la lista de pendientes para futuras iteraciones
- La duración total del proyecto va a depender de la cantidad de iteraciones y con frecuencia no se puede calcular al inicio



Documentación

Desarrollo de un sistema requiere la especificación de los **objetivos del negocio**, para ambos modelos (predictivo y adaptativo)
Objetivos del negocio: Especifican el beneficio que se espera alcanzar con el producto y establecen el marco dentro del cual se asignan **prioridades** y se resuelven conflictos

→ El principal documento es el **backlog**. Puede incluir también modelos del circuito de negocios, reglas del negocio y defectos a corregir

- Los efectos a corregir pueden modelarse también como **historias de usuarios** que describen características a modificar o funcionalidades a reemplazar (se les asigna prioridad también)

Responsabilidades

De **clientes** y **usuarios**

- Estar **comprometidos** en el proceso. Comparten la responsabilidad de la calidad del producto
- Establecer **prioridades** entre los requerimientos, seleccionando unas pocas funcionalidades para el alcance
- Definir **criterios de aceptación** que se aplicarán en la etapa de verificación

De los miembros del **equipo de desarrollo**

- Seleccionar usuarios representativos
- Formular *historias de usuario*
- Asignar prioridades en función de los objetivos del negocio
- Definir tests de aceptación
- Moderar las negociaciones
- Evaluar el impacto de los cambios

Del **product owner** → cliente o usuario especial

- Aceptar o rechazar el alcance del proyecto para cada iteración
    - El conjunto de *historias de usuarios*, cada una de las cuales incluye el criterio de aceptación y el test de aceptación
- Establecer la prioridad cuando hay conflictos
- Aceptar o rechazar la entrega en cada iteración, luego de verificar el producto de acuerdo a los test de aceptación

Gestión de cambios

Algunas metodologías ágiles destinan algunos días al desarrollo de requerimientos y una vez establecidos, la iteración se completa sin cambios, siguiendo con las etapas de modelos en cascada
→ En este caso, cualquier propuesta de cambio se tratará en la siguiente iteración
Otras metodologías ágiles admiten que durante una iteración se incorporen nuevas *historias de usuario* que provoquen cambios en los requerimientos o en las prioridades

Administración de Requerimientos

Incluye todas las actividades destinadas a mantener la **integridad, consistencia, precisión** y **vigencia de los requerimientos** a través del proyecto

- Control de versiones
- Seguimiento de estado de los requerimientos
- Gestión de cambios
- Rastreo de requerimientos

Acuerdos y convenciones

Definición de las actividades que le quipo debe llevar a cabo como parte del proceso de administración de requerimientos y establecimiento de:

- Herramientas, técnicas y convenciones para distinguir las versiones de requerimientos (individuales y conjuntos)
- Mecanismo para la aprobación de nuevos requerimientos
- Mecanismo para manejar la propuesta, evaluación, negociación y comunicación de nuevos requerimientos o cambios en los ya existentes
- Mecanismo de evaluación de impacto de los cambios propuestos
- Atributos y estados de los requerimientos, y procedimiento de seguimiento de estado
- Responsable de actualizar la información de rastreo de los requerimientos
- Mecanismo de identificación y resolución de conflictos
- Mecanismo de actualización del plan del proyecto en función de los cambios en los requerimientos
- Uso de herramientas de administración de requerimientos

Atributos de los requerimientos

Cada requerimiento puede pensarse como un objeto con propiedades que lo distinguen de otros requerimientos
(Además de su especificación) Cada requerimiento tiene un conjunto de atributos que determinan su contexto
Ej.:

- Identificador
- Autor
- Fecha de creación
- Responsable de la última versión
- Fecha de última modificación
- Número de versión actual

- Prioridad
- Estado
- Fuente del requerimiento
- Racionalidad o justificación
- Número de iteración asignado
- Participante referente para la toma de decisiones
- Mecanismo de validación del criterio de aceptación

- Utilizar una cantidad de atributos puede resultar inconveniente para el equipo de trabajo
- Posiblemente quedan *valores sin definir*, y el equipo no pueda utilizar la información de los atributos de manera efectiva
- Se recomienda comenzar con un conjunto reducido de atributos, y añadir otros por necesidad siempre y cuando se determine que aportan un valor agregado

Racionalidad o Justificación
- Justifica por que el requerimiento es relevante para el producto
- Es útil para asignar **prioridades** y, por lo tanto, para replanificar o recortar requerimientos cuando surgen cambios y se agregan nuevos requerimientos
- Un nuevo requerimiento puede provocar también que otro anterior deje de tener justificación y por lo tanto pueda eliminarse

Estado

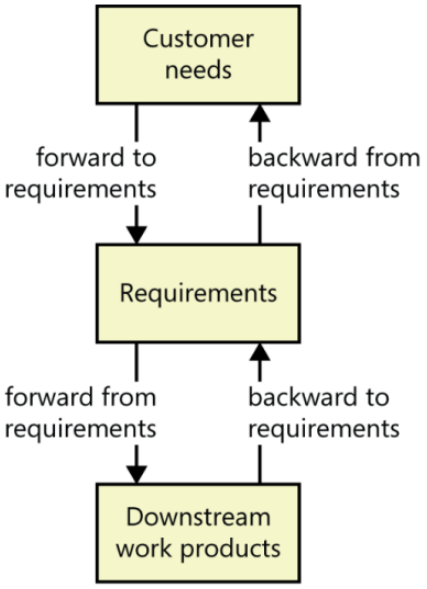
Este atributo permite realizar un seguimiento de estado y así determinar el grado de avance del proyecto en cada momento de tiempo
Posibles estados
- **Propuesto**: Se capturó un requerimiento originado por una fuente autorizada
- **Es análisis**: Se está trabajando activamente en el requerimiento, determinando su prioridad, y se busca identificar inconsistencias o ambigüedades
- **Especificado**: Se especificó la versión inicial del requerimiento
- **Postergado**: El requerimiento había sido aprobado pero se postergó para otra entrega o iteración
- **Eliminado**: El requerimiento había sido aprobado pero se eliminó del acuerdo base
- **Rechazado**: El requerimiento se propuso pero no aprobó
- **Seguimiento de Estado**: Realizar un seguimiento del estado de los requerimientos requiere:
    - Definir el conjunto de estados posibles
        - Todos los miembros del equipo deben acordar qué significa cada estado, y las condiciones que deben satisfacerse para la transición de un estado a otro
    - Actualizar el estado de cada requerimiento
    - Analizar periódicamente la distribución de estados

Rastreo de requerimientos requiere **documentar las dependencias** y enlaces lógicos entre los requerimientos y otros elementos del sistema
Para que los requerimientos sean **rastreables** o **trazables** deben tener una etiqueta persistente que los identifique unívocamente

Rastreo de Requerimientos

La información de rastreo permite enlazar un requerimiento hacia atrás y hacia adelante, desde su origen hasta su implementación y viceversa, definiendo cuatro situaciones de rastreo
Componentes hacia las que se puede rastrear:

- otros requerimientos, reglas del negocio, arquitectura y otros componentes de diseño, código fuente, componentes de testing, etc



Links de Rastreo

Multiplicidad
Uno a uno
Uno a muchos
Muchos a muchos

Beneficios

El **rastreo de requerimientos** permite demostrar el cumplimiento de una especificación, contrato o regulación
Brinda diversos beneficios en el proceso de desarrollo, incrementando la calidad del producto, reduciendo los costos de mantenimiento, y facilitando el reuso. Entre otras cosas permite:

- Encontrar requerimientos ausentes
- Encontrar requerimientos innecesarios
- Certificar el cumplimiento de reglamentaciones o estándares
- Evaluar el impacto de los cambios
- Mantener el código
- Reingeniería o reemplazo de componentes de código
- Reusar componentes
- Verificar componentes de código

Matriz de Trazabilidad

La forma más común de representar los link entre los requerimientos y otro componentes del sistema es la matriz de trazabilidad o tabla de trazabilidad
Alternativas

User requirement	Functional requirement	Design element	Code element	Test
UC-28	catalog.query.sort	Class catalog	CatalogSort()	search.7 search.8
UC-29	catalog.query.import	Class catalog	CatalogImport() CatalogValidate()	search.12 search.13 search.14

	Use case			
Functional requirement	**UC-1**	**UC-2**	**UC-3**	**UC-4**
FR-1	↵			
FR-2	↵			
FR-3			↵	
FR-4			↵	
FR-5		↵		↵
FR-6			↵	

- Los links pueden ser entre:
    - Requerimientos del mismo tipo
    - Requerimientos de distinto tipo
    - Requerimientos de un tipo y test

Gestión de cambios

En el **mundo ideal** todos los requerimientos están especificados y documentados antes de que comience el diseño del sistema, y se mantienen estables durante todo el proceso de desarrollo
En el **mundo real** durante el desarrollo del producto se generan oportunidades en el mercado, cambios en las reglamentaciones o nuevas necesidades, se detectan omisiones o inconsistencias

Problema → El alcance inalcanzable

- Si los requerimientos cambian y los tiempos se modifican, el proceso desarrollo se alarga y aumenta la probabilidad de que surjan otros cambios
- Si se aceptan cambios sin ajustar la planificación, es muy probable que no pueda cumplirse
    - En ambos casos la entrega final puede ir demorándose, incluso indefinidamente

Aspectos a evitar

- El cliente propone cambios y las modificaciones se atienden sin considerar los requerimientos de los usuarios directos
- El usuario propone cambios y se atienden sin considerar su alienación con los requerimientos del negocio por el cliente
- El desarrollador recibe propuestas de cambio y los atiende sin la participación del analista
- Los cambios se aceptan sn evaluar el impacto
- Los cambios se aceptan y no se modifican el plan del proyecto de manera acorde al impacto de los cambios
- Los cambios se aceptan y no se modifican la documentación de los requerimientos

El costo de los cambios

Aceptar un cambio cuando el producto está en desarrollo tiene un **costo**
Rechazar un cambio también un **costo**
Todos los participantes deben entender que los cambios van a ocurrir, pero es necesario estar preparados para **gestionarlos**
La **gestión del cambio** en sí misma, ya sea cuando se acepta o se rechaza un cambio, también demanda tiempo y esfuerzo

Política de Gestión de Cambios

Los administradores del proyecto deben definir y comunicar una política de gestón de cambios que establece las expectativas de **cómo** el equipo de trabajo debe **manejar las propuestas de cambios** en los requerimientos y otros componentes
Adoptar un política de gestión de cambios involucra

- Designar un Grupo de Control de Cambios (GCC)
- Definir un procedimiento y reglas para la gestión de cambios
- Seleccionar herramientas para la gestión de cambios del proyecto
- Mantener al administrador o líder del proyecto informado de las decisiones

Declaraciones

- Todos los cambios deben seguir el proceso establecido. Si una solicitud de cambio no es registrada de acuerdo al proceso, no será considerada
- No se llevará a cabo trabajo de diseño e implementación en cambios no aprobados, más allá de la exploración de factibilidad
- Solicitar un cambio no garantiza que será llevado a cabo. El **grupo de Control de Cambios** decidirá qué cambios serán implementados
- El contenido de la base de datos de cambios debe ser visible para todas las partes interesadas del proyecto
- Cada solicitud de cambio deberá ser sometida a un análisis o evaluación de impacto
- Cada cambio incorporado debe poder rastrearse a una solicitud de cambio aprobada
- Debe registrarse la justificación detrás de la aprobación o rechazo de cada **solicitud de cambio**

Solicitudes de Cambio

Las solicitudes de cambio en los requerimientos u otros componentes del sistema deben contener la información necesario para poder evaluar su aprobación o rechazo, y permitir tracearla a los demás elementos involucrados
El GCC pueden definir un conjunto de atributos para cada solicitud de cambio, cuyos valores serán determinados en distintas etapas del proceso

Atributos

Item	Description
Change origin	Functional area that requested the change; possible groups include marketing, management, customer, development, and testing
Change request ID	Unique identifier assigned to the request
Change type	Type of change request, such as requirement change, proposed enhancement, or defect report
Date submitted	Date the Originator submitted the change request
Date updated	Date the change request was most recently modified
Description	Free-form text description of the change being requested
Implementation priority	The relative importance of making the change as determined by the CCB: low, medium, or high
Modifier	Person who is primarily responsible for implementing the change
Originator	Person who submitted this change request
Originator priority	The relative importance of making the change from the Originator's point of view: low, medium, or high
Planned release	Product release or iteration for which an approved change is scheduled
Project	Name of the project in which a change is being requested
Response	Free-form text of responses made to the change request; multiple responses can be made over time; do not change existing responses when entering a new one
Status	The current status of the change request, selected from the options in Figure 28-2
Title	One-line summary of the proposed change
Verifier	Person who is responsible for determining whether the change was made correctly

Grupo de Control de Cambios (GCC)

→ Debe tener autoridad para decidir
→ Encargado de decidir **qué** cambios sobre los requerimientos serán aceptados, **cuáles** serán aceptados con revisiones, y **cuáles** serán rechazados. También decide **qué** defectos reportados deberán ser corregidos, y **cuándo** deberán corregirse
→ Debe establecer un mecanismo para la toma de decisiones, los roles del grupo, la cantidad de miembros requeridos para sesionar
→ Debería conformarse antes de que surjan los cambios y, de ser necesario, incorporar miembros para atender modificaciones específicas
→ Al menos uno de los miembros del grupo debe tener conocimiento profundo del negocio y la organización

Proceso de Gestión de Cambios

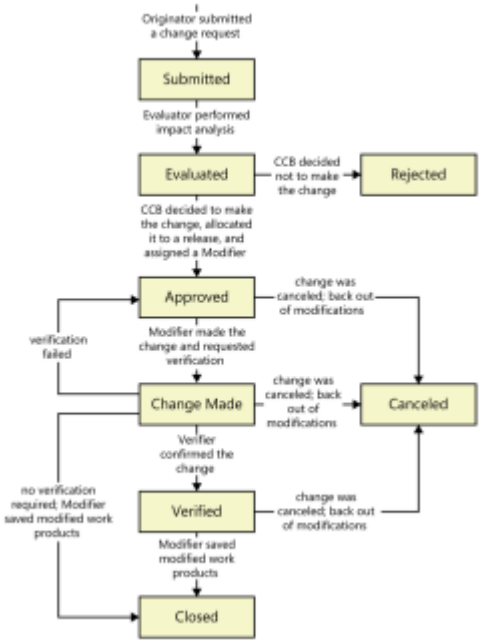
Tareas del proceso de Gestión de Cambios

→ **Evaluar una solicitud de cambio** : se considera la factibilidad técnica, el costo, y la alineación con los requerimientos del negocio. SE analizan los riesgos y el impacto de aceptar o rechazar el cambio
→ **Tomar una decisión sobre el cambio** : el GCC toma una decisión siguiendo el mecanismo de decisión adoptado. Cada cambio aprobado es priorizado y asignado a una iteración

→ **Implementar el cambio** : el encargado de la modificación actualiza los componentes necesarios para implementar el cambio en el producto, basado en la información de rastreo del requerimiento. Se actualiza la información de rastreo para reflejar los cambios realizados
→ **Verificar el cambio** : se realiza una revisión de los componentes modificados para asegurar que los entregables abordan todos los aspectos del cambio requerido

Estados de una solicitud de cambio

→ Durante las distintas etapas del proceso de gestión de cambios una solicitud de cambio en los requerimientos atraviesa distintos estados



→ Cuando se cumple el criterio definido en cada transición (por ejemplo, cuando se finaliza la tarea correspondiente), se actualiza el estado de la solicitud

Análisis de Impacto

→ El líder del GCC solicita a un conjunto de personas (analistas, desarrolladores y/o testers) la evaluación del impacto de una solicitud de cambio
→ El análisis de impacto permite comprender las implicancias de cada cambio propuesto, ayudando al GC a tomar decisiones informadas sobre qué solicitudes de cambios aprobar
→ El resultado de este análisis permite renegociar los acuerdos con los clientes

Pasos

1) Entender las implicancias de realizar el cambio. Un cambio de requerimientos frecuentemente produce un efecto en cadena, originando cambios en otros requerimientos, la arquitectura del sistema, piezas de diseño o código

   Los cambios pueden originar conflictos con otros requerimientos, o comprometer atributos de calidad como seguridad o performance.

2) Identificar los requerimientos, archivos, modelos y documentos que deberían ser creados, modificados o descartados si se acepta el cambio solicitado
3) Identificar las tareas requeridas para implementar el cambio, y estimar el esfuerzo necesario para llevarlas a cabo. Estimar el esfuerzo de modificar cada modelo de análisis, pieza de diseño o componente de software, y el esfuerzo realizado hasta el momento que se pierde si se acepta el cambio

Plantillas de evaluación

Como parte del proceso de análisis de impacto del cambio solicitado, el equipo evaluador puede utilizar checklists o plantillas que le permiten:

- Entender las implicancias de aceptar el cambio propuesto
- Identificar los componentes afectados por el cambio propuesto
- Estimar el esfuerzo requerido para implementar el cambio

Resultado de la evaluación

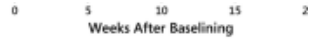
A partir del análisis de impacto para una solicitud el GCC puede

- Rechazar el cambio
- No aceptar el cambio en la iteración actual, y analizar agregar una iteración al proceso de desarrollo del producto para el cual se establecerá un nuevo acuerdo
- Aceptar el cambio en la iteración actual negociando las condiciones del acuerdo y el plan del proyecto

Evaluación de cambios

Medir la actividad de los cambios permite evaluar la estabilidad de los requerimientos, relevando oportunidades para mejorar el proceso de gestión de cambios de manera de reducir los cambios a futuro
Aspectos a medir:

- Número de solicitudes de cambio recibidas, pendientes y cerradas.



- Número acumulado de requerimientos añadidos, eliminados o modificados.
- Número de solicitudes de cambio realizadas por cada fuente de requerimientos.
- Número de cambios realizados sobre cada requerimiento desde su inclusión en el acuerdo base.
- Esfuerzo en hs. invertido para procesar, implementar y verificar las solicitudes de cambio.

Control de requerimientos

El control de versiones está ligado a la gestión de cambios
Permite identificar unívocamente las distintas versiones de los requerimientos, tanto en forma individual como de conjuntos de requerimientos
Cada miembro del equipo de desarrollo debe tener acceso a la última versión de cada requerimiento
Los cambios deben quedar documentados y ser comunicados a todos los afectados, asegurándose de que el identificador de la versión del requerimiento se actualice con cada cambio realizado

Objetivos

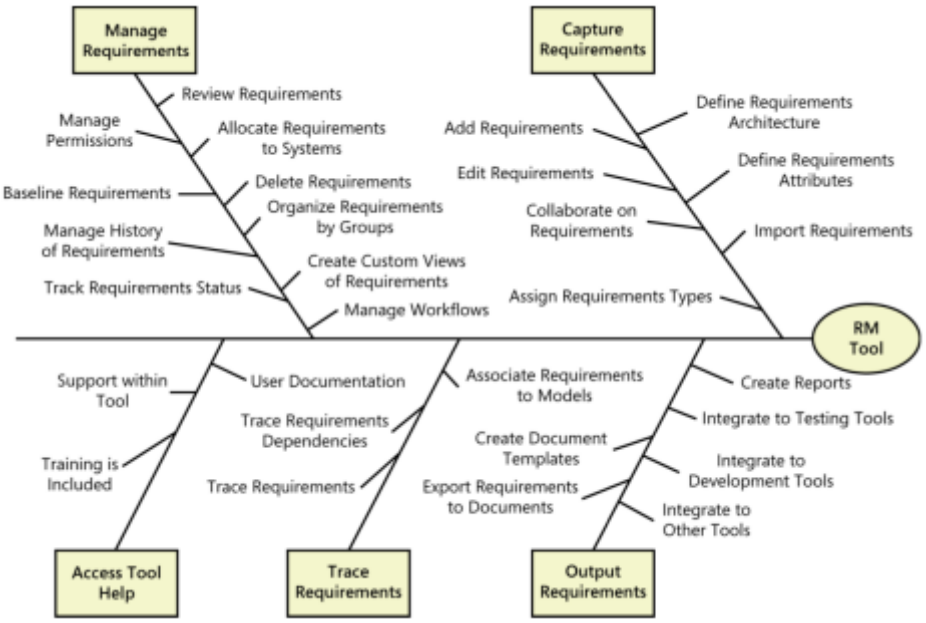
- Retener la historia de los cambios
- Hacer un seguimiento histórico de los cambios a través de las versiones de requerimientos o conjuntos de requerimientos
- Permitir deshacer cambios para restablecer un ítem que ha sido modificado

Uso de herramientas

Existen herramientas de software que integran funcionalidades para cada una de las tareas del desarrollo y administración de requerimientos

- **Herramientas** de administración de requerimientos **que integran** control de versiones, seguimiento de estado, rastreo de requerimientos y gestión de cambios
- Otras **herramientas** brindan funcionalidades **específicas** para una de estas tareas, por ejemplo: control de versiones
- También es posible adoptar convenciones y usar **utilitarios de propósito general**

Herramientas Integrales o Específicas



Uso de Utilitarios de Propósito General

Desafíos

El uso de herramientas de propósito general para la administración de requerimiento conlleva diversos desafíos:

- Mantener los documentos actualizados y sincronizados
- Comunicar los cambios a los participantes y establecer vínculos entre los requerimientos, y entre los requerimientos y otros componentes
- Rastrear el estado de los requerimientos individuales y de los conjuntos de requerimientos

Convenciones

Al decidir utilizar una herramienta de propósito general para la administración de requerimientos el equipo de trabajo debe establecer convenciones

- Mecanismo de sincronización o servicio de almacenamiento compartido
- Pautas para la generación de versiones
- Procedimiento para registrar y controlar cambios posteriores al acuerdo base
- Mecanismo de rastreo y control de versiones