**Clase 1:**

3 conceptos claves: **comprender, representar, comunicar**.

Comprender: diferentes técnicas para entender el funcionamiento del dominio.

Representación: usar un **lenguaje** técnico y formal para poder explicar el sistema.

* Lenguajes: diferentes artefactos para usar a la hora de la comunicación.

Comunicar: a través de lo anterior dicho, poder comunicarlo.

**El #1 objetivo es producir un producto que satisfaga la necesidad del cliente.**

## Proceso de desarrollo de SW

* Boceto:
  + Modelar el problema
  + Análisis de más problemas
  + Asegurar entendimiento / divergencias
  + Más barato si se descubren errores en esta etapa

### Equipo de desarrollo

* Analista
* Diseñador
* Implementador
* Manager

### Equipos del cliente

* Experto (alguien que puede o no ser el que paga, que puede ser o no el que usa el sistema, pero la característica única: es el que la tiene clara)
* Sponsor / facilitador (aceitar los mecanismos para que la rueda gire, va a ayudar)
* Usuario
* Cliente

Hay roles que se pueden fusionar.

El usuario del sistema no siempre es el cliente.

**Clase 2:**

Ingeniería de requerimientos: disciplina donde intervienen procesos, actividades, técnicas, etc, para describir, documentar y mantener. Estamos hablando de lo que obtenemos del cliente, lo que el stage holder nos dice que necesita del sistema.

Control de calidad: una vez que tengo el producto terminado, lo analizo si cumple con la función

Aseguramiento de la calidad: mientras estoy produciendo el producto asegurar que se cumplan las premisas del los requisitos, esto es mucho mas barato arreglarlo si ocurre un error.

Ingeniería de requisitos: descubrir los requisitos del sistema, luego los documento en ERS (especificación de requisitos del sistema) y luego los tengo que mantener actualizados por los cambios.

Requisitos del sistema: lo que el sistema tiene que realizar y lo que necesita, dicho por el cliente (es la salida).

Requisitos del cliente: lo que el cliente quiere que realice el sistema, no son los mismos que los requisitos del sistema, porque tal vez no sabe lo que quiere o no las sabe transmitir (es la entrada).

Nosotros trabajamos con el requisito “esencial del sistema”, es lo más importante para el sistema.

El ingeniero de requisitos transforma los requisitos del cliente en requisitos del sistema.

**Proceso de ingeniería de requisitos: Wiegiers**

Diagrama

Descripción generada automáticamente

Desarrollo de requisitos: obtención de los requisitos del sistema, saber lo que el sistema tiene que proveer.

Estas cuatro faces no se hacen lineal todo el tiempo. Pueden hacerse de diferentes formas.

* Elicitación: educir, tomar la información del cliente. No quedarse con lo que el cliente me dice, sino, lo que realmente es necesario para el sistema. Trabajar en colaboración con el cliente para descubrir lo que quiere.
* Análisis: comprender que significa toda esa información que se juntó.
* Especificación: plasmar la información, para no olvidármela y para poder compartirla, por eje, con el cliente.
* Validación: le traducimos la información al cliente para que la pueda entender y comprobamos si es lo que necesita.

Verificación: no esta en el modelo, pero lo hace el propio equipo y analiza si esta bien lo que se hizo con el proyecto.

**Estas cuatro fases no son lineales y pueden repetirse.**

La **ingeniería de requisitos** es un área dentro de sistemas / informática

* INGRESAN requisitos del cliente
* PROCESAMOS el análisis
* SALEN los requisitos del sistema **(lo que el sistema debe proveer y el cliente necesita)**

Gestión de requisitos: proceso trasversal a todas las etapas. Identifico cambios, modificando y analizando los cambios hechos.

**Proceso de ingeniería de requisitos: Sommerville**

Diagrama

Descripción generada automáticamente

* Estudio de viabilidad: se realiza antes de todas las etapas posteriores. Si se puede realizar el proyecto o aprovechar la oportunidad de negocio, desde el punto de vista económica, tecnológica o política. Producimos el primer artefacto: **el informe de viabilidad** “si es posible realizar el proyecto o no”.
* Obtención y análisis de requerimientos: la información la tomo del cliente de lo que quiere y la analizo. Produce el segundo artefacto: **modelos del sistema**, es lo que se entendido, transmitido por el cliente.

Junta la información, la analiza y realiza la especificación, pero hay una retroalimentación, esto quiere decir que le falta más información, y analizarla, para realizar una especificación más exhaustiva.

* Especificación de requerimientos: plasmar la información, para no olvidármela y para poder compartirla, por ej, con el cliente. Produce el tercer artefacto: **requerimientos del usuario y del sistema. Este artefacto es preliminar.**
* Validación de requerimientos: es mostrar lo que le produje al cliente, me dirá si está bien.

También hay retroalimentación con la especificación de requerimientos y en el peor de los casos, tendría que hacer la obtención y análisis de nuevo.

Una vez que el cliente aprueba los requisitos del sistema se produce el ultimo artefacto: **documentos de requerimientos ERS** (especificación de requisitos del sistema), lo que el sistema tiene que hacer y como lo hace (y no de qué manera).

**Proceso de ingeniería de requisitos: profe facultad**

Tabla

Descripción generada automáticamente

**Clase 3:**

Requisitos de alto nivel: expresados coloquialmente, en términos muy generales, extensión breve, no hay una forma estructurada.

Sirve para permitir, de una forma muy rápida y sencilla, introducirnos a la funcionalidad.

Ej: el sistema único de boleto electrónico (SUBE) debera permitir abonar el costo de un viaje en un transporte público de pasajeros.

Requisitos de bajo nivel: expresados muy detalladamente, con términos muy completos, con mucha información, de extensión mediana o grande, porque abarca el requisito en profundidad. Aborda el tema de las alternativas o escenarios.

Permite que no se tenga ninguna duda sobre el requisito. Ej, no es lo mismo viajar en colectivo que en tren.

Ejemplo: el sistema único de boleto electrónico (SUBE) en el caso de un transporte ómnibus de corta y mediana distancia de pasajeros debera permitir abonar el costo de un viaje a través de un saldo precargado en una cuenta asociada a un toquen personal intransferible, de modo tal que informara el nro. de tarjeta, saldo y monto del viaje registrándose la transacción informándose la fecha, la hora y el saldo actualizado. En el caso que el monto del saldo precargado sea insuficiente para abonar el costo del viaje el sistema permitirá abonarle en negativo siempre en cuando el mismo no exceda el monto máximo.

Los requisitos del cliente se expresan en alto nivel, por que el cliente me explica que es lo que quiere.

Los requisitos del sistema se expresan en bajo nivel, porque necesitan mucha explicación.

No es excluyente, pero lo ideal es así.

El requisito describe una función que el sistema cumple o debe cumplir.

Requisitos funcionales: es algo que el sistema tiene que hacer, servicios que el sistema provee.

Requisitos no funcionales: características que de alguna u otra forma son consecuencia del sistema, como se llevan a cabo las funcionalidades. Deben cumplir con una medida cuantificable y/o calificable. Un requisito no funcional que no sea cuantificable, va a generar complicaciones.

El funcional cumple o no cumple

El no funcional si es mesurable

Somerville dice que son más importante, a veces, los no funcionales que los funcionales.

Para justificar esto Somerville dice: “Muchas veces cuando un sistema no satisface una funcionalidad, a veces el usuario encuentra otro camino para satisfacerla”.

Requisitos no funcionales por Somerville:

Texto

Descripción generada automáticamente

* **PRODUCTO**
  + De usabilidad: fácil de usar, habla de la utilización del sistema. Es complejo expresar lo que es fácil o difícil de usar.
  + Eficiencia: cumplir un objetivo con el menor costo posible (no es lo mismo eficacia que eficaz, eficaz es cumplir con un objetivo simplemente)
    - Rendimiento: del tiempo de una transacción, por ejemplo
    - Espacio: no debe ocupar más de 200 Megabytes, por ejemplo
  + Confiabilidad: tasa de fallo < 1 % sobre una media de 5000 operaciones, por ejemplo, tiempo entre fallos.
  + Seguridad: el sistema SUBE debe ser inmune a la copia de sus datos a través de un dispositivo estándar de lectura presente en el mercado, o el sistema SUBE debe garantizar que los datos personales del titular de la cuenta no sean accesibles por la empresa de colectivo
* **ORGANIZACIÓN**
  + Ambientales: ser resistente a vibraciones (por ejemplo: el sistema SUBE en el colectivo), el tipo de equipo que funciona dentro de una fábrica o equipamiento militar (por ejemplo, un equipo de comunicaciones militares no puede ser un iPhone, debe resistir más).
  + Operacionales: No sería operacional que mientras arranca el colectivo tengamos que introducir algo en un teclado, no sería operacional. Otro ejemplo es que operativamente el sistema SUBE es un sistema offline, la máquina que está arriba del colectivo tiene que resolver por si sola si cobra o no cobra el viaje.
  + De desarrollo: de qué forma se va a desarrollar ese producto que yo quiero tener. Es importante tener en cuenta que estamos en etapa de análisis, el analista NO debe decidir el lenguaje en el que se va a realizar y demás. Por ejemplo: en la republica argentina hay una ley que dice que todos los sistemas de uso público tienen que ser de licencia GNU. Entonces hay un requerimiento de desarrollo que te dice a qué software tendrías que orientarte.  
    Otro ejemplo: una empresa con convenio con Microsoft, entonces el software debe realizarse con .NET porque la organización así lo requiere.
* **EXTERNOS**
  + Regulatorios: El cumplimiento de alguna regla, ley o norma. Por ejemplo: el sistema SUBE debe respetar la ley xxxx de protección de datos personales.
  + Éticos: tiene que ver con valores o principios de la organización o dominio, podría ser: evitar la caracterización por género.
  + Legales
    - Contables: parte especifica de la normativa que regula la contabilidad, manejo de las transacciones económicas.
    - Protección/seguridad: como la protección de datos personales, habeas data, o requisitos /requerimientos.

Reglas del negocio/Reglas del dominio: es donde se desarrolla el sistema y no es parte del mismo. No es un requisito, si el sistema lo desarrolla, entonces seria un requisito funcional y una regla de dominio. Ej: no vender bebidas alcohólicas a menores de 18 años. Sera un requisito si el sistema tuviera que hacer una verificación de edad.

**Adquisición de conocimiento:**

Extracción de conocimiento: adquirimos conocimiento de documentales, videos, presentaciones, tutoriales, etc. De fuentes no humanas.

Educción de conocimiento: fuente humana, profesor, etc.

Elicitar: actitud activa. educir la información del cliente. No quedarse con lo que el cliente me dice, sino, lo que realmente es necesario para el sistema. Trabajar en colaboración con el cliente para descubrir lo que quiere.

Relevar: actitud pasiva. quedarme con lo dicho por el cliente y no indagar en lo que realmente necesita.

**Clase 4:**

## ¿Qué es la elicitacion?

Es diferente al relevamiento.

Es de escucha, análisis crítico, orientación.

Es un proceso de adquisición de conocimiento donde se aplican técnicas para entender mejor el negocio que será impactado por el proyecto, para identificar a los interesados y para mejorar o definir los requisitos (RF, RNF, RN)

* Requisitos funcionales: qué debe hacer el sistema
* Requisitos No funcionales: en torno al CÓMO (condicionamientos, características, circunstancias, entornos de aplicación) que trabajan sobre los funcionales.
* Requerimientos Negocio: entender que necesita el negocio.

Objetivo: obtener información de manera proactiva junto a las partes interesadas utilizando técnicas seleccionadas en la preparación.

Tres grandes cortes: análisis, diseño e implementación.

Pressman dice que hay 6 (grandes etapas) --ver etapas de pressman--

Técnicas de elecitacion: se utilizan para recopilar información sobre el negocio del cliente o sobre los requisitos del sistema a desarrollar.

* Técnicas de extracción de información (de fuentes no humanas)
  + Lectura
    - Documentos
    - Informes, etc.
  + Visualización multimedia
    - Videos
    - Software existente, etc.
* Técnicas de educción de información
  + Entrevistas
  + Cuestionarios
  + JAD
    - Joint Application Design: Consiste en un taller donde los trabajadores del conocimiento y los especialistas en tecnologías de información se reúnen, algunas veces durante varios días, para definir y revisar los requerimientos de negocio para el sistema.
  + Brainstorming

**Para llevarla a cabo es necesario:**

Diagrama

Descripción generada automáticamente

Se hace un análisis TOP-DOWN primero una visión general y se va metiendo más en profundidad, por eso las actividades son cíclicas porque se realizan varias veces.

* Pensamiento analítico: para poder entender y analizar
* Empatía: ponerse en el lugar del otro, no es simpatía, ni antipatía. Tener un lugar de entendimiento sin festejar todo lo que dice.
* Resolución de conflictos: tener la mente fría.
* Capacidad de moderación: separar la paja del trigo.
* Capacidad de conducción (agregado por el profe): no dejar que se vayan por las ramas las personas y obtener lo que se busca.
* Seguridad: sin ser fanfarrón.
* Capacidad de persuasión: lograr que el otro se abra.

**Puntos de dolor:** “donde aprieta el zapato”, es importante identificarlo.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Entrevista abierta: voy sacando temas para ver a donde ir. Son preguntas a desarrollar sin límite de tiempo.

Entrevista cerrada: tener preguntas ya hechas desde antes. Pueden ser por si o por no. Preguntas más detalladas.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Encuesta o cuestionario: consiste en plantear un conjunto de preguntas a individuos representativos, con el fin de tener información cuantitativa o cualitativa.

Los cuestionarios tienen:

Preguntas cerradas: preguntas por si o por no, respuestas cortas, respuestas fáciles de estandarizar, más fácil de analizar.

Preguntas abiertas: usualmente la pregunta lleva que, como, cuando. Respuestas mas largas, toma más tiempo responderlas, más difícil de estandarizar, más difícil de analizar.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Observación: ver como se trabaja para entender la problemática.

No es aplicable a todos los trabajos.

Observación participante: estar participando en la actividad desarrollada.

Observación no participante: No intervenís para nada en el hecho.

Interfaz de usuario gráfica

Descripción generada automáticamente

Brainstorming: tantos participantes de la reunión y generar ideas. Se hace para repensar el negocio. Una desventaja es que se necesita que los participantes estén compenetrados.

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza baja

Desarrollo conjunto de aplicaciones: se basa en organizar reuniones integradas por directivos, desarrolladores y miembros de la organización cliente para la que se va a desarrollar el software.

Se aportan ideas pero es un proceso más normalizado, no hay ideas que no tengan que ver con el proyecto.

Texto

Descripción generada automáticamente

Análisis de protocolos: ver como el experto explica el trabajo a realizar. Sirve para ver como el experto piensa el trabajo.

Interfaz de usuario gráfica

Descripción generada automáticamente

Mapas de concepto: son grafos en que los vértices representan los conceptos y las aristas posibles relaciones con los conceptos. permiten aclarar conceptos relacionados al sistema a desarrollar.

Texto

Descripción generada automáticamente

Maquetas de interacción: su principal objetivo es un feedback por parte de los usuarios acerca del sistema, en las primeras etapas del desarrollo.

No siempre se tiene que usar este método, depende de las circunstancias.

Se utiliza mucho para las interfaces de usuario.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Card sorting: técnica que se usa para categorizar contenidos, se basa en la observación de como los usuarios agrupan y asocian, entre si un numero determinado de tarjetas.

Abierto: agrupa las categorías como quiera

Cerrado: ordenar dentro de categorías ya predefinidas.

**Clase 5:**

## El modelo de dominio

Elemento o grafico que permite determinar distintos elementos conceptuales que intervienen en un espacio que se está analizando, permite modelar una parte de la realidad

Si yo quiero modelar toda la universidad, modelemos un parte: cursado de una materia. Qué materia, que comisión se cursa, quienes cursan esa comisión y quienes la dictan 4 elementos: materia, comisión, docente y alumnos nos permite ver y conocer cuáles son los datos que se requieren, no el proceso, no le interesa, le interesa representar cuáles son esas entidades/elementos conceptuales que intervienen en la situación que se quiere modelar.

Todo esto, está en el libro UML y patrones

## Dominio

Porción de la realidad sobre la que cuál va a operar un sistema, el sistema está influido por esa realidad y busca que cambie.

El objetivo del sistema es modificar al dominio.

Todo lo que está fuera del dominio al sistema no le interesa porque no lo modifica.

Si el sistema no toma datos, el dominio no sirve.

Los sistemas cerrados, es decir: un sistema que no interactúa con nadie, como sistema informático obvio que no sirve.

**DATOS -------- PROCESO ----------- INFORMACIÓN ----------- TOMA DE DECISIONES**

El tema de hoy: **modelo de dominio.**

Es un dibujo, diagrama, representación del dominio de un sistema.

Tenemos que tener herramientas profesionales para poder adquirir información, es decir: para entender lo que un cliente necesita, aunque no sepamos nada de su negocio (en dominios que no conoce). Entonces, la idea es obtener información sobre el dominio.

Lo primero que analizo no es el sistema que tengo que armarle el cliente, sino que tengo que conocer la pequeña porción de realidad en la que va a funcionar el sistema.

Ejercicio 1 (Air Patagonia) está en formato de texto (lenguaje coloquial, natural, no técnico, cualquier persona que haya viajado lo puede comprender), es muy ambiguo.

Usar una plantilla de texto / organización del texto es bueno porque se entiende más, pero es mejor un modelo, un dibujo **(un tipo especifico: Modelo de Dominio - Craig Larman). UML y patrones - Capítulo 9.**

Pregunta el profe: ¿El dominio de un sistema suele ser todo el universo? No, es solo una porción.

¿Si el dominio fuera todo el universo, el sistema que película sería? La matrix.

El modelo de dominio es como un ecosistema. Tiene conceptos (una COSA que hay en ese dominio), esos conceptos tienen atributos y además se relacionan entre sí.

De las relaciones es importante conocer la cardinalidad.

Larman (autor) habla de elementos que se representan a través de clases conceptuales.

Se evita la palabra objeto, entonces se usa concepto o clase conceptual.

Una técnica para realizar un modelo la propone una autora (Rebecca Wirfs-Brock) que usa la estructura sintáctica del texto para extraer los conceptos (en este caso nos viene bien porque el ejercicio 1 está dado en un texto). Busca sustantivos primero (suelen ser conceptos), los verbos van a estar relacionados con los vínculos entre conceptos (atributos), los artículos nos van a dar una pauta sobre las cardinalidades (esto de los artículos lo agrega el profe, no la autora).

No todo sustantivo es automáticamente un concepto o atributo.

No tenemos siempre certeza del 100%. Para hacer un MDD perfecto hay que tener mucha experiencia en hacer modelos

Un chico consulta si el sustantivo "despachante" es importante y el profesor contesta: relativamente porque es lo que llamamos un actor.

La mayoría de los sistemas informáticos están vinculados a transacciones (algo que pasa/eventos), si no pasa nada, no se va a realizar un sistema informático si no pasa nada, no habría dominio.

En Air Patagonia el evento/transacción por excelencia es el despacho.

Menciona el utilizar la regla #1 del cartógrafo (el que hace mapas :P)

1. cuando usted haga un mapa (modelo de dominio: mapa) utilice los nombres existentes en el territorio. (en nuestro caso, llamemos al despacho por su nombre, no le pongamos otro)

Conceptos físicos/tangibles: vuelo, avión, aeropuerto, despachante, pasajero, valija.

Concepto abstracto: despacho

Transacción: intercambio.

No todo evento es una transacción, pero toda transacción es un evento.

Transacción: le doy mi valija, me dan un ticket.

Siempre es aconsejable arrancar por el concepto del modelo.

Creamos una clase "despacho"

Botón derecho sobre "despacho"--"specification"--"add"

agregamos un atributo de nombre "fecha"---"apply"

agregamos también otro atributo de nombre "hora"

entonces nuestro concepto despacho tiene de atributos: fecha y hora

Agregamos ahora "código"

---------------INTERVALO

Agrega: número de vuelo como atributo y explica que o tengo la clase conceptual vuelo y tengo el número de vuelo allí o tengo el número de vuelo en la clase conceptual despacho. No está bien que un mismo atributo esté en varios lados.

La consigna dice "Air Patagonia opera en el aeropuerto local" entonces no es relevante poner el origen del vuelo en ningún lado, siempre va a ser el mismo.

Fecha y hora si son muy importantes porque el servicio de despacho solo opera en las horas posteriores.

Estamos metiendo atributos que son dependientes de otro atributo, como fecha y hora de salida, que son dependientes de vuelo, por eso nos damos cuenta de que tendríamos que tener el concepto vuelo

Entonces de momento tenemos:

Despacho: código, fecha, hora,

Vuelo: número de vuelo, fecha de salida, hora salida

También se podría pensar un concepto de salida y un concepto de llegada, es todo opinable, pero lo que no se puede dudar es que el concepto vuelo tiene que existir, el concepto despacho también.

Ahora a través de una flecha "association" vinculamos ambos conceptos, y en medio de la flecha colocamos que en un vuelo transporta un despacho, o que un despacho viaja en un vuelo. Aún no tenemos claro el sentido de la flecha, ni vamos a hablar de que en un vuelo puede haber muchos despachos. De momento la relación es 1 a 1.

Luego hacemos clic en una flechita que apunta al despacho y aparece una etiqueta, ahora nos preguntamos como máximo ¿cuántos despachos puede tener un vuelo? rta: N (muchos)

¿Cómo mínimo? rta: puede tener 0 ya que si viajas con carry on no despachas nada.

Entonces, la etiqueta en Multiplicity que debemos colocar es: 0..\* es decir que un vuelo puede tener 0 como mínimo y \* (muchos) despachos como máximo.

Luego cuando ponemos ok, aparece una etiqueta con un número default que se puede cambiar.

Luego del otro lado (el lado del vuelo), y nos preguntamos ¿Cada despacho, a cuantos vuelos puede pertenecer máximo? rta: 1 vuelo

¿Cómo mínimo? rta: 1 vuelo

Entonces en Multiplicity aplicamos la etiqueta 1 simplemente.

Multiplicity = cardinalidad.

## Cardinalidad de 1 a muchos

Entonces nos queda que la cardinalidad es 1 (vuelo) -------------- muchos (despachos)

"Un vuelo transporta varios despachos"

"Un despacho puede ir en un solo vuelo"

Es el tipo más común de cardinalidad.

Cuando hay una **cardinalidad de 1 a muchos**, el sentido va del que tiene menos al que tiene más.

En este caso vuelo ---(transporta)--->despacho

Abrimos entonces "Especificaciones" haciendo clic derecho en la flecha y en lado de vuelo en "Navigable" ponemos "<Unespecified>" y en el lado de despacho ponemos "True".

Diagrama

Descripción generada automáticamente

Hay dos tipos de cardinalidades más:

## Cardinalidad muchos a muchos

Ejemplo: veterinaria que solo atiende a perritos

Concepto: perro

Atributos de perro: nombre, raza, fecha de nacimiento, código de perro

Concepto: dueño

Atributos de dueño: nombres, apellidos, dirección, teléfono

Un dueño puede tener como máximo muchos perros, como mínimo 1.

Un perro puede obedecer a varios dueños, como máximo a muchos, como mínimo a 0 porque puede ser un perrito de la calle.

Entonces en la especificación

1..\* en perros

0..\* en dueños

Normalmente suele ser problemática esta cardinalidad.

Es altamente probable que haya otro concepto, a veces hay que ponerlo, a veces no es necesario.

Muchas veces la cardinalidad de muchos a muchos se descompone en dos cardinalidades de 1 a muchos.

En este caso para saber para qué lado va la flecha, nos preguntamos en nuestro dominio ¿Cuál es más importante? ¿Dueño o perro? En nuestro caso decidimos que el perro, entonces colocamos la flecha apuntando hacia dueño.

perro ------(obedece)------- dueño

## Cardinalidad 1 a 1

A nuestra veterinaria le agregamos el concepto Tarjeta de crédito

atributos: numero, banco, vencimiento

Un dueño tiene una tarjeta

Cada TC corresponde como máximo a 1 dueño y como mínimo 1, no hay TC que no tengan dueño.

¿Un dueño cuantas tarjetas va a tener? máximo 1 y mínimo 0 (en un dominio de un sistema muy muy sencillo)

Este modelo dice si hay una tarjeta, tiene dueño.

En este modelo:

¿Un dueño está obligado a tener tarjeta? no

¿Un dueño está obligado a tener perro? si

¿Un perro está obligado a tener dueño? no

¿Un perro tiene relación directa con una tarjeta de crédito? hay relación, pero no es directa.

En el caso de la relación dueño - tarjeta tenemos problemas para determinar el sentido de la flecha, pero en esta cardinalidad pensamos en la opcionalidad

¿El dueño obligatoriamente tiene TC? no

¿La tarjeta obligatoriamente tiene dueño? si

En este caso la fecha va desde dueño ---(tiene)--->TC

Diagrama

Descripción generada automáticamente

# Modelo de Dominio de Larman

**(**[**link al pdf**](https://miel.unlam.edu.ar/data7/data2/contenido/3630/Modelo-de-Dominio-de-Larman-(AS---2010).pdf)**)**

El **modelo de dominio** captura los tipos de **clases** más importantes en el **contexto** del sistema. Las clases del dominio representan “**cosas**” que existen o los eventos que suceden en el entorno en el que trabaja el sistema.

Las clases del dominio aparecen en formas típicas:

* Objetos del negocio
* Objetos del mundo real y conceptos
* Sucesos que ocurrirán o han ocurrido

El MDD se describe mediante diagramas de UML (diagrama de clases) ; es el artefacto más importante que se crea durante el análisis orientado a objetos. Es una representación de las clases conceptuales del mundo real.

Muestra:

* Clases conceptuales.
* Asociaciones entre las clases conceptuales.
* Atributos de las clases conceptuales.

## Clases conceptuales

Una clase conceptual es una idea, cosa u objeto. Más formalmente, una clase conceptual podría considerarse en términos de sus símbolos, intención, extensión. Símbolo: Palabras o imágenes que representan una clase conceptual. Intención: La definición de una clase conceptual. Extensión: El conjunto de ejemplos a los que se aplica la clase conceptual.

**No** hay que confundir **una clase conceptual** con una **clase software**, ya que la primera solo tiene atributos, pertenece al dominio del problema y a la etapa de análisis. La segunda tiene atributos y métodos, pertenece al dominio de la solución y a la etapa del diseño.

## Estrategias para identificar clases conceptuales

**MÉTODO 1**- Distinguir de un texto determinado los sustantivos y los verbos. Los sustantivos son posibles clases candidatas o atributos de las mismas. Los verbos son posibles responsabilidades. Las clases candidatas pasaran a ser definitivas cuando de acuerdo con el contexto le asignemos los atributos.

**MÉTODO 2-** Tabla de categorías de objetos de Larman. Todo objeto va a formar parte de una o más categorías.

Tabla

Descripción generada automáticamente

Tabla

Descripción generada automáticamente

## Nombrar y modelar cosas: El cartógrafo

La estrategia del cartógrafo se aplica tanto a los mapas como al modelo del dominio. Se hace un modelo del dominio con el espíritu del modo de trabajo de los cartógrafos:

* Utilice los nombres existentes en el territorio. *Para un modelo del dominio, significa que utilice el vocabulario del dominio al asignar nombres a las clases conceptuales y los atributos.*
* Excluya las características irrelevantes. *Un modelo de dominio podría excluir clases conceptuales del dominio del problema que no son pertinentes para los requerimientos.*
* No añada cosas que no están ahí. *El modelo del dominio debería excluir cosas que no se encuentran en el dominio del problema que se está estudiando.*

## Añadir atributos a las clases conceptuales

Un atributo es una característica o propiedad de una clase. Los atributos que se le asignan a una clase tienen que ver con el propósito para el cual se realiza.

Diagrama, Tabla

Descripción generada automáticamente

## Añadir asociaciones

Una asociación es una relación semántica entre dos o más clases conceptuales que implica conexiones entre sus instancias.

Una asociación entre una clase conceptual A y una clase conceptual B se indica como una línea entre las mismas, con un nombre, un sentido de lectura de la asociación y una cardinalidad.

El sentido de lectura de la asociación es de la clase conceptual A a la clase conceptual B o viceversa.

La cardinalidad si el sentido de lectura es de A hacia B significa que una instancia de la clase conceptual A está relacionada con una o muchas instancias de la clase conceptual B. Este recorrido es puramente abstracto; no se trata de una sentencia sobre conexiones entre entidades software.

Diagrama

Descripción generada automáticamente

## Lista de asociaciones comunes de Larman

**Tabla

Descripción generada automáticamente**

**Tabla

Descripción generada automáticamente**

# (CAP 4) SOMMERVILLE - INGENIERÍA DE SOFTWARE

**(**[**link al pdf**](https://miel.unlam.edu.ar/data7/data2/contenido/3630/Ingenieria-de-Software---Somerville,-Ian---9---Edic.pdf)**)**

Los r**equerimientos para un sistema** son descripciones de lo que el sistema debe hacer: el servicio que ofrece y las restricciones en su operación. Tales requerimientos **reflejan las necesidades de los clientes** por un sistema que atienda cierto propósito.

Al proceso de descubrir, analizar, documentar y verificar estos servicios y restricciones se le llama **ingeniería de requerimientos** (IR).

Los requerimientos del usuario y los requerimientos del sistema se definen del siguiente modo:

1. Los **requerimientos del usuario** son enunciados, en un lenguaje natural junto con diagramas, acerca de qué servicios esperan los usuarios del sistema, y de las restricciones con las cuales éste debe operar.
2. Los **requerimientos del sistema** son descripciones más detalladas de las funciones, los servicios y las restricciones operacionales del sistema de software. El documento de requerimientos del sistema (llamado en ocasiones especificación funcional) tiene que definir con exactitud lo que se implementará. Puede formar parte del contrato entre el comprador del sistema y los desarrolladores del software.

Es necesario escribir los requerimientos con diferentes niveles de detalle, ya que varios lectores los usarán de distintas formas.

Es necesario escribir los requerimientos con diferentes niveles de detalle, ya que varios lectores los usarán de distintas formas.

## 4.1 Requerimientos funcionales y no funcionales

A menudo, los requerimientos del sistema de software se clasifican como requerimientos funcionales o requerimientos no funcionales:

1. **Requerimientos funcionales** Son enunciados acerca de servicios que el sistema debe proveer, de cómo debería reaccionar el sistema a entradas particulares y de cómo debería comportarse el sistema en situaciones específicas. En algunos casos, los requerimientos funcionales también explican lo que no debe hacer el sistema.
2. **Requerimientos no funcionales** Son limitaciones sobre servicios o funciones que ofrece el sistema. Incluyen restricciones tanto de temporización y del proceso de desarrollo, como impuestas por los estándares. Los requerimientos no funcionales se suelen aplicar al sistema como un todo, más que a características o a servicios individuales del sistema.

La distinción entre los diferentes tipos de requerimientos no es tan clara como sugieren estas definiciones sencillas.

Los requerimientos no son independientes y que un requerimiento genera o restringe normalmente otros requerimientos.

### 4.1.1 Requerimientos Funcionales

Los requerimientos funcionales para un sistema refieren lo que el sistema debe hacer.

Al expresarse como requerimientos del usuario, los requerimientos funcionales se describen por lo general de forma abstracta que entiendan los usuarios del sistema. Sin embargo, requerimientos funcionales más específicos del sistema detallan las funciones del sistema, sus entradas y salidas, sus excepciones, etcétera.

Varían desde requerimientos generales que cubren lo que tiene que hacer el sistema, hasta requerimientos muy específicos que reflejan maneras locales de trabajar

|  |
| --- |
| Requerimientos de dominio:  Los requerimientos de dominio se derivan del dominio de aplicación del sistema, más que a partir de las necesidades específicas de los usuarios del sistema. Pueden ser requerimientos funcionales nuevos por derecho propio, restricciones a los requerimientos funcionales existentes o formas en que deben realizarse cálculos particulares. El problema con los requerimientos de dominio es que los ingenieros de software no pueden entender las características del dominio en que opera el sistema. Por lo común, no pueden indicar si un requerimiento de dominio se perdió o entró en conflicto con otros requerimientos. |

La inexactitud en la especificación de requerimientos causa muchos problemas en la ingeniería de software. Es natural que un desarrollador de sistemas interprete un requerimiento ambiguo de forma que simplifique su implementación. Sin embargo, con frecuencia, esto no es lo que desea el cliente.

En principio, la especificación de los requerimientos funcionales de un sistema debe ser **completa y consistente.** Totalidad (completa -.-) significa que deben definirse todos los servicios requeridos por el usuario. Consistencia quiere decir que los requerimientos tienen que evitar definiciones contradictorias.

### 4.1.2 Requerimientos NO Funcionales

Son requerimientos que no se relacionan directamente con los servicios específicos que el sistema entrega a sus usuarios. Pueden relacionarse con propiedades emergentes del sistema, como fiabilidad, tiempo de respuesta y uso de almacenamiento. De forma alternativa, pueden definir restricciones sobre la implementación del sistema.

a menudo son más significativos que los requerimientos funcionales individuales, el fracaso para cubrir los requerimientos no funcionales haría que todo el sistema fuera inútil.

por lo general es más difícil relacionar componentes con requerimientos no funcionales. La implementación de dichos requerimientos puede propagarse a lo largo del sistema. Para esto existen dos razones:

1. Los requerimientos no funcionales afectan más la arquitectura global de un sistema que los componentes individuales. Por ejemplo, para garantizar que se cumplan los requerimientos de rendimiento, quizá se deba organizar el sistema para minimizar las comunicaciones entre componentes.
2. Un requerimiento no funcional individual, como un requerimiento de seguridad, podría generar algunos requerimientos funcionales relacionados que definan nuevos servicios del sistema que se requieran. Además, también podría generar requerimientos que restrinjan los requerimientos ya existentes.

#### Tipo de requerimientos no funcionales

Diagrama

Descripción generada automáticamente

1. Requerimientos del producto Estos requerimientos especifican o restringen el comportamiento del software. Los ejemplos incluyen requerimientos de rendimiento sobre qué tan rápido se debe ejecutar el sistema y cuánta memoria requiere, requerimientos de fiabilidad que establecen la tasa aceptable de fallas, requerimientos de seguridad y requerimientos de usabilidad.
2. Requerimientos de la organización Son requerimientos de sistemas amplios, derivados de políticas y procedimientos en la organización del cliente y del desarrollador. Los ejemplos incluyen requerimientos del proceso operacional que definen cómo se usará el sistema, requerimientos del proceso de desarrollo que especifican el lenguaje de programación, estándares del entorno o el proceso de desarrollo a utilizar, y requerimientos ambientales que definen el entorno de operación del sistema.
3. Requerimientos externos Este término cubre todos los requerimientos derivados de factores externos al sistema y su proceso de desarrollo. En ellos se incluyen requerimientos regulatorios que establecen lo que debe hacer el sistema para ser aprobado en su uso por un regulador, como sería un banco central; requerimientos legislativos que tienen que seguirse para garantizar que el sistema opere conforme a la ley, y requerimientos éticos que garanticen que el sistema será aceptable para sus usuarios y el público en general.

Un problema común con requerimientos no funcionales es que los usuarios o clientes con frecuencia proponen estos requerimientos como metas generales, como facilidad de uso, capacidad de que el sistema se recupere de fallas, o rapidez de respuesta al usuario.

Siempre que sea posible, se deberán escribir de manera cuantitativa los requerimientos no funcionales, de manera que puedan ponerse objetivamente a prueba.

#### Métricas para especificar requerimientos no funcionales

Tabla

Descripción generada automáticamente con confianza media

En la práctica, los usuarios de un sistema suelen encontrar difícil traducir sus metas en requerimientos mensurables. Para algunas metas no hay métricas para usarse. En otros casos los clientes no logran relacionar sus necesidades con dichas especificaciones. Más aún, el costo por verificar objetivamente los requerimientos no funcionales mensurables suele ser muy elevado, y los clientes que pagan por el sistema quizá piensen que dichos costos no están justificados

Los requerimientos no funcionales entran a menudo en conflicto e interactúan con otros requerimientos funcionales o no funcionales.

En la práctica, en el documento de requerimientos, resulta difícil separar los requerimientos funcionales.

|  |
| --- |
| Estándares del documento de requerimientos:  Algunas organizaciones grandes, como el Departamento de Defensa estadounidense y el Institute of Electrical and Electronic Engineers (IEEE), definieron estándares para los documentos de requerimientos. Comúnmente son muy genéricos, pero útiles como base para desarrollar estándares organizativos más detallados. |

## 4.2 El documento de requerimientos de software

Llamado algunas veces especificación de requerimientos de software o SRS) es un comunicado oficial de lo que deben implementar los desarrolladores del sistema. Incluye tanto los requerimientos del usuario para un sistema, como una especificación detallada de los requerimientos del sistema. En ocasiones, los requerimientos del usuario y del sistema se integran en una sola descripción. En otros casos, los requerimientos del usuario se definen en una introducción a la especificación de requerimientos del sistema. Si hay un gran número de requerimientos, los requerimientos del sistema detallados podrían presentarse en un documento aparte.

Los métodos de desarrollo ágiles argumentan que los requerimientos cambian tan rápidamente que un documento de requerimientos se vuelve obsoleto tan pronto como se escribe, Sin embargo, aún resulta útil escribir un breve documento de apoyo que defina los requerimientos de la empresa y los requerimientos de confiabilidad para el sistema;

El documento de requerimientos tiene un conjunto variado de usuarios, eso significa que el documento de requerimientos debe ser un compromiso entre la comunicación de los requerimientos a los clientes, la definición de los requerimientos con detalle preciso para desarrolladores y examinadores, y la inclusión de información sobre la posible evolución del sistema.

El nivel de detalle que se incluya en un documento de requerimientos depende del tipo de sistema a diseñar y el proceso de desarrollo utilizado.

#### Usuarios de un documento de requerimientos

Diagrama

Descripción generada automáticamente

Este estándar es genérico y se adapta a usos específicos. En este caso, el estándar se extendió para incluir información de la evolución prevista del sistema.

#### Estándar genérico para un documento de requerimientos (posible)

En este caso, el estándar se extendió para incluir información de la evolución prevista del sistema.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

El enfoque estará en especificar los requerimientos del usuario y los requerimientos no funcionales de alto nivel del sistema.

Sin embargo, cuando el software sea parte de un proyecto de sistema grande que incluya la interacción de sistemas de hardware y software, será necesario por lo general definir los requerimientos a un nivel detallado.

Para documentos extensos, es muy importante incluir una tabla de contenido global y un índice del documento.

## 4.3 Especificación de requerimientos

La **especificación de requerimientos** es el proceso de escribir, en un documento de requerimientos, los requerimientos del usuario y del sistema. De manera ideal, los requerimientos del usuario y del sistema deben ser claros, sin ambigüedades, fáciles de entender, completos y consistentes. Es difícil de lograr.

Los requerimientos del usuario para un sistema deben describir los requerimientos funcionales y no funcionales, de forma que sean comprensibles para los usuarios del sistema que no cuentan con un conocimiento técnico detallado. De manera ideal, deberían especificar sólo el comportamiento externo del sistema. El documento de requerimientos no debe incluir detalles de la arquitectura o el diseño del sistema. En consecuencia, si usted escribe los requerimientos del usuario, no tiene que usar jerga de software, anotaciones estructuradas o formales. Debe escribir los requerimientos del usuario en lenguaje natural, con tablas y formas sencillas, así como diagramas intuitivos.

Los **requerimientos del sistema** son versiones extendidas de los requerimientos del usuario que los ingenieros de software usan como punto de partida para el diseño del sistema. Añaden detalles y explican cómo el sistema debe brindar los requerimientos del usuario. Se pueden usar como parte del contrato para la implementación del sistema y, por lo tanto, deben ser una especificación completa y detallada de todo el sistema.

Al nivel de detalle requerido para especificar por completo un sistema de software complejo, es prácticamente imposible excluir toda la información de diseño. Para ello existen varias razones

1. Tal vez se tenga que diseñar una arquitectura inicial del sistema para ayudar a estructura la especificación de requerimientos. Los requerimientos del sistema se organizan de acuerdo con los diferentes subsistemas que constituyen el sistema.
2. En la mayoría de los casos, los sistemas deben interoperar con los sistemas existentes, lo cual restringe el diseño e impone requerimientos sobre el nuevo sistema
3. Quizá sea necesario el uso de una arquitectura específica para cubrir los requerimientos no funcionales.

#### Formas de escribir una especificación de requerimientos del sistema

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Los requerimientos del usuario se escriben casi siempre en lenguaje natural, complementado con diagramas y tablas adecuados en el documento de requerimientos. Los requerimientos del sistema se escriben también en lenguaje natural, pero de igual modo se utilizan otras notaciones basadas en formas, modelos gráficos del sistema o modelos matemáticos del sistema.

Los modelos gráficos son más útiles cuando es necesario mostrar cómo cambia un estado o al describir una secuencia de acciones.

### 4.3.1 Especificación en lenguaje natural

Es expresivo, intuitivo y universal.

Para minimizar la interpretación errónea al escribir los requerimientos en lenguaje natural, se recomienda seguir algunos lineamientos sencillos

1. Elabore un formato estándar y asegúrese de que todas las definiciones de requerimientos se adhieran a dicho formato.
2. Utilice el lenguaje de manera clara para distinguir entre requerimientos obligatorios y deseables. Los primeros son requerimientos que el sistema debe soportar y, por lo general, se escriben en futuro “debe ser”. En tanto que los requerimientos deseables no son necesarios y se escriben en tiempo pospretérito o como condicional “debería ser”.
3. Use texto resaltado (negrilla, cursiva o color) para seleccionar partes clave del requerimiento.
4. No deduzca que los lectores entienden el lenguaje técnico de la ingeniería de software
5. Siempre que sea posible, asocie una razón con cada requerimiento de usuario. Debe explicar por qué.

### 4.3.1 Especificación estructuradas

Las anotaciones en lenguaje estructurado emplean plantillas para especificar requerimientos del sistema.

Los Robertson (Robertson y Robertson, 1999), en su libro del método de ingeniería de requerimientos VOLERE, recomiendan que se escriban los requerimientos del usuario inicialmente en tarjetas, un requerimiento por tarjeta.

Cuando use una forma estándar para especificar requerimientos funcionales, debe incluir la siguiente información:

1. Una descripción de la función o entidad a especificar.
2. Una descripción de sus entradas y sus procedencias.
3. Una descripción de sus salidas y a dónde se dirigen.
4. Información sobre los datos requeridos para el cálculo u otras entidades en el sistema que se utilizan (la parte “requiere”).
5. Una descripción de la acción que se va a tomar.
6. Si se usa un enfoque funcional, una precondición establece lo que debe ser verdadero antes de llamar a la función, y una postcondición especifica lo que es verdadero después de llamar a la función.
7. Una descripción de los efectos colaterales (si acaso hay alguno) de la operación.

Sin embargo, en ocasiones todavía es difícil escribir requerimientos sin ambigüedades.

## 4.4 Proceso de Ingeniería de Requerimientos

Los procesos de ingeniería de requerimientos incluyen cuatro actividades de alto nivel. Éstas se enfocan en valorar si el sistema es útil para la empresa (estudio de factibilidad), descubrir requerimientos (adquisición y análisis), convertir dichos requerimientos en alguna forma estándar (especificación) y comprobar que los requerimientos definan realmente el sistema que quiere el cliente (validación).

#### Vista en espiral del proceso de ingeniería de requerimientos

Gráfico, Gráfico radial

Descripción generada automáticamente

Las actividades están organizadas como un proceso iterativo alrededor de una espiral, y la salida es un documento de requerimientos del sistema. La cantidad de tiempo y esfuerzo dedicados a cada actividad en cada iteración depende de la etapa del proceso global y el tipo de sistema que está siendo desarrollado. En el inicio del proceso, se empleará más esfuerzo para comprender los requerimientos empresariales de alto nivel y los no funcionales, así como los requerimientos del usuario para el sistema. Más adelante en el proceso, en los anillos exteriores de la espiral, se dedicará más esfuerzo a la adquisición y comprensión de los requerimientos detallados del sistema.

Algunas personas consideran la ingeniería de requerimientos como el proceso de aplicar un método de análisis estructurado, tal como el análisis orientado a objetos (Larman, 2002). Esto implica analizar el sistema y desarrollar un conjunto de modelos gráficos del sistema.

|  |
| --- |
| Estudio de factibilidad:  Un estudio de factibilidad es un breve estudio enfocado que debe realizarse con oportunidad en el proceso de IR. Debe responder tres preguntas clave: a) ¿El sistema contribuye con los objetivos globales de la organización? b) ¿El sistema puede implementarse dentro de la fecha y el presupuesto usando la tecnología actual? c) ¿El sistema puede integrarse con otros sistemas que se utilicen? Si la respuesta a cualquiera de estas preguntas es negativa, probablemente no sea conveniente continuar con el proyecto |

## 4.5 Adquisición y Análisis de Requerimientos

Después de un estudio de factibilidad inicial, la siguiente etapa del proceso de ingeniería de requerimientos es la adquisición y el análisis de requerimientos. En esta actividad, los ingenieros de software trabajan con clientes y usuarios finales del sistema para descubrir el dominio de aplicación, qué servicios debe proporcionar el sistema, el desempeño requerido de éste, las restricciones de hardware, etcétera.

#### El proceso de adquisición y análisis de requerimientos

Diagrama

Descripción generada automáticamente

En una organización, la adquisición y el análisis de requerimientos pueden involucrar a diversas clases de personas. Un participante en el sistema es quien debe tener alguna influencia directa o indirecta sobre los requerimientos del mismo.

Incluyen a usuarios finales, ingenieros que desarrollan o mantienen otros sistemas relacionados, administradores de negocios, expertos de dominio y representantes de asociaciones sindicales, etc.

Las actividades del proceso son:

1. **Descubrimiento de requerimientos** Éste es el proceso de interactuar con los participantes del sistema para descubrir sus requerimientos. También los requerimientos de dominio de los participantes y la documentación se descubren durante esta actividad.
2. **Clasificación y organización de requerimientos** Esta actividad toma la compilación no estructurada de requerimientos, agrupa requerimientos relacionados y los organiza en grupos coherentes
3. **Priorización y negociación de requerimientos** Inevitablemente, cuando intervienen diversos participantes, los requerimientos entrarán en conflicto. Esta actividad se preocupa por priorizar los requerimientos, así como por encontrar y resolver conflictos de requerimientos mediante la negociación.
4. **Especificación de requerimientos** Los requerimientos se documentan e ingresan en la siguiente ronda de la espiral. Pueden producirse documentos de requerimientos formales o informales.

La adquisición y el análisis de requerimientos es un proceso iterativo con retroalimentación continua de cada actividad a otras actividades.

La adquisición y la comprensión de los requerimientos por parte de los participantes del sistema es un proceso difícil por diferentes razones:

* Los participantes con frecuencia no saben lo que quieren de un sistema de cómputos, solo en líneas generales, les es difícil articular, pueden hacer peticiones inalcanzables.
* Los participantes de un sistema expresan naturalmente los requerimientos con términos y conocimientos implícitos, haciendo difícil la comprensión.
* Diferentes participantes tienen distintos requerimientos
* Los factores políticos de la empresa influyen en los requerimientos del sistema (por ejemplo, pedir un requerimiento que suba de status a alguien)
* El ambiente económico y empresarial donde se hace el análisis es dinámico.

En la etapa de especificación de requerimientos, los requerimientos adquiridos hasta el momento se documentan de tal forma que puedan usarse para ayudar al hallazgo de requerimientos. En esta etapa, podría generarse una primera versión del documento de requerimientos del sistema, con secciones faltantes y requerimientos incompletos.

### 4.5.1 Descubrimiento de requerimientos

Las fuentes de información durante la fase de descubrimiento de requerimientos incluyen documentación, participantes del sistema y especificaciones de sistemas similares. La interacción con los participantes es a través de entrevistas y observaciones, y pueden usarse escenarios y prototipos para ayudar a los participantes a entender cómo será el sistema.

Todas estas diferentes fuentes de requerimientos (participantes, dominio, sistemas) se representan como puntos de vista del sistema, y cada visión muestra un subconjunto de los requerimientos para el sistema.

### 4.5.2 Entrevistas

Las entrevistas son de dos tipos:

1. Entrevistas cerradas, donde los participantes responden a un conjunto de preguntas preestablecidas.
2. Entrevistas abiertas, en las cuales no hay agenda predefinida. El equipo de ingeniería de requerimientos explora un rango de conflictos con los participantes del sistema y, como resultado, desarrolla una mejor comprensión de sus necesidades.

En la práctica, las entrevistas con los participantes son por lo general una combinación de ambas.

Las entrevistas son valiosas para lograr una comprensión global sobre qué hacen los participantes, cómo pueden interactuar con el nuevo sistema y las dificultades que enfrentan con los sistemas actuales.

Por dos razones resulta difícil asimilar el conocimiento de dominio a través de entrevistas:

* Todos los especialistas en la aplicación usan terminología y jerga que son específicos de un dominio.
* Cierto conocimiento del dominio es tan familiar a los participantes que encuentran difícil de explicarlo, o bien, creen que es tan fundamental que no vale la pena mencionarlo.

Las entrevistas tampoco son una técnica efectiva para adquirir conocimiento sobre los requerimientos y las restricciones de la organización, porque existen relaciones sutiles de poder entre los diferentes miembros en la organización.

Los entrevistadores efectivos poseen dos características:

* Tienen mentalidad abierta, evitan ideas preconcebidas sobre los requerimientos y escuchan a los participantes.
* Instan al entrevistado con una pregunta de trampolín para continuar la plática, dar una propuesta de requerimientos o trabajar juntos en un sistema de prototipo.

La información de las entrevistas se complementa con otra información del sistema de documentación que describe los procesos empresariales o los sistemas existentes, las observaciones del usuario, etcétera.

### 4.5.3 Escenarios

Por lo general, las personas encuentran más sencillo vincularse con ejemplos reales que con descripciones abstractas. Pueden comprender y criticar un escenario sobre cómo interactuar con un sistema de software.

Los escenarios son particularmente útiles para detallar un bosquejo de descripción de requerimientos.

Un escenario comienza con un bosquejo de la interacción. Durante el proceso de adquisición, se suman detalles a éste para crear una representación completa de dicha interacción. En su forma más general, un escenario puede incluir:

1. Una descripción de qué esperan el sistema y los usuarios cuando inicia el escenario.
2. Una descripción en el escenario del flujo normal de los eventos.
3. Una descripción de qué puede salir mal y cómo se manejaría.
4. Información de otras actividades que estén en marcha al mismo tiempo.
5. Una descripción del estado del sistema cuando termina el escenario.

### 4.5.4 Casos de Uso

Los casos de uso son una técnica de descubrimiento de requerimientos que se introdujo por primera vez en el método Objectory (Jacobson et al., 1993).

En su forma más sencilla, un caso de uso identifica a los actores implicados en una interacción, y nombra el tipo de interacción. Entonces, esto se complementa con información adicional que describe la interacción con el sistema. La información adicional puede ser una descripción textual, o bien, uno o más modelos gráficos como una secuencia UML o un gráfico de estado. Los casos de uso se documentan con el empleo de un diagrama de caso de uso de alto nivel.

No hay distinción tajante y rápida entre escenarios y casos de uso. Algunas personas consideran que cada caso de uso es un solo escenario; otras, como sugieren Stevens y Pooley (2006), encapsulan un conjunto de escenarios en un solo caso de uso. Cada escenario es un solo hilo a través del caso de uso.

Los casos de uso identifican las interacciones individuales entre el sistema y sus usuarios u otros sistemas.

### 4.5.5 Etnografía

Los sistemas de software no existen aislados. Se usan en un contexto social y organizacional, y dicho escenario podría derivar o restringir los requerimientos del sistema de software.

La etnografía es una técnica de observación que se usa para entender los procesos operacionales y ayudar a derivar requerimientos de apoyo para dichos procesos. Un analista se adentra en el ambiente laboral donde se usará el sistema. Observa el trabajo diario y toma notas acerca de las tareas existentes en que intervienen los participantes.

Las personas con frecuencia encuentran muy difícil articular los detalles de su trabajo, porque es una segunda forma de vida para ellas.

Los factores sociales y organizacionales que afectan el trabajo, que no son evidentes para los individuos, sólo se vuelven claros cuando los percibe un observador sin prejuicios.

#### Etnografía y creación de prototipos para análisis de requerimientos

Diagrama

Descripción generada automáticamente

La etnografía es muy efectiva para descubrir dos tipos de requerimientos:

1. Requerimientos que se derivan de la forma en que realmente trabaja la gente, en vez de la forma en la cual las definiciones del proceso indican que debería trabajar.
2. Requerimientos que se derivan de la cooperación y el conocimiento de las actividades de otras personas.

La etnografía puede combinarse con la creación de prototipos. La etnografía informa del desarrollo del prototipo, de modo que se requieren menos ciclos de refinamiento del prototipo. Más aún, la creación de prototipos se enfoca en la etnografía al identificar problemas y preguntas que entonces pueden discutirse con el etnógrafo.

La etnografía no es un enfoque completo para la adquisición por sí misma, y debe usarse para complementar otros enfoques, como el análisis de casos de uso.

|  |
| --- |
| Revisiones de requerimientos:  Una revisión de requerimientos es un proceso donde un grupo de personas del cliente del sistema y el desarrollador del sistema leen con detalle el documento de requerimientos y buscan errores, anomalías e inconsistencias. Una vez detectados y registrados, recae en el cliente y el desarrollador la labor de negociar cómo resolver los problemas identificados |

## 4.6 Validación de Requerimientos

Es el proceso de verificar que los requerimientos definan realmente el sistema que en verdad quiere el cliente.

En general, el costo por corregir un problema de requerimientos al hacer un cambio en el sistema es mucho mayor que reparar los errores de diseño o codificación. La razón es que un cambio a los requerimientos significa generalmente que también deben cambiar el diseño y la implementación del sistema. Más aún, el sistema debe entonces ponerse a prueba de nuevo. Durante el proceso de validación de requerimientos, tienen que realizarse diferentes tipos de comprobaciones.

1. **Comprobaciones de validez** Un usuario quizá crea que necesita un sistema para realizar ciertas funciones. Sin embargo, con mayor consideración y análisis se logra identificar las funciones adicionales o diferentes que se requieran.

2. **Comprobaciones de consistencia** Los requerimientos en el documento no deben estar en conflicto.

3. **Comprobaciones de totalidad** El documento de requerimientos debe incluir requerimientos que definan todas las funciones y las restricciones pretendidas por el usuario del sistema.

4. **Comprobaciones de realismo** Al usar el conocimiento de la tecnología existente, los requerimientos deben comprobarse para garantizar que en realidad pueden implementarse. Dichas comprobaciones también tienen que considerar el presupuesto y la fecha para el desarrollo del sistema.

5. **Verificabilidad** Para reducir el potencial de disputas entre cliente y contratista, los requerimientos del sistema deben escribirse siempre de manera que sean verificables. Esto significa que usted debe ser capaz de escribir un conjunto de pruebas que demuestren que el sistema entregado cumpla cada requerimiento especificado.

#### Evolución de requerimientos

Diagrama

Descripción generada automáticamente

Hay algunas técnicas de validación de requerimientos que se usan individualmente o en conjunto con otras:

1. Revisiones de requerimientos Los requerimientos se analizan sistemáticamente usando un equipo de revisores que verifican errores e inconsistencias.
2. Creación de prototipos En esta aproximación a la validación, se muestra un modelo ejecutable del sistema en cuestión a los usuarios finales y clientes. Así, ellos podrán experimentar con este modelo para constatar si cubre sus necesidades reales.
3. Generación de casos de prueba Los requerimientos deben ser comprobables. Si las pruebas para los requerimientos se diseñan como parte del proceso de validación, esto revela con frecuencia problemas en los requerimientos. Si una prueba es difícil o imposible de diseñar, esto generalmente significa que los requerimientos serán difíciles de implementar, por lo que deberían reconsiderarse.

## 4.7 Validación de Requerimientos

Los requerimientos para los grandes sistemas de software siempre cambian. Una razón es que dichos sistemas se desarrollaron por lo general para resolver problemas “horrorosos”: aquellos problemas que no se pueden definir por completo. Como el problema no se logra definir por completo, los requerimientos del software están condenados también a estar incompletos.

|  |
| --- |
| Requerimientos duraderos y volátiles: Algunos requerimientos son más susceptibles a cambiar que otros. Los requerimientos duraderos son los requerimientos que se asocian con las actividades centrales, de lento cambio, de una organización. También estos requerimientos se relacionan con actividades laborales fundamentales. Por el contrario, los requerimientos volátiles tienen más probabilidad de cambio. Se asocian por lo general con actividades de apoyo que reflejan cómo la organización hace su trabajo más que el trabajo en sí. |

Existen muchas razones por las que es inevitable el cambio:

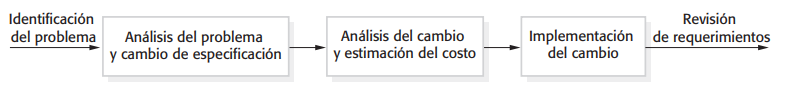
1. Los ambientes empresarial y técnico del sistema siempre cambian después de la instalación.
2. Los individuos que pagan por un sistema y los usuarios de dicho sistema, por lo general, no son los mismos.
3. Los sistemas grandes tienen regularmente una comunidad de usuarios diversa, en la cual muchos individuos tienen diferentes requerimientos y prioridades que quizás estén en conflicto o sean contradictorios.

La **administración de requerimientos** es el proceso de comprender y controlar los cambios en los requerimientos del sistema. Es necesario seguir la pista de requerimientos individuales y mantener los vínculos entre los requerimientos dependientes, de manera que pueda valorarse el efecto del cambio en los requerimientos. También es preciso establecer un proceso formal para hacer cambios a las propuestas y vincular éstos con los requerimientos del sistema. El **proceso formal de la administración de requerimientos** debe comenzar tan pronto como esté disponible un borrador del documento de requerimientos.

### 4.7.1 Planeación en la Administración de Requerimientos

Esta etapa establece el nivel de detalle que se requiere en la administración de requerimientos. Durante la etapa de administración de requerimientos, usted tiene que decidir sobre:

#### Administración del cambio de requerimientos



1. Identificación de requerimientos Cada requerimiento debe identificarse de manera exclusiva, de forma que pueda tener referencia cruzada con otros requerimientos y usarse en las evaluaciones de seguimiento.
2. Un proceso de administración del cambio Éste es el conjunto de actividades que valoran el efecto y costo de los cambios. En la siguiente sección se estudia con más detalle este proceso.
3. Políticas de seguimiento Dichas políticas definen las relaciones entre cada requerimiento, así como entre los requerimientos y el diseño del sistema que debe registrarse. La política de seguimiento también tiene que definir cómo mantener dichos registros.
4. Herramientas de apoyo La administración de requerimientos incluye el procesamiento de grandes cantidades de información acerca de los requerimientos. Las herramientas disponibles varían desde sistemas especializados de administración de requerimientos, hasta hojas de cálculo y sistemas de bases de datos simples.

La administración de requerimientos necesita apoyo automatizado y herramientas de software, para lo cual deben seleccionarse durante la fase de planeación. Se necesitan herramientas de apoyo para:

1. Almacenamiento de requerimientos Los requerimientos tienen que mantenerse en un almacén de datos administrado y seguro, que sea accesible para todos quienes intervienen en el proceso de ingeniería de requerimientos.
2. Administración del cambio El proceso de administración del cambio se simplifica si está disponible la herramienta de apoyo activa.
3. Administración del seguimiento Como se estudió anteriormente, la herramienta de apoyo para el seguimiento permite la identificación de requerimientos relacionados. Algunas herramientas que están disponibles usan técnicas de procesamiento en lenguaje natural, para ayudar a descubrir posibles relaciones entre los requerimientos.

### 4.7.2 Administración del cambio en los requerimientos

La administración del cambio en los requerimientos debe aplicarse a todos los cambios propuestos a los requerimientos de un sistema, después de aprobarse el documento de requerimientos. La administración del cambio es esencial porque es necesario determinar si los beneficios de implementar nuevos requerimientos están justificados por los costos de la implementación. La ventaja de usar un proceso formal para la administración del cambio es que todas las propuestas de cambio se tratan de manera consistente y los cambios al documento de requerimientos se realizan en una forma controlada. Existen tres etapas principales de un proceso de administración del cambio:

1. **Análisis del problema y especificación del cambio**
2. **Análisis del cambio y estimación del costo** El efecto del cambio propuesto se valora usando información de seguimiento y conocimiento general de los requerimientos del sistema. El costo por realizar el cambio se estima en términos de modificaciones al documento de requerimientos.
3. **Implementación del cambio** Se modifican el documento de requerimientos y, donde sea necesario, el diseño y la implementación del sistema.

Si un nuevo requerimiento tiene que implementarse urgentemente, siempre existe la tentación para cambiar el sistema y luego modificar de manera retrospectiva el documento de requerimientos. Hay que tratar de evitar esto, pues casi siempre conducirá a que la especificación de requerimientos y la implementación del sistema se salgan de ritmo.

## Puntos Clave

Texto

Descripción generada automáticamente