

Ingeniería de Requisitos: Actividades y Herramientas

Edgar Sarmiento Calisaya

Escuela Profesional de Ciencia de la Computación
Universidad Nacional de San Agustín de Arequipa,
Arequipa, Perú

Contenido

1. Ingeniería de Requisitos

- Elicitación
 - Léxico Extendido del Lenguaje
- Especificación
 - Usando Lenguaje Natural
 - Usando Modelos
- Análisis
 - Verificación
 - Validación
- Gestión

2. Documento de Requisitos

3. Herramientas

4. Conclusión

¿Qué son los requisitos de software ?



Descripciones

Son las descripciones de lo que un sistema debe hacer

Incluyen

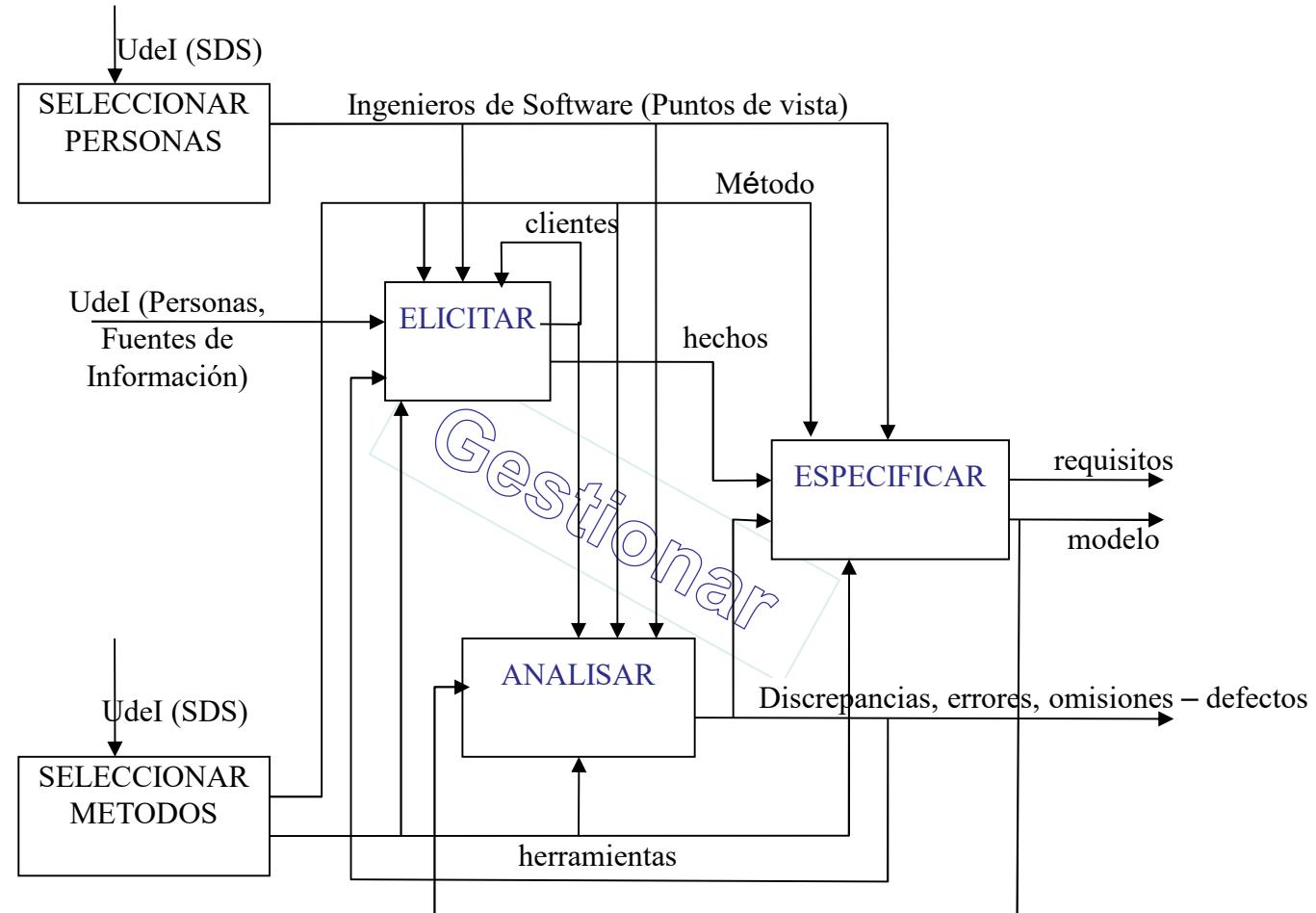
Los servicios prestados por el sistema, sus cualidades específicas y sus restricciones operativas.

Reflejan

Estos requisitos reflejan las necesidades reales de los usuarios de un sistema.

Principales Actividades de la Ingeniería de Requisitos

SADT – Actividades



Julio Leite, 1994

Actividades de la Ingeniería de Requisitos

1

Ingeniería de Requisitos (I.R.) Definición

La I.R. establece el proceso de definición de Requisitos como un proceso en el cual lo que debe hacer es **elicitar, modelar** y **analizar**. Este proceso debe manejar diferentes puntos de vista, y utilizar una combinación de métodos, herramientas y personal. El producto de este proceso es un modelo, del que se produce un documento de requisitos. Este proceso ocurre en un contexto previamente definido al que llamamos el **Universo de Información**.

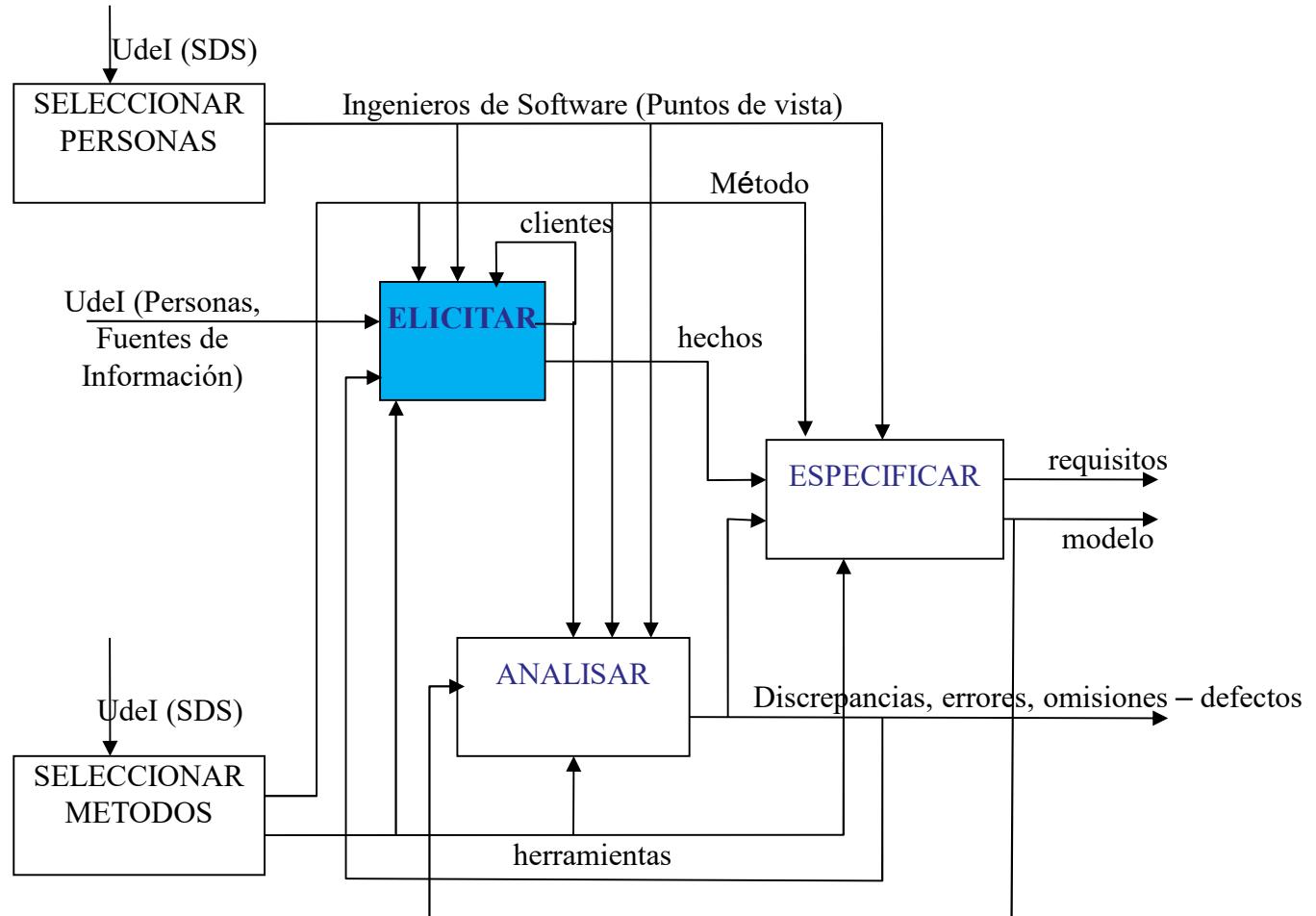
(Júlio Leite, 1994)

Universo de Información

Es el conjunto general en el que se desarrollará el software.

Incluye todas las fuentes de información y todas las personas relacionadas con el software, a las que denominamos agentes de ese universo. El **Udel** es la realidad circunstanciada por el conjunto de objetivos definidos por quien solicitó el software

Principales Actividades de la Ingeniería de Requisitos



1.1 Elicitación

- Elicitar = Eliciar(Provocar) + Aclarar + Extraer + Descubrir, hacer explícito, obtener el máximo de información para el conocimiento del objeto en cuestión.
 - Elicitar = Hacer salir, extraer, traer a la superficie (la verdad).
- HAY TRES ACTIVIDADES PRINCIPALES:
 - Identificación de fuentes de información
 - Recopilación de hechos
 - Comunicación

1.1 Elicitación

● NECESIDAD DE LA ELICITACIÓN:

- "There is no service in being precise about something when you do not even know what you are talking about" (von Neumann).
- RESOLUCIÓN DE PROBLEMAS:
 - Qué es lo desconocido?
 - Conoces un problema relacionado?
- COSTOS CRECIENTES PARA LA CORRECTA DE ERRORES

1.1 Elicitación

- REALIZA identificación de fuentes de información
- REALIZA Recolección de hechos
- REALIZA comunicación

- REALIZA/USA herramientas
- USA personal
- USA métodos
- DEPENDE de puntos de vista

Identificación de las fuentes de información

- Udel: Contiene toda la información necesaria
- Agentes (Actores, Usuarios)
- Otras fuentes de información:
 - Documentación del macrosistema
 - Políticas Manuales
 - Memos, actas, contratos ...
 - Libros sobre temas relacionado
 - Otros sistemas de la empresa
 - Otros sistemas externos .
 - Libros sobre temas relacionados

Identificación de las fuentes de información

● Importante:

- Priorizar las Fuentes de Información.

○ Heurísticas:

- Actores más importantes
- Documentos más mencionados
- Establecer Red de comunicaciones entre los componentes del macro-sistema
- Identificar grupos de interes
- Documentos más citados
- Libros u otras soluciones relacionadas con el tema – estado del arte

Recolección de Hechos

- Lectura de documentos
- Observación
- Entrevistas
- Cuestionarios
- Análisis de Protocolos
- Participación activa de los agentes de el Udel
- Reuniones
- Brainstorming
- Reutilización
- Recuperación (Ing. reversa) del proyecto de software

Recolección de Hechos

○ Heuristica:

- **Pregunte:** qué, por qué, cómo, por quién?,
- Pregunte lo obvio.
- Observe
- Aprenda
- Estudie
- **Vuelva a preguntar**
- Sea humilde

Comunicación

- ◉ Actividad fundamental para que la fase de elicitación tenga éxito.
- ◉ Se trata de la comunicación entre clientes / agentes y los ingenieros de software.
 - Presentación: de que manera la información es presentada
 - Entendimiento: establecimiento de un contexto común.
 - **Lenguaje**: entender el lenguaje de los clientes
 - Nivel de Abstracción
 - Retroalimentación

Comunicación

● Heuristicas:

- Una buena imagen vale mas que 1000 palabras
- Retroalimentacion (feedback)
- Evitar ruidos
- Evitar metaforas con tu area de conocimiento (informatica)
- Punto de vista del usuario
- Aprenda con humildad

Léxico Extendido del Lenguaje - LEL

- El LEL es un meta-modelo diseñado para ayudar a la **elicitación** y **representación del lenguaje usado en la aplicación – dominio**.
- Este modelo está centrado en la idea que una descripción de los términos del lenguaje mejora la comprensión del Udel.
- Para generar el LEL se registran **símbolos** (palabras o frases) peculiares o relevantes del dominio.
- Cada entrada del léxico se identifica con un **nombre** (o más de uno en caso de sinónimos) y tiene dos tipos de descripciones. Una llamada **Noción** que describe la denotación del símbolo y la otra **Impacto** que describe la connotación del mismo.
- Las entradas se **clasifican** en cuatro tipos de acuerdo a su uso general en el Udel. Estos tipos son: **Sujeto, Objeto, Verbo** y **Estado**.

Léxico Extendido del Lenguaje

Objetivo	Consecuencia
CONOCER EL VOCABULARIO DEL USUARIO	<ul style="list-style-type: none">• Asegurar la comunicación• Facilitar la validación de los requerimientos con el usuario• Mantener el mismo vocabulario durante todo el proceso de desarrollo
CONTAR CON UN INSTRUMENTO SIMPLE DE TRACEABILITY	<ul style="list-style-type: none">• Documentar consistentemente• Capacitar a nuevos miembros del equipo en la terminología empleada• Generar versiones del LEL a medida que evoluciona el proceso de desarrollo

Objetivo-Consecuencia del LEL (fuente Hadad 97)

Léxico Extendido del Lenguaje

- **Nombre:** Obra / Publicación
- **Tipo:** Objeto
- **Noción:**
 - Es un ítem de la colección.
 - Es un libro, folleto, tesis, publicación-PUC o periódico.
 - Puede ser una obra de referencia.
 - Puede ser una obra de tapa roja.
 - Puede tener más de un ejemplar.
- **Impacto:**
 - Después de la adquisición, el bibliotecario realiza el registro y el procesamiento técnico.
 - Después del procesamiento técnico, se puede localizar, consultar, prestar, devolver, renovar, reservar y prestar para fotocopiar.

Periódico
Tipo: Objeto
Noción:
...
Impacto:
...

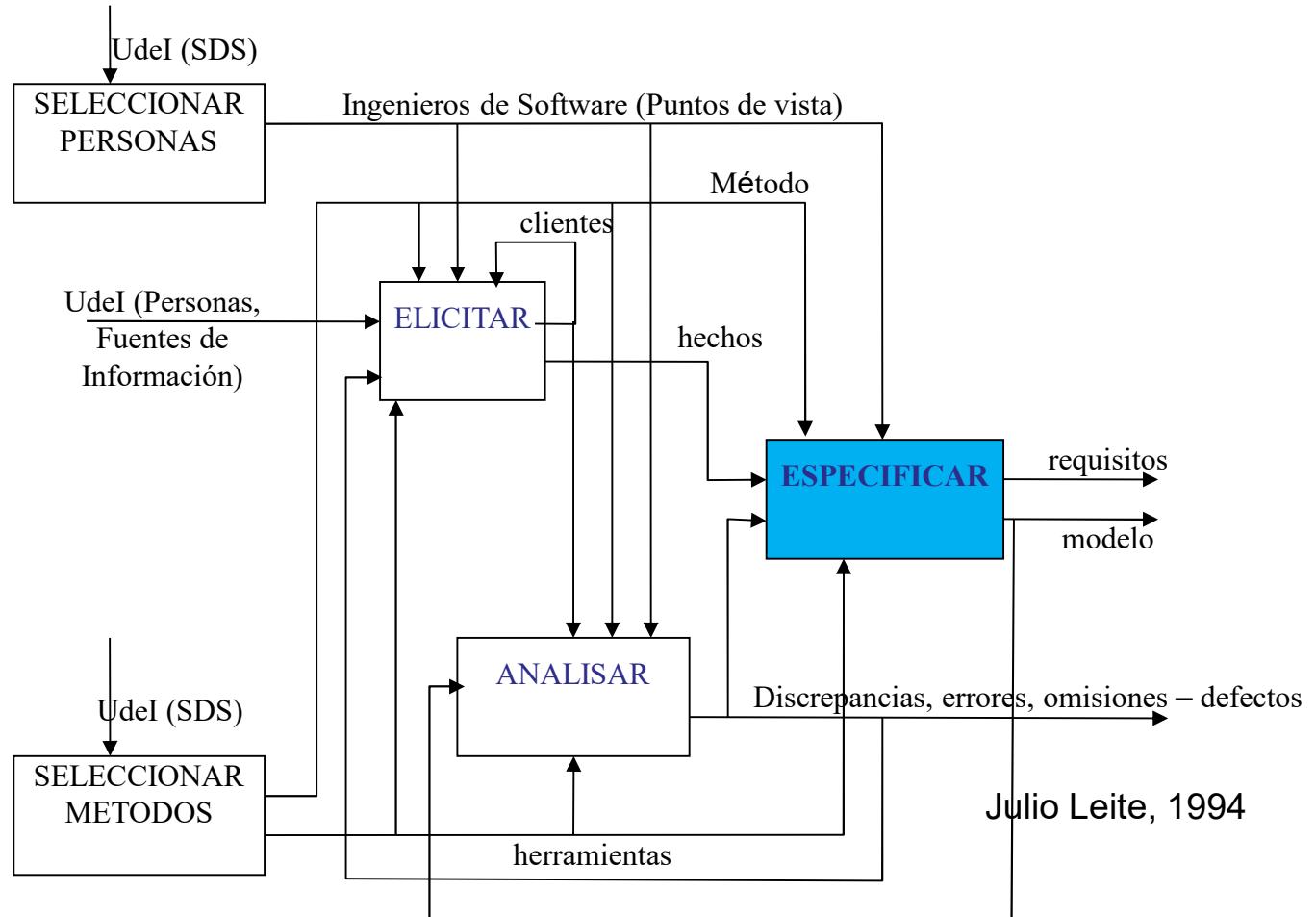
Léxico Extendido del Lenguaje

- Al describir los símbolos, dos **principios** deben cumplirse:
 - ◉ El principio de **circularidad** que postula la **maximización del uso de símbolos** en la descripción de **otros símbolos**;
 - ◉ El principio de **vocabulario mínimo** que postula la **minimización del uso de términos** que son **externos** al léxico

Un enfoque similar es usado en la práctica de desarrollo denominada “*Domain-driven Design - DDD*”: [Ubiquitous Language](#) (Eric Evans, 2003)

BDD – Behavior-driven development: [Ubiquitous Language](#)

Principales Actividades de la Ingeniería de Requisitos



1.2 ESPECIFICACIÓN

- Construcción de modelos del sistema utilizando técnicas y métodos.
- Hay tres actividades:
 - Representación
 - Organización
 - Almacenamiento

1.2 ESPECIFICACIÓN

REALIZA Representación

REALIZA Organización

REALIZA Almacenamiento

USA Personal

USA Métodos

USA Herramientas

DEPENDE DE Puntos de Vista

Representación:

Tipos
Relaciones
Operaciones

Organización:

Niveles de Abstracción
Reglas de Refinamiento
Reglas de Consistencia Interna

Almacenamiento:

clasificación
Indexación
Aspectos Generales

1.2 ESPECIFICACIÓN

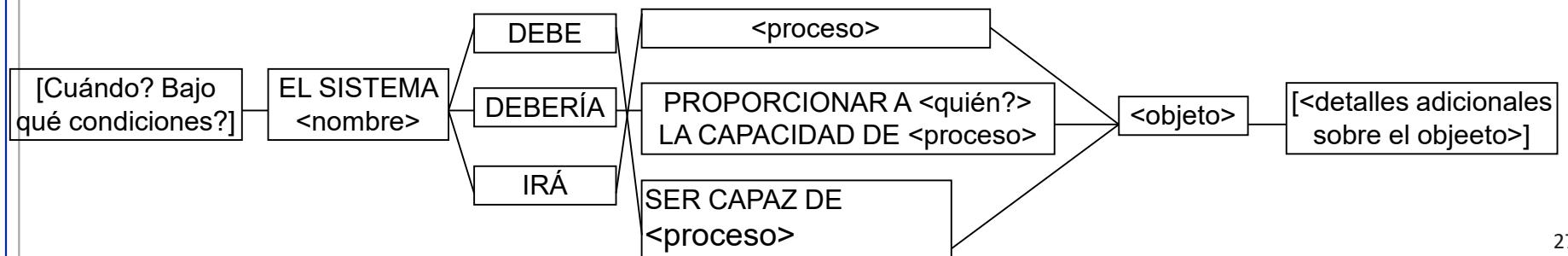
- Una vez que los requisitos son elicítados necesitan ser descritos en un documento de **Especificación de Requisitos de Software: *Requisitos de Usuario o Requisitos de Sistema***
- **Métodos / Técnicas :**
 - Especificación Usando Lenguaje Natural
 - Especificación Usando Modelos

Especificación Usando Lenguaje Natural

- Se han propuesto varios **templates/formatos/plantillas** para definir requisitos de usuario (o requisitos de alto nivel).
- Un **template** organiza la estructura sintáctica de un requisito en una serie de **partes predefinidas**
- Mostramos tres **templates**, a saber, las plantillas de **Rupp**, **EARS** y **Cohn**.
- Esta elección está motivada por el uso de estas plantillas en la **industria**.

Especificación Usando Lenguaje Natural

- **Rupp's template:** está hecha de **seis partes** (la primera y la última son opcionales) y distingue tres tipos de funcionalidades:
 - **Requisito autónomo:** actividad que el sistema realiza de forma autónoma, el usuario no interactúa con la actividad.
 - **Requisito de interacción del usuario:** el sistema interactúa directamente con un usuario (por ejemplo, mediante una interfaz de entrada).
 - **Requisito de interfaz:** el sistema depende de los sistemas vecinos para ejecutar una actividad específica.



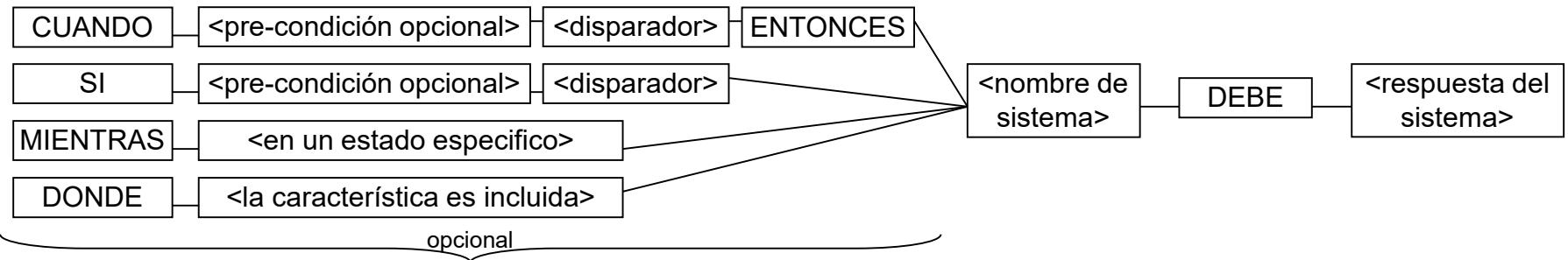
Especificación Usando Lenguaje Natural

◉ Rupp's template:

Tipo	Template/Ejemplo
<i>Requisito Autonomo</i>	El <nombre del sistema> DEBE / DEBERÍA / IRÁ <verbo de proceso> <objeto> Tan pronto como se detecte un corte de energía, el módulo de Vigilancia y Seguimiento DEBE registrar una advertencia en el archivo de registro de alertas del sistema.
<i>Requisito Interacción del Usuario</i>	El <nombre del sistema> DEBE / DEBERÍA / IRÁ PROPORCIONAR A <¿a quién?> LA CAPACIDAD DE <verbo de proceso> <objeto> El módulo de Vigilancia y Seguimiento DEBE PROPORCIONAR A el administrador del sistema LA CAPACIDAD DE monitorear los cambios del sistema de configuración publicados en la base de datos.
<i>Requisito Interface</i>	El <nombre del sistema> DEBE / DEBERÍA / IRÁ SER CAPAZ DE <verbo de proceso> <objeto> El módulo de Vigilancia y Seguimiento DEBE SER CAPAZ DE recibir datos de las cámaras de la biblioteca.

Especificación Usando Lenguaje Natural

- **EARS template:** está hecha de **cuatro partes** (la primera es opcional) y distingue cinco tipos de funcionalidades:
 - **Requisito ubicuo:** sin condición previa, y se usa para los requisitos que siempre están activos.
 - **Requisito dirigido por evento:** se inicia solo cuando se detecta un evento desencadenante en el límite del sistema. Comienza con la palabra clave WHEN.
 - **Requisito de comportamiento no deseado:** se usa para cubrir todas las situaciones no deseadas (fallas, alteraciones, desvíos). Utiliza las palabras clave IF / THEN.
 - **Requisito dirigido por estado:** está activo mientras el sistema está en un estado definido. Se utiliza la palabra clave WHILE.
 - **Requisito de característica opcional:** es aplicable solo en sistemas que incluyen una característica particular. Se utiliza la palabra clave WHILE.
 - **Requisito complejo:** incluye cláusulas condicionales complejas, combinaciones de las palabras clave WHEN, WHILE y WHERE que pueden ser necesarias.



Especificación Usando Lenguaje Natural



EARS template:

Tipo	Template/Ejemplo
Requisito ubicuo	El <nombre del sistema> DEBE <respuesta del sistema> El sistema de control DEBE evitar el exceso de velocidad del motor
Requisito dirigido por evento	CUANDO <condiciones previas opcionales> <disparador> el <nombre del sistema> DEBE <respuesta del sistema> CUANDO el encendido continuo es ordenado por la aeronave, el sistema de control DEBE encender el encendido continuo.
Requisito de comportamiento no deseado	SI <condiciones previas opcionales> <disparador>, LUEGO el <nombre del sistema> DEBE <respuesta del sistema> SI el indicador de falla de velocidad aerodinámica es calculada, ENTONCES , el sistema de control DEBE usar velocidad aerodinámica modelada.
Requisito dirigido por estado	MIENTRAS que <en un estado específico> el <nombre del sistema> deberá <respuesta del sistema> MIENTRAS que la aeronave está en vuelo, el sistema de control DEBE mantener el flujo de combustible del motor por encima de XXlbs / seg.
Requisito de característica opcional	DONDE <se incluye la característica> el <nombre del sistema> DEBE <respuesta del sistema> DONDE el sistema de control incluya una función de protección de exceso de velocidad, el sistema de control DEBE probar la disponibilidad de la función de protección de exceso de velocidad antes del despacho de la aeronave.
Requisito complejo	MIENTRAS que la aeronave está en tierra, cuando se ordena el impulso de retroceso, el sistema de control DEBE habilitar el despliegue del inversor de empuje.

Especificación Usando Lenguaje Natural

- **Cohn's template – Historia de Usuario:** Una historia de usuario es una breve descripción de la funcionalidad o característica contada desde la perspectiva de la persona que desea la capacidad, generalmente un usuario o un comprador de un sistema o software.
- Las historias de usuario se escriben a mano tradicionalmente en tarjetas de notas en papel (story card) y se componen de tres aspectos:
 - **descripción** escrita de la historia utilizada para la planificación y como recordatorio.
 - **conversaciones** sobre la historia que sirven para dar forma a los detalles.
 - **pruebas** que transmiten y documentan detalles y que se pueden utilizar para determinar si la historia esta completa.

A company can pay for a job posting with a credit card.

Note: Will we accept Discover cards?

Note for UI: Don't have a field for card type (it can be derived from first two digits on the card).

Test with Visa, MasterCard and American Express (pass).

Test with Diner's Club (fail).

Test with good, bad and missing card ID numbers.

Test with expired cards.

Test with over \$100 and under \$100

Especificación Usando Lenguaje Natural

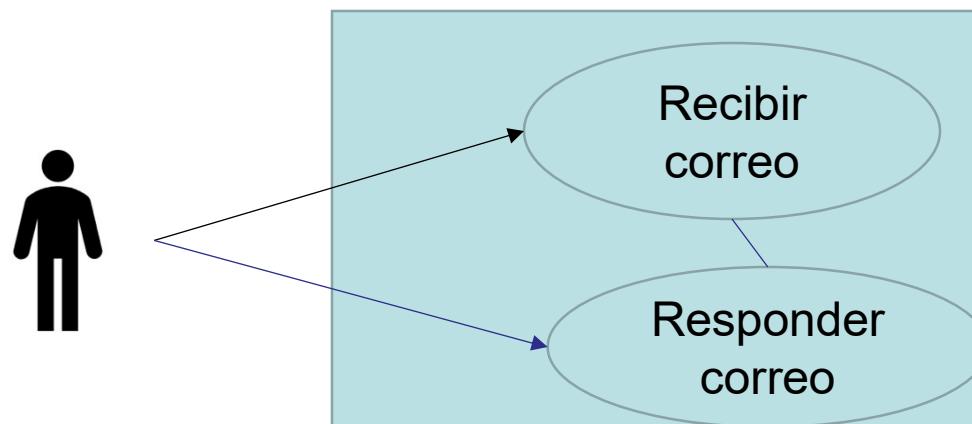
● Cohn's template – Historia de Usuario

● El template más conocido, popularizado por Cohn es:

COMO UN <tipo de usuario>, QUIERO/DESEO <objetivo>, [DE MANERA QUE <alguna razón>]

Por ejemplo:

COMO UN Administrador, QUIERO recibir un correo electrónico cuando se envíe un formulario de contacto, DE MANERA QUE pueda responder a él.



Especificación Usando Modelos

- **Orientado a Metas:** describen las intenciones de los stakeholders o de grupos de stakeholders. Metas pueden estar en conflicto;
 - Arboles AND/OR
 - KAOS
 - I star
 - NFR Framework
- **Orientados a interacción usuario-sistema:** Casos de uso o escenarios: documentan secuencias de uso del sistema;
 - Diagramas de casos de uso
 - Descripciones de casos de uso o escenarios
 - Behavior-driven Development – BDD
- **Requisitos del sistema** (generalmente conocidos como requisitos): describen Funciones y cualidades detalladas que el sistema a desarrollar debe implementar o poseer.
 - Diagramas de Entidad - Relación
 - Diagramas de Clases
 - Diagramas de flujo de datos
 - Diagramas de actividad UML
 - Diagramas de estado UML
 - Statecharts
 - Redes de Petri

p.e. para diseñar
software con
cierta autonomía

Casos de Uso

○ Casos de Uso

- **Diagramas de casos de uso** documentan las **funciones** relevantes del sistema a partir de la perspectiva del usuario, así como las **relaciones** entre funciones o entre las funciones y determinados aspectos del **contexto**.
- **Diagramas de casos de uso** **no** documentan **información** sobre **casos de uso individuales**, tal como la **interacción entre un caso y un actor**;
- Esta **información** es documentada **textualmente** a través de **templates** de casos de uso o escenarios.

Diagramas de Casos de Uso

- **Sokoban** es un juego de varios niveles.
- Cada nivel está compuesto por un jugador, cajas, repisas y muros.
- El objetivo del jugador es el repisas.
- Cuando esto sucede el jugador pierde una vida.
- Para mover una caja, el jugador la empujará. Si la casilla hacia donde se empuja la caja está libre la caja se moverá.
- Si el jugador se queda bloqueado no podrá terminar el nivel, puede reiniciar el nivel perdiendo una vida.
- Cuando el jugador pierde todas sus vidas la partida termina.



Este diagrama de casos de uso es correcto pero muy pobre.

Descripción de Casos de Uso/Escenario

- ◉ Según Leite et al., **escenario** es "una técnica de descripción que se centra tanto en el proceso como en el usuario".
- ◉ Según Glinz, los **casos de uso o escenarios** se describen como "un conjunto ordenado de interacciones entre socios, generalmente entre un sistema y un conjunto de actores externos al sistema".
- ◉ Se usa ampliamente en ingeniería de requisitos porque **ayuda a los ingenieros, desarrolladores y otros stakeholders** a **comprender mejor los requisitos** del software y su interfaz con el entorno.
- ◉ Glinz → Escenario = Caso De Uso
- ◉ Cockburn → Escenario = camino en un caso de uso

Descripción de Casos de Uso

- Hay una gran variedad de **templates** de casos de uso en la literatura, cada una con propósitos muy diferentes.
- Los **templates** de casos de uso hacen hincapié en especificar el **flujo principal** y los **flujos alternativos**.
- Cada **template** de casos de uso utiliza un **formato** de una o dos **columnas** para especificar las **descripciones textuales** en un texto plano.

Descripción de Casos de Uso

Elemento	Descripción
Titulo/Nombre	<identificación del caso de uso>
Objetivo/Descripción	<breve descripción del propósito del caso de uso>
Pre-condición	<Que debería ser VERDADERO antes que el caso de uso inicie>
Pos-condición	< Que debería ser VERDADERO despues que el caso de uso termine>
Actor	<Entidades activas directamente involucradas en la situación>
Recursos	<Entidades pasivas usadas durante la ejecución del caso de uso>
Episodios/Flujo Principal	<p>El flujo de eventos más común y exitoso.</p> <p>Acciones o eventos - se numeran secuencialmente.</p> <p><Paso> <Acción o nombre de otro caso de uso></p> <p>...</p>
Alternativas	<p>Describe una desviación en otro flujo de eventos.</p> <p>Un flujo alternativo siempre depende de una condición que ocurra en un paso específico en un flujo de referencia.</p> <p>Un flujo alternativo se compone de una secuencia de pasos numerados.</p> <p><Paso> <Referencia> <Condición que causa la ramificación></p> <p style="padding-left: 40px;"><Acción o nombre de otro caso de uso></p> <p>...</p>

Descripción de Casos de Uso

Titulo/Nombre	Mover jugador
Objetivo/Descripción	El usuario desea mover al jugador
Pre-condición	Partida iniciada
Pos-condición	
Actor	Usuario
Recursos	Pantalla
Episodios/Flujo Principal	<ol style="list-style-type: none">1 El usuario solicita realizar un movimiento.2 El sistema comprueba que el movimiento es válido y lo realiza.3 El sistema muestra la pantalla de juego con la posición actual del jugador y las cajas4 El sistema comprueba si el usuario ha completado el nivel, muestra la pantalla de juego y espera un nuevo movimiento
Alternativas	<ol style="list-style-type: none">2.1 Si el movimiento no es válido, el sistema no hace nada y espera un nuevo movimiento.4.1 Si el usuario ha completado el nivel, el sistema carga el siguiente nivel, muestra la pantalla de juego, y espera un nuevo movimiento.

Descripción de Casos de Uso

- La mayoría de las recomendaciones para escribir sentencias en casos de uso están basadas en el formato **<SUJETO> <PREDICADO>**

FORMATO/Ejemplo

<SUJETO> <VERBO> <OBJETO>

El cliente examina la oferta

<SUJETO> <VERBO> <OBJETO> <FRASE PREPOSICIONAL>

El sistema envía la oferta al cliente.

<SUJETO> <VERBO> <FRASE PREPOSICIONAL>

El sistema pregunta por el login del cliente

Behavior-driven Development – BDD

- Com BDD, Behavior-Driven Development (desenvolvimento orientado a comportamento) , são especificados cenários de funcionamento para a aplicação e os testes são desenvolvidos com base nesses cenários.
 - TBD – en elaboración

- <https://medium.com/@jrnmagalhaes/unit-test-x-tdd-x-bdd-913278f33b80>
- <https://blog.locaweb.com.br/artigos/metodologias-ageis/diferenca-entre-bdd-tdd/>
- <https://cio.com.br/do-bdd-ao-tdd-os-pros-e-contras-de-varias-tecnicas-agile/>
- <https://www.devmedia.com.br/desenvolvimento-orientado-por-comportamento-bdd/21127>
- <https://www.eduardopires.net.br/2012/06/ddd-tdd-bdd/>

Behavior-driven Development – BDD

A funcionalidade é escrita segundo um padrão:

- Funcionalidade : [Nome]
- Para [Valor ao Negócio] Eu, como [Papel]
Desejo poder realizar [Funcionalidade]

Dessa forma fica muito claro a todos quais são os objetivos que uma funcionalidade deseja atingir e é de muito fácil entendimento.

Os cenários acompanham a funcionalidade de forma que demonstram comportamentos para atendê-la:

- Cenário : [Nome]
 - Dado que [Estado inicial do sistema]
 - Quando [Ação a ser realizada no sistema]
 - Então [Coisas que o sistema deve fazer após a ação do Quando]

Feature: BDD implementation using **Cucumber**

As a student I want to log in to cucumber so that I can write requirements and test using the **Gherkin language.**

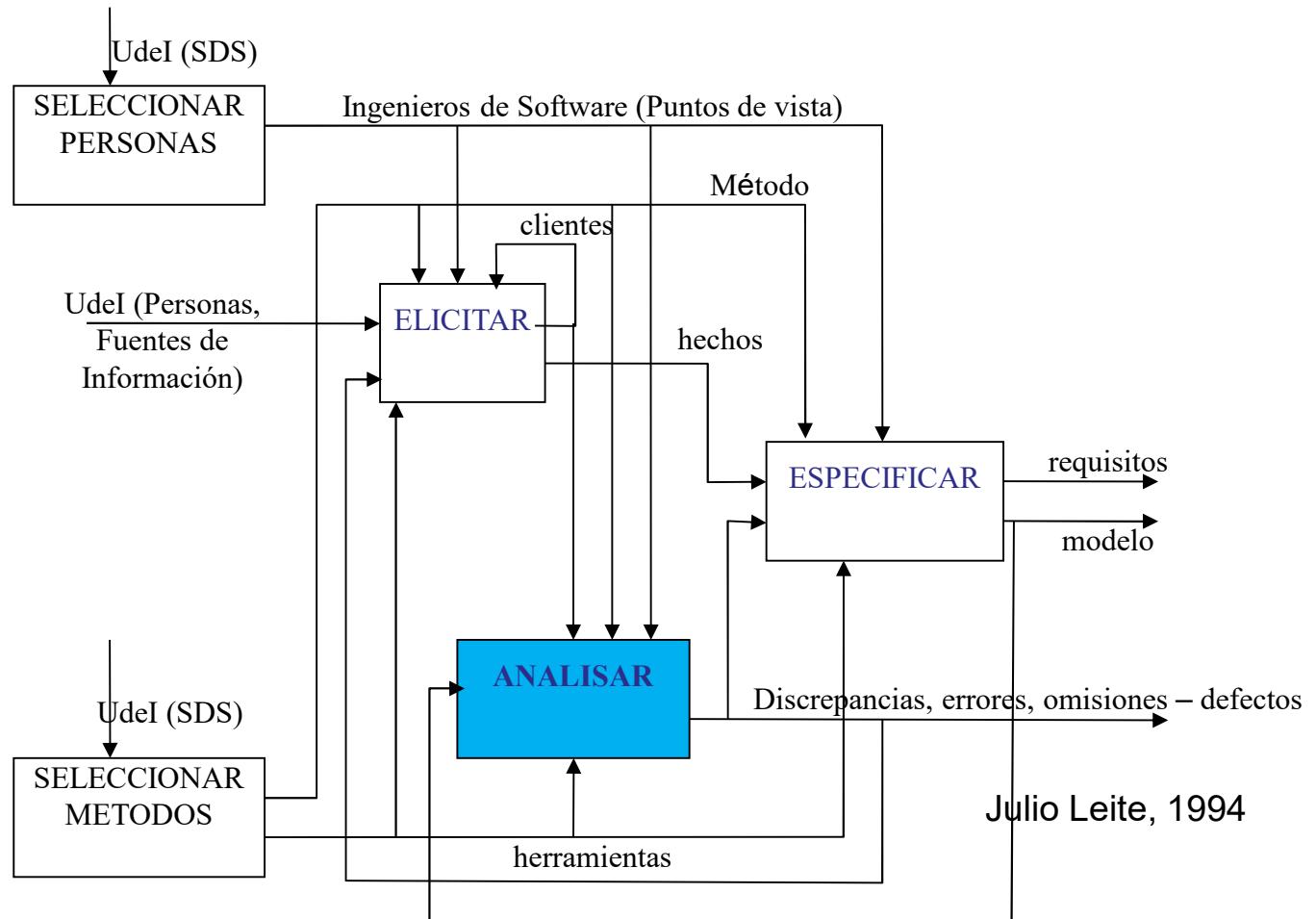
Scenario: Login to G-mail using Cucumber plugin

Given User is navigating to G-mail Login Page

When User need to enter username as "Username" and password as "Password" AND ...

Then User is successfully navigated to the G-mail Mail Box

Principales Actividades de la Ingeniería de Requisitos



1.3 Análisis

- Fundamental para el éxito del proceso de desarrollo del software.
- Implica **tres actividades principales:**
 - Identificación de partes
 - Verificación
 - Validación

1.3 Análisis

REALIZA Identificación de Partes

REALIZA Verificación

REALIZA Validación

USA Personal

USA Métodos

USA Herramientas

DEPENDE DE Puntos de Vista

Identificación de Partes::

Organización

Almacenamiento

1.3 Análisis

- **Verificación:** software bien construido, libre de defectos?
 - Análisis usando métodos formales, como Z y VDM
 - **Inspección**
 - Checklists
 - Simulación
- **Validación:** software satisface las necesidades reales del usuario?
 - Uso de comprobación informal
 - Uso de **prototipos**
 - Reutilización de dominios
 - Uso de puntos de vista
 - **CON LA PARTICIPACIÓN DEL USUARIO!**

Objetivo:
Especificación
libre de
Discrepancias
(conflictos),
Errores,
Omisiones –
DEFECTOS

Inspección

- El **objetivo** de la inspección es verificar si el producto producido por un proceso de producción está de acuerdo con los criterios de éxito del proceso.
- LAS INSPECCIONES SON FUNDAMENTALES PARA EL ASEGURAMIENTO DE LA CALIDAD
- La inspección **Fagan**, cuyo nombre proviene de su inventor Michael E. Fagan, ingeniero de **IBM** en los años setenta, es sin duda la técnica más potente para la revisión de documentos, contratos, código, etc.
- **Objetivo** de las inspecciones Fagan:
 - Encontrar tempranamente los defectos.
 - Prevenir el mal funcionamiento de los procesos o planes establecidos.
 - Proporcionar mejoras en la fiabilidad, disponibilidad y factibilidad del mantenimiento de software.
 - Descubrir continuamente la información técnica, asociada con las funciones, formularios, actividades internas que aseguran el producto.
 - Continuar el mejoramiento del proceso de desarrollo.
 - Establecer una igualdad de conocimiento dentro de los desarrolladores para la buena práctica de estándares y técnicas de desarrollo.

Inspección

- Un **equipo** de inspección de **Fagan** consiste en: un **asesor**, un **lector**, un **inspector**, y un **autor**.
 - El **asesor** debe asegurarse de que la preparación del equipo se centre en la detección de defectos sin desviarse.
 - El papel del **lector** es explicar y leer el producto que será inspeccionado a un paso razonable, pues el procedimiento se puede hacer de manera rápida, pero tal vez no se inspeccione lo que se desea.
 - El papel de un **inspector** es **examinar** el software desde el punto de vista de un usuario.
 - La presencia del **autor** en las reuniones de la inspección, generalmente se considera beneficiosa la presencia del autor porque puede asistir al equipo de la inspección para entender mejor el producto, además está preparado para entender la naturaleza de los defectos en los hallazgos del equipo.

Inspección

El método Fagan establece 6 pasos:

Planificación	Cuando los materiales para ser inspeccionados pasan por los criterios de entrada (por ejemplo, el código fuente compila con éxito sin errores de sintaxis), miembros del equipo de inspección se seleccionan, y se establecen los horario de la inspección (por ejemplo, tiempo y lugar).
Visión General	Se dan instrucciones previas a los miembros del equipo del material a ser inspeccionado, y se asignan los papeles.
Preparación	Los miembros del equipo estudian el material individualmente para prepararse para satisfacer los papeles asignados.
Inspección	El equipo realiza una reunión de inspección para encontrar defectos, y registrarlos. El propósito es la detección de los defectos o de violaciones de estándares, alternativas debe ser eliminada por el asesor.
Corrección	El autor revisa el resumen de los defectos detectados, clarificando cuales son realmente defectos y que son mal entendidos en el proceso de la inspección. Entonces, el autor debe modificar para corregir los defectos.
Seguimiento	El asesor o el equipo entero de inspección repasa el producto otra vez, para asegurar que todos los arreglos son eficaces y de que no se ha introducido ningún defecto adicional durante la remodelación.

1.3 Análisis

- Heurísticas de verificación de **Requisitos en Alto Nivel (YUE, 2013; ARORA, 2015)**

Description
Describe one action per sentence. A simple sentence must contain only one subject and one verb.
The present simple tense and active form of a verb should be used (user point of view).
Use declarative sentence only. “Is the system idle?” is a non-declarative sentence.
Use words in a consistent way. Keep one term to describe one thing.
Don’t use modal verbs (e.g., might, may, should, can, must)
Avoid adverbs (e.g., very).
Don’t use negative adverb and adjective (e.g., hardly, never), but it is allowed to use not or no.
Don’t use pronouns (e.g. he, this)
Don’t use participle phrases as adverbial modifier. For example, the italic-font part of the sentence “ATM is idle, displaying a Welcome message” is a participle phrase.
Use “the system” to refer to the system under design consistently.

1.3 Análisis

- Heurísticas de verificación de **Historias de Usuario** (Lucasen, 2015)

Atomic	A user story expresses a requirement for exactly one feature
Minimal	A user story contains nothing more than role, means and ends
Well-formed	A user story includes at least a role and a means
Conflict-free	A user story should not be inconsistent with any other user story
Conceptually sound	The means expresses a feature and the ends expresses a rationale, not something else
Problem-oriented	A user story only specifies the problem, not the solution to it
Unambiguous	A user story avoids terms or abstractions that may lead to multiple interpretations
Complete	Implementing a set of user stories creates a feature-complete application, no steps are missing
Explicit dependencies	Link all unavoidable, non-obvious dependencies on user stories
Full sentence	A user story is a well-formed full sentence
Independent	The user story is self-contained, avoiding inherent dependencies on other user stories
Scalable	User stories do not denote too coarse-grained requirements that are difficult to plan and prioritize
Uniform	All user stories follow roughly the same template
Unique	Every user story is unique, duplicates are avoided

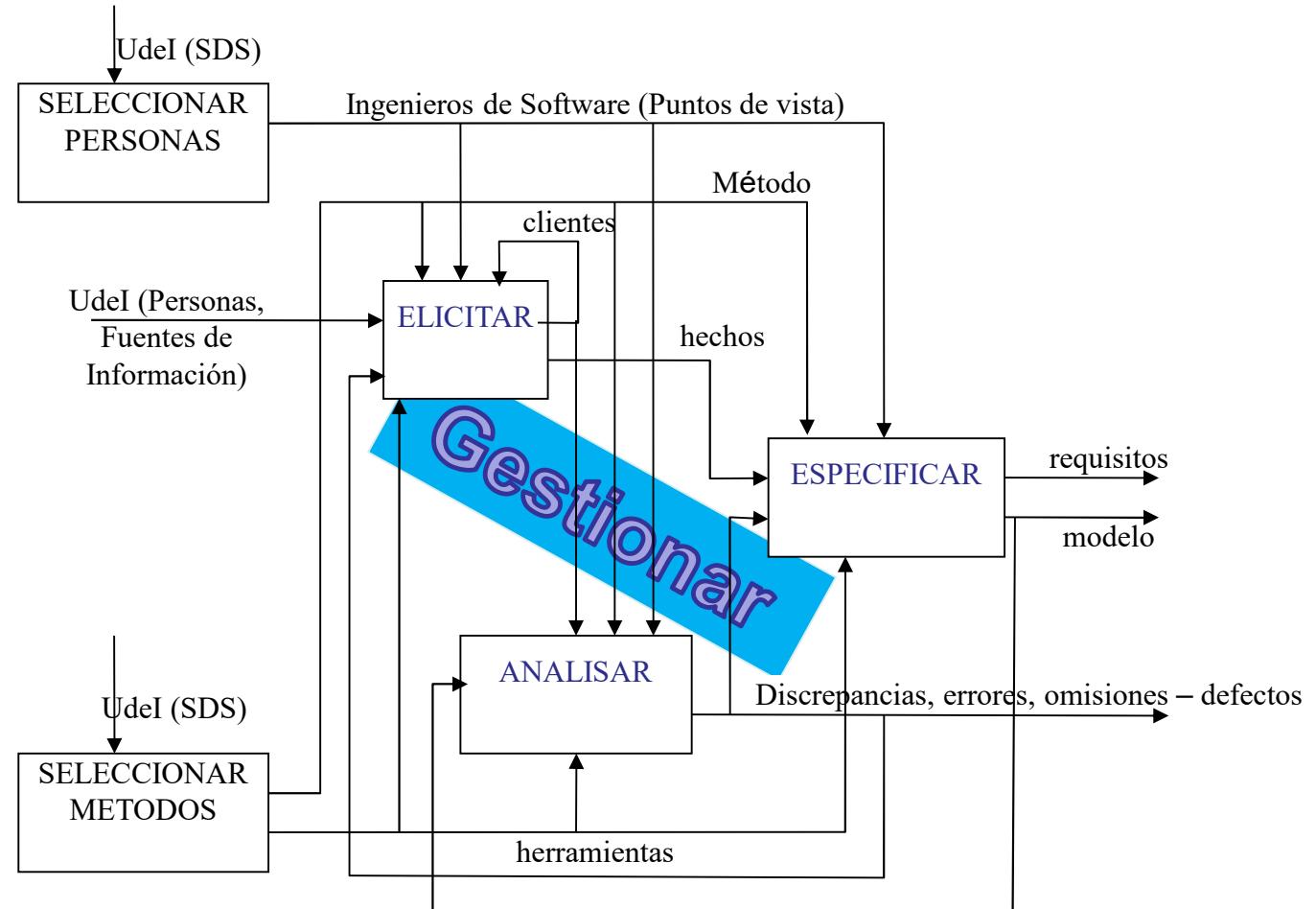
1.3 Análisis

- Heurísticas de verificación de **Casos de Uso** (**Cockburn, 2000; Leite, 2001; YUE, 2013**)

Description
Check that Title defines exactly one situation, i.e., one verb in infinitive (base) form and an object
Check that Goal satisfies exactly one purpose
The subject of a sentence in main and alternative flows should be the system or an actor.
Describe the flow of events sequentially.
Actor-to-actor interactions are not allowed.
Clearly describe the interaction between the system and actors without omitting its sender and receiver.
Check the existence of more than two and less to 10 main flow steps.
Check that every actor participates in at least one step.
Check that every actor mentioned in main flow is included in the actor component.
Check that every resource is used in at least one main flow step.
Check that every resource mentioned in main flow steps is included in the resources component.
Check coherence between the title and the goal.
Ensure that the set of main flow steps satisfies the goal and is within the context.
Ensure that main flow steps contain only actions to be performed.
Check that every use case referenced in main and alternative flows exists within the set of scenarios.
Check that the set of main flow steps of every scenario is not already included in another scenario.
Check that complex alternate/exception (more than 2 steps) is treated by a scenario.

Principales Actividades de la Ingeniería de Requisitos

SADT – Actividades



Julio Leite, 1994

1.4 Gestión

- Gestión de requisitos es transversal al proceso de IR, y comprende las siguientes actividades:
 - Definición de atributos,
 - Visualización,
 - Priorización,
 - Trazabilidad,
 - Control de versión,
 - Configuración y Baseline
 - Control de cambio.

Definición de Atributos

Designación de Atributos para requisitos basado en templates

Attribute Type	Meaning
Identifier	Short, unique identifier of a requirement artifact from the set of all regarded requirements.
Name	Unique, characterizing name.
Description	Briefly describes the content of the requirement.
Version	Current version of the requirement.
Author	Specifies the author of the requirement.
Source	Specifies the source or sources of the requirement.
Stability	Specifies the approximate stability of the requirement. The stability is the amount of changes that are to be expected with regard to the requirement. Possible values can be "fixed", "established", and "volatile".
Risk	Specifies the risk based on an estimate of the amount of damage and loss and the probability of occurrence.
Priority	Specifies the priority of the requirement regarding the chosen prioritization properties, e.g., "importance for market acceptance", "order of implementation", "loss/opportunity cost if not implemented".

Table 8-1 Frequently used attribute types

Definición de Atributos

Attribute Type	Meaning
Person responsible	Specifies the person, group of stakeholders, organization, or organizational unit that is responsible for the content of the requirement.
Requirement type	Specifies the type of requirement (e.g., "functional requirement", "quality requirement", or "constraint") depending on the applied classification scheme.
Status regarding the content	Specifies the current status of the content of the requirement, e.g., "idea", "concept", "detailed content".
Status regarding the validation	Specifies the current status of the validation, e.g., "unvalidated", "erroneous", "in correction".
Status regarding the agreement	Specifies the current status of the negotiation, e.g., "not negotiated", "negotiated", "conflicting".
Effort	Estimated/actual effort to implement the requirement
Release	The designation of the release in which the requirement shall be implemented.
Legal obligation	Specifies the degree of legal obligation of the requirement.
Cross references	Specifies relations to other requirements, for example, if it is known that the implementation of the requirement requires prior implementation of another requirement.
General information	In this attribute, arbitrary information that is considered relevant can be documented, for example, if the negotiation of this requirement is scheduled for discussion during the next meeting with the stakeholders.

Table 8-2 Additional types of requirement attributes

Definición de Atributos

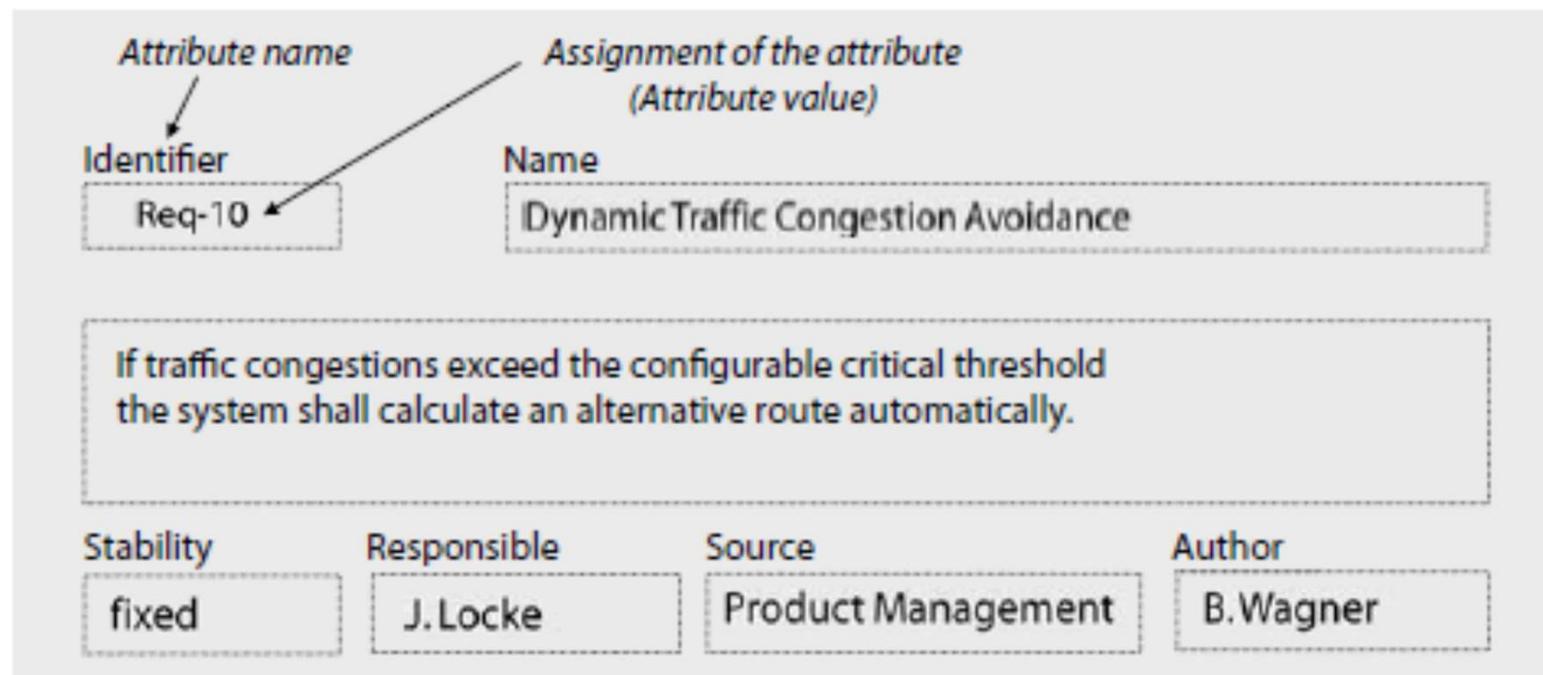


Figure 8-1 Example of requirement attribute assignment

Definición de Atributos

Titulo/Nombre	Mover jugador
Objetivo/Descripción	El usuario desea mover al jugador
Pre-condición	Partida iniciada
Pos-condición	
Actor	Usuario
<i>Recursos</i>	Pantalla
Episodios/Flujo Principal	<ol style="list-style-type: none">1 El usuario solicita realizar un movimiento.2 El sistema comprueba que el movimiento es válido y lo realiza.3 El sistema muestra la pantalla de juego con la posición actual del jugador y las cajas4 El sistema comprueba si el usuario ha completado el nivel, muestra la pantalla de juego y espera un nuevo movimiento
Alternativas	<p>2.1 Si el movimiento no es válido, el sistema no hace nada y espera un nuevo movimiento.</p> <p>4.1 Si el usuario ha completado el nivel, el sistema carga el siguiente nivel, muestra la pantalla de juego, y espera un nuevo movimiento.</p>

Visualización de Requisitos

Definición de visualizaciones a partir de roles/papeles específicos

Visualizaciones selectivas:

- Seleccionar requisitos
- Ocultar atributos
- Combinar ambos principios

Visualizaciones consolidadas:

- Datos generados o consolidados que no están en descritos en los atributos de los requisitos

Visualizaciones selectivas:

Requirements Basis

Identifier	Name	Description	Author	Source	Responsible	Stability	Status Content	Status Validation	X-Ref
Req-1	"Keyboard Input ..."	"The System ..."	J. Locke	PM	P. Wagner	fixed	concept	unvalidated	Req-3; Req-9
Req-2	"Voice Input ..."	"The System ..."	E. Kurt	PM	P. Wagner	established	concept	in validation	Req-5; Req-123
Req-3	"Reception of ..."	"The System ..."	H. Escher	Maintanence	P. Wagner	established	idea	unvalidated	Req-4; Req-1
Req-4	"Remote Diagn ..."	"The System ..."	M. Bom	Maintanence	M. Bom	volatile	idea	unvalidated	Req-47
Req-5	"Input of miscel ..."	"The System ..."	H. Miller	F. Goldstein	H. Miller	fixed	concept	in validation	Req-33
Req-6	"Acessing reco ..."	"The System ..."	J. Locke	F. Goldstein	M. Bom	fixed	detailed content	validated	Req-45; Req-11
Req-7	"Automatic ..."	"The System ..."	M. Bom	H. Licht	M. Bom	fixed	detailed content	in correction	Req-11
Req-8	"Display of ..."	"The System ..."	H. Miller	J. Locke	M. Bom	volatile	idea	unvalidated	Req-11
Req-9	"Input of miscel ..."	"The System ..."	J. Locke	J. Locke	P. Wagner	volatile	concept	in validation	Req-49
Req-10	"Dynamic ..."	"The System ..."	M. Bom	Customer	P. Wagner	established	detailed content	erroneous	Req-51; Req-9
Req-11	"Voice Control ..."	"The System ..."	H. Miller	Customer	P. Wagner	established	concept	validated	Req-7; Req-81; Req-6
...

Views

Identifier	Name	Description	Author
Req-2	"Voice Input ..."	"The System ..."	E. Kurt
Req-3	"Reception of ..."	"The System ..."	H. Escher
Req-10	"Dynamic ..."	"The System ..."	M. Bom
Req-11	"Voice Control ..."	"The System ..."	H. Miller

Selection: Requirements that "J. Locke" is responsible for and have a stability of "fixed."

Display: "Identifier", "Name", "Description", "Author"



Identifier	Name	Author
Req-1	"Keyboard Input ..."	J. Locke
...

Selection: Requirements that are not validated and originate from the source "Product Management" (PM).

Display: "Identifier", "Name", "Author"

Identifier	Name	Description	Author	Source
Req-6	"Acessing reco ..."	"The System ..."	J. Locke	F. Goldstein
Req-7	"Automatic ..."	"The System ..."	M. Bom	H. Licht
Req-8	"Display of ..."	"The System ..."	H. Miller	J. Locke

Selection: Requirements that have a cross reference to requirement R-11.

Display: "Identifier", "Author", "Source", "Responsible", "Stability"

Figure 8-2 Selective views on the requirements

Visualizaciones consolidadas:

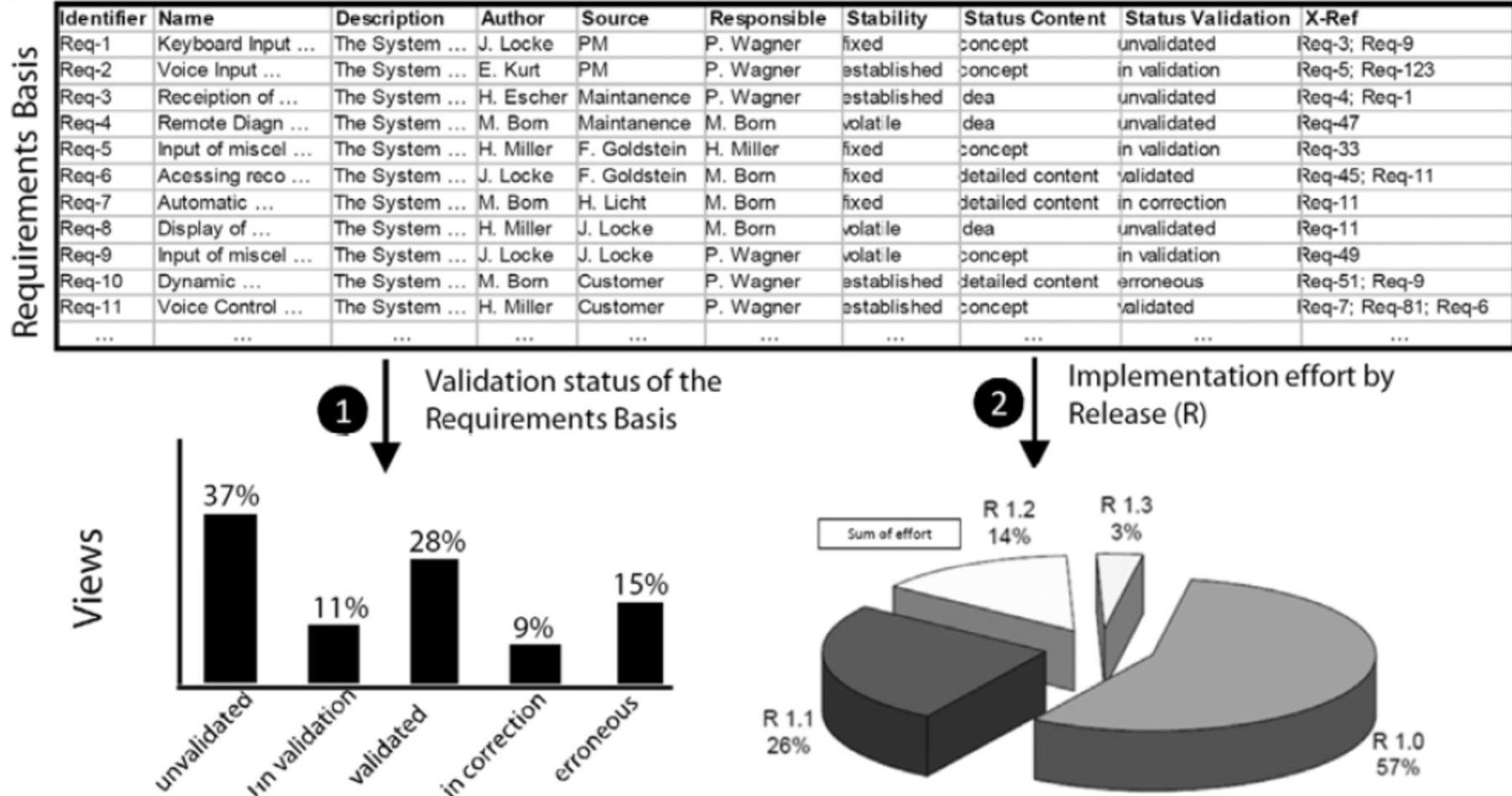


Figure 8-3 Condensed view generated from a requirements basis

Priorización

Todo proyecto que tenga recursos limitados necesita definir prioridades en los requisitos que va a incluir en el producto de software.

Determinar la meta y las restricciones de la priorización;

Determinar los criterios de priorización:

- Costo de implementación,
- Riesgo,
- Costo de una implementación sin éxito,
- Importancia,
- Volatilidad
- Duración de la implementación,
- Precedencia entre requisitos.

Determinar los stakeholders:

Priorización

Técnicas de priorización:

- The spectrum of prioritization techniques spans from simple, single criterion classification to elaborate analytic prioritization approaches, such as [AHP \(Analytical Hierarchy Process\)](#) [Saaty 1980], [Cost-Value-Analysis](#) [Karlsson and Ryan 1997], or [QFD \(Quality Function Deployment\)](#) [Akao 1990].
- **Ranking & Top-Ten:** técnicas consagradas
 - **Ranking:** en esta técnica, los stakeholders seleccionados organizan los requisitos a priorizar con respecto a un criterio específico.
 - **Top-Ten:** en esta técnica, se seleccionan los “n” requisitos más importantes para un criterio definido. Para estos requisitos, un orden de clasificación se determina después. Este orden de “ranking” representa la importancia de los requisitos seleccionados con respecto al criterio definido.

Priorización

Relative weight ①	→ 2 (WeightBenefit)	→ 1 (WeightDetri- ment)			→ 1 (WeightCost)		→ 0.5 (WeightRisk)			
Requirement ②	Relative Benefit	Relative Detriment	Total	Value %	Relative Cost	Cost %	Relative Risk	Risk %	Priority	Rank
R ₁	5	3	13	16.8	2	13,3	1	9,1	0.941	1
R ₂	9	7	25	32.5	5	33,3	3	27,2	0.692	3
R ₃	5	7	17	22.1	3	20,0	2	18,2	0.759	2
R ₄	2	1	5	6.5	1	6,7	1	9,1	0.577	4
↓ R ₅	4	9	17	22.1	4	26,7	4	36,4	0.489	5
Total	25	27	77	100	15	100	11	100	—	
	③	④	⑤	⑥	⑦	⑧	⑨			

Figure 8-4 Calculation of priorities in a prioritization matrix according to Wiegers

$$\text{Value\%}(R_i) = \text{Benefit}(R_i) \times \text{WeightBenefit} + \text{Detriment}(R_i) \times \text{WeightDetriment}$$

$$\text{Priority}(R_i) = \text{Value\%}(R_i) / [\text{Cost\%}(R_i) \times \text{WeightCost} + \text{Risk\%}(R_i) \times \text{WeightRisk}]$$

Trazabilidad

Trazabilidad pre-especificación
Trazabilidad post-especificación

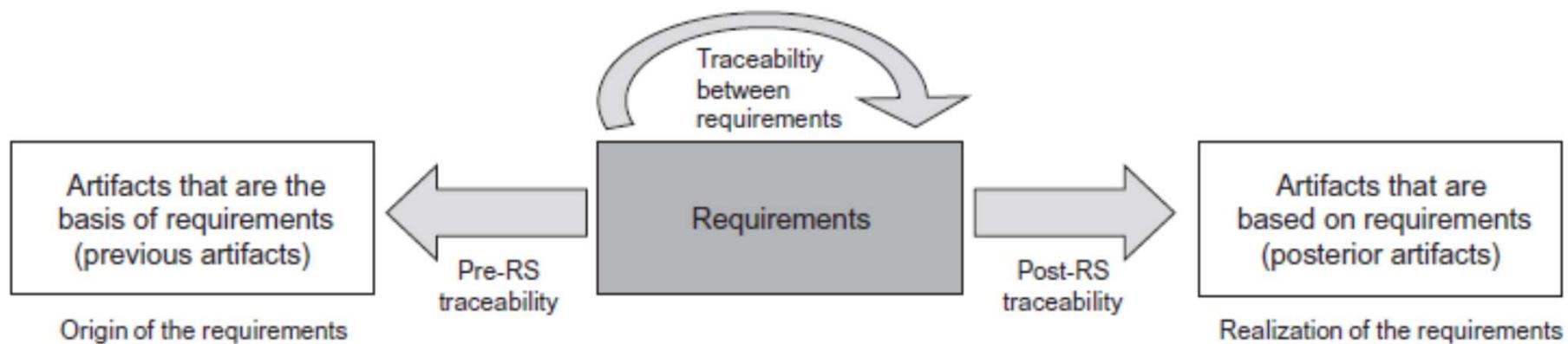


Figure 8-5 Types of requirements traceability

Trazabilidad

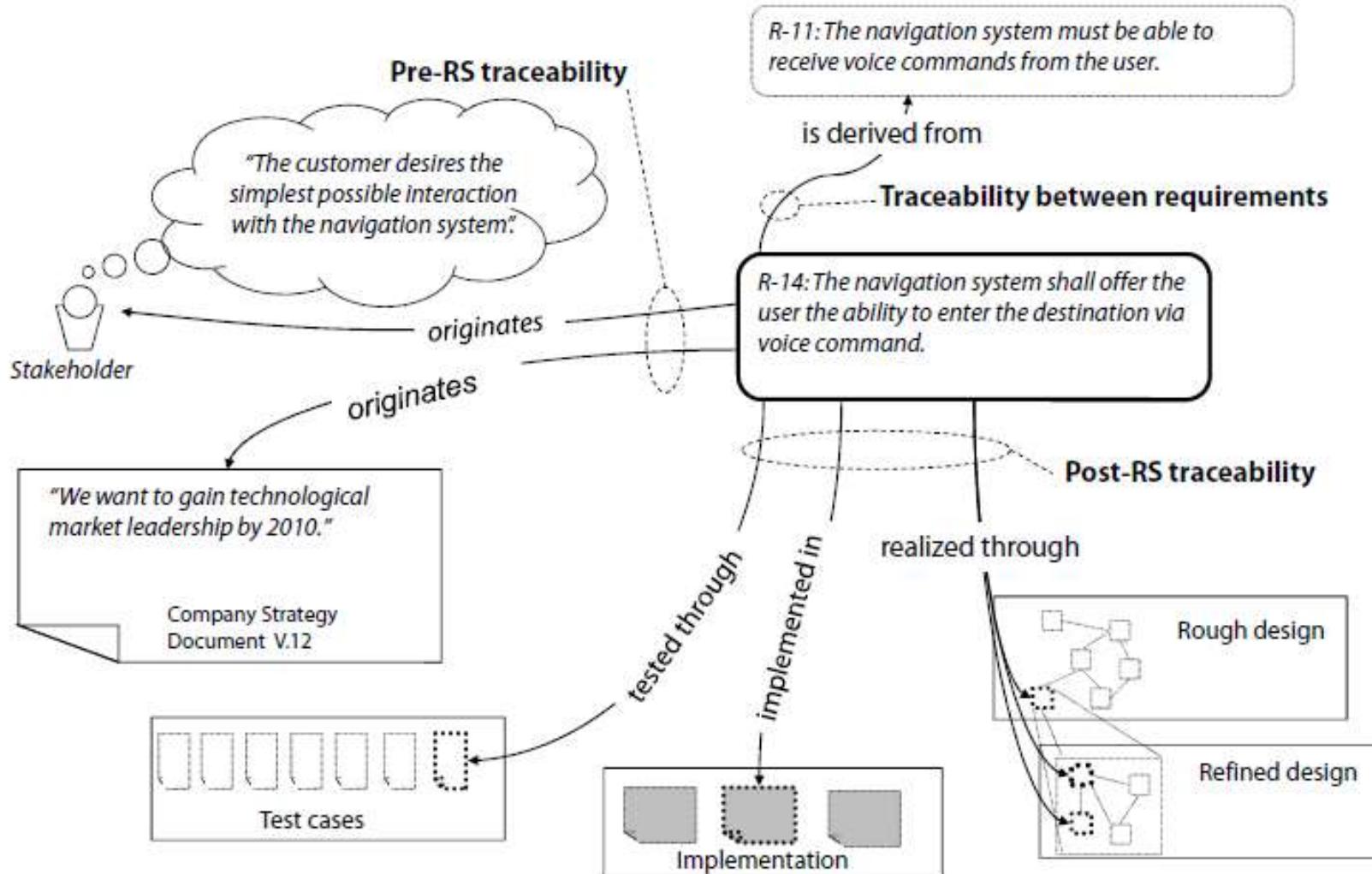


Figure 8-6 Example of the three types of requirements traceability

Trazabilidad

		Target artifacts					
		derived	Req-1	Req-2	Req-3	Req-4	Req-5
Initial artifacts	Req-1		X				
	Req-2			X			
	Req-3						X
	Req-4				X		
	Req-5						

Figure 8-7 Representation of traceability information in a trace matrix

Trazabilidad

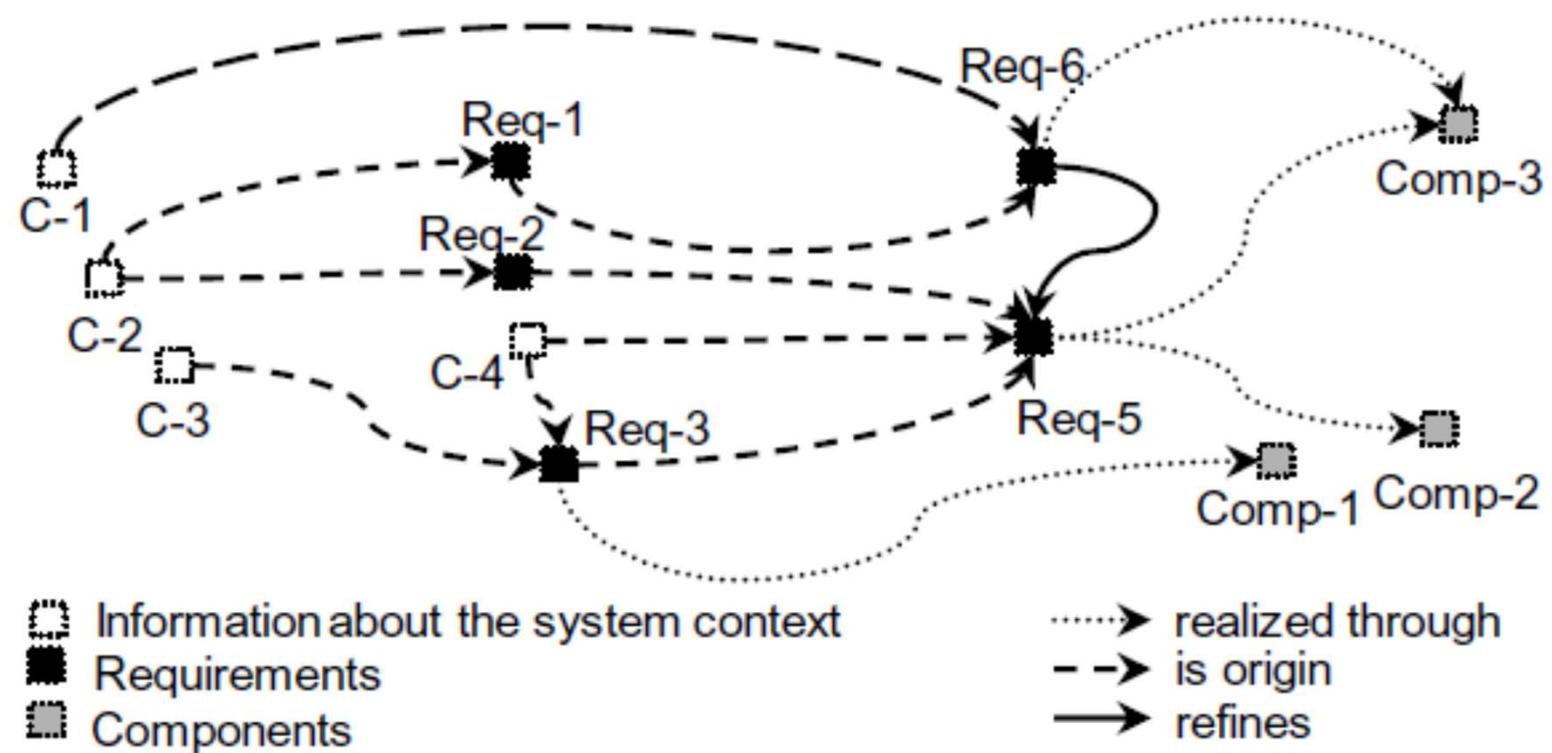


Figure 8-8 Representation of traceability in a trace graph (extract)

Control de Versión

Se puede distinguir la **versión** y el **incremento** de una versión

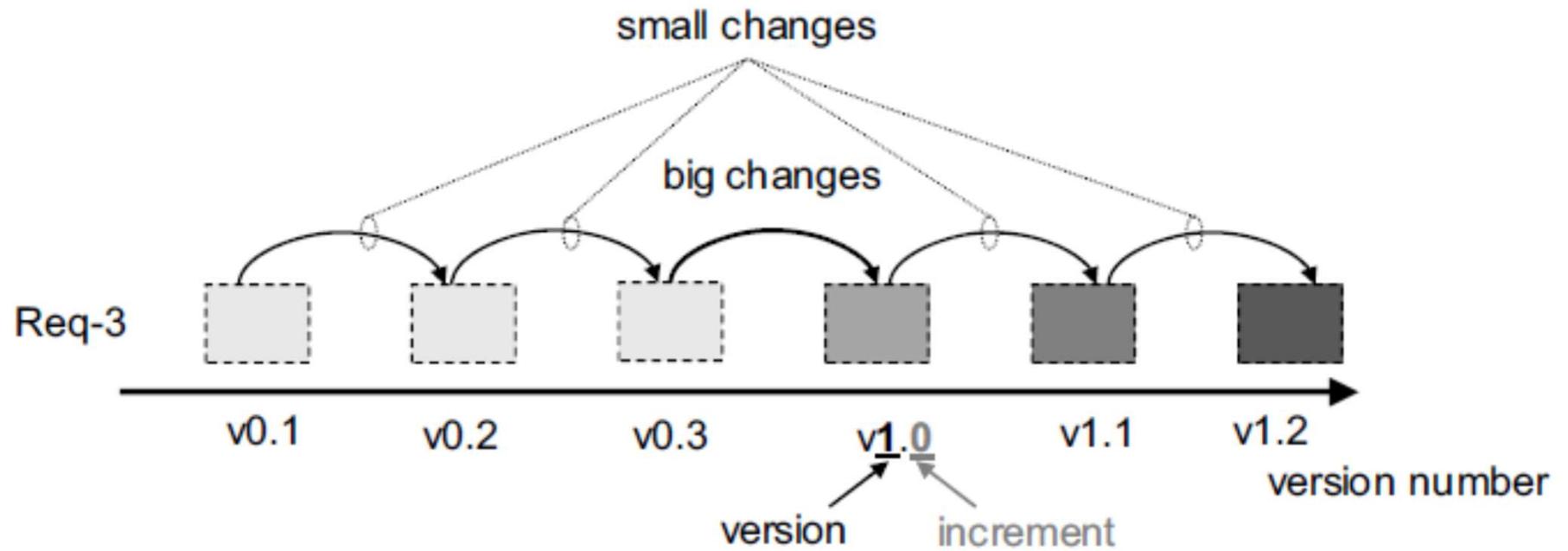


Figure 8-9 Requirements versions

Configuración y Baseline

Configuración: conjunto de requisitos donde cada requisito está presente con una versión

Baseline: configuración con versiones estables de requisitos. Sirve para:

- **Planificación** de entregas (release)
- **Estimación** de esfuerzo de implementación
- **Comparación** con productos de la competencia

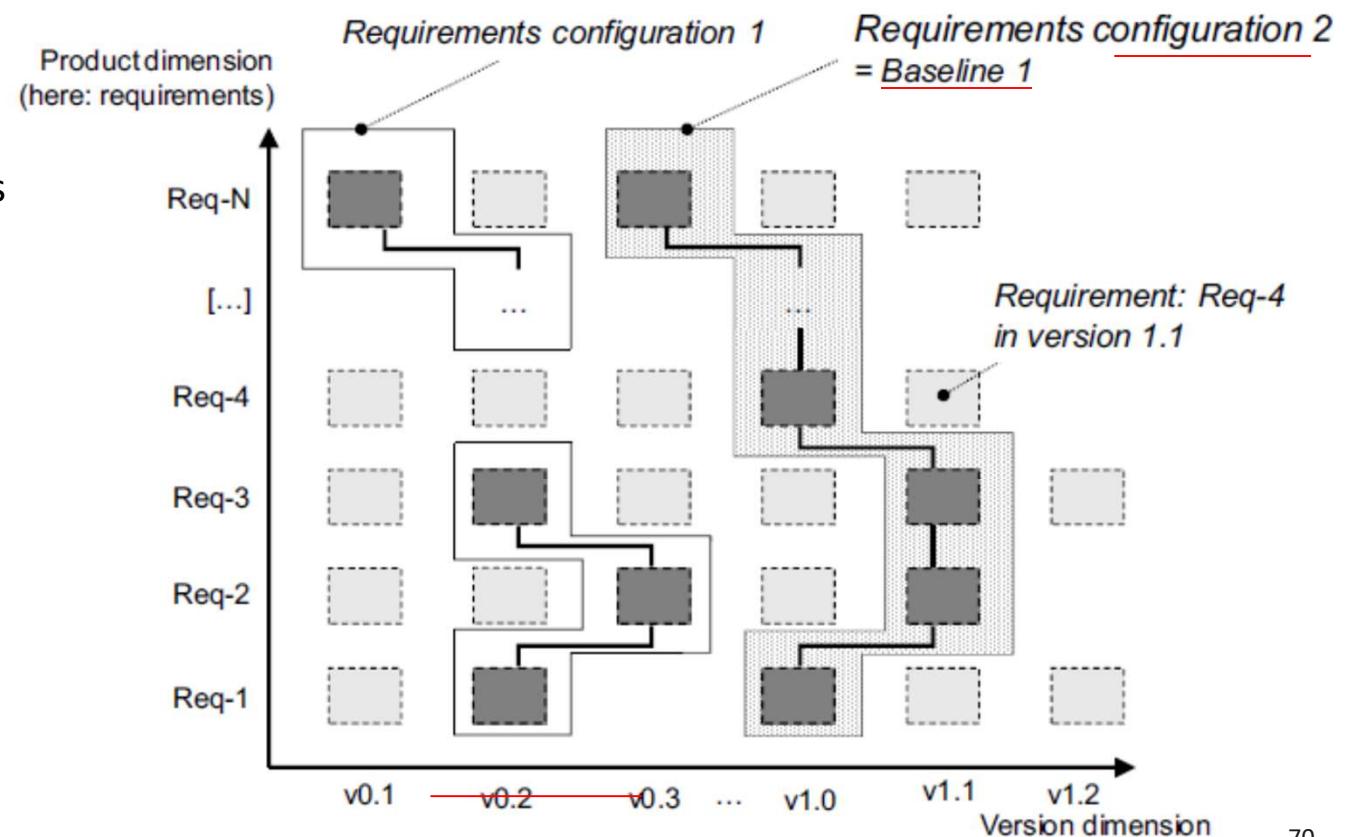


Figure 8-10 Dimensions of configuration management of requirements
(based on [Conradi and Westfechtel 1998])

Control de Cambio

Requisitos cambian a lo largo del ciclo de desarrollo de software.

- El efecto del cambio propuesto se evalúa a través de la información de trazabilidad y el conocimiento general de los requisitos
- Decidir si un cambio en los requisitos debe ser **aceptado o no**.

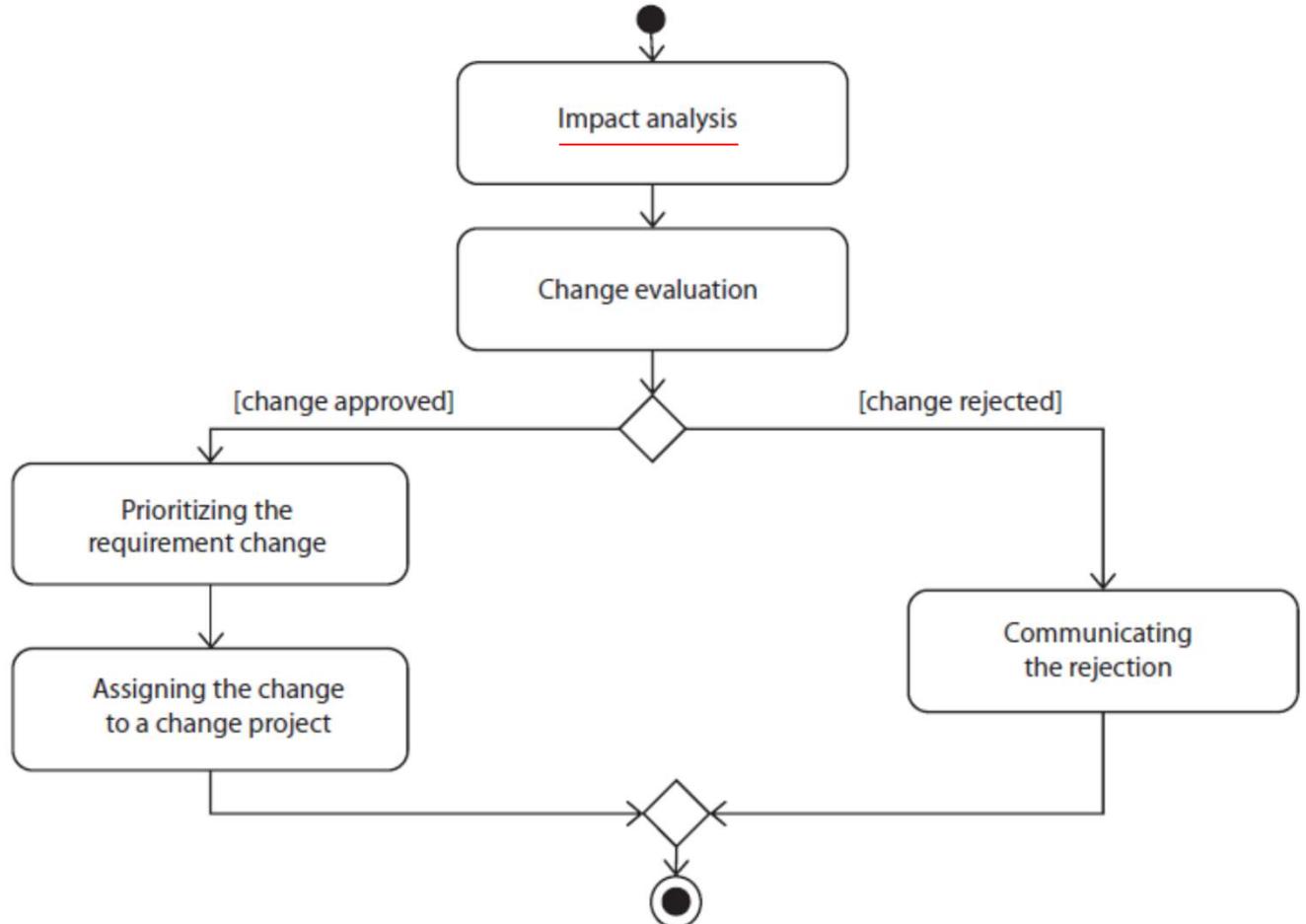


Figure 8-11 Method for handling change requests

DOCUMENTO DE REQUISITOS

2

2. Documento de requisitos

- Como resultado del proceso de elicitación se desarrolla el **documento de especificación de requisitos del software**.
- Este documento incluye **requisitos de usuario** (o requisitos de alto nivel) y **requisitos de sistema** (requisitos detallados) separadamente. Sin embargo, en algunos casos puede contener solo uno o la combinación de ambos.
- Este **documento** contiene la especificación de todos los **requisitos funcionales y no funcionales** del software, **incluidas las capacidades del producto, los recursos disponibles, los beneficios y los criterios de aceptación**.

2. Documento de requisitos

- Errores más comunes cometidos en el desarrollo del documento de requisitos:
 - Omitir un grupo de clientes;
 - Ignorar a un solo cliente;
 - Omitir un grupo de requisitos;
 - Permitir incoherencias entre grupos de requisitos;
 - Aceptar un requisito inadecuado;
 - Aceptar un requisito incorrecto, indefinido, o impreciso;
 - Aceptar un requisito ambiguo e inconsistente;

2. Documento de requisitos

1. El documento de requisitos del sistema debe estar compuesto por **sentencias en lenguaje natural**, siguiendo ciertos **estándares**:
 - a. Iniciar con "El sistema debe ...".
 - b. Utilizar frases cortas. Ejemplo: "El sistema debe girar en microcomputadoras de la línea IBM que tengan microprocesador Pentium III o superior."
2. Registrar los requisitos en términos del cliente.
 1. Desde el punto de vista del usuario.

2. Documento de requisitos

2. Los requisitos deben estar organizados lógicamente y pueden estar **organizados** de diversas formas:
 - a. Requisitos **funcionales** y no funcionales.
 - b. Secuencia de ejecución:
Entrada, Procesamiento, Salida.
 - c. Todas las entradas, todas las salidas, etc.

Muchas veces, se supone que el usuario elabora este documento.

2. Documento de requisitos

3. Cada requisito debe tener un **identificador** único, por ejemplo, un identificador numérico, para su posterior referencia.
4. Los requisitos del software deben estar divididos en **requisitos funcionales y no funcionales** (de calidad).
5. Los requisitos **no deben contener detalles de implementación**, lo que no es conveniente en esta fase de desarrollo. Es importante no utilizar términos relacionados con la implementación, como "archivo" y "menú".

2. Documento de requisitos

6. Explicación de los términos del dominio de la aplicación no deben estar presentes en los requisitos, debiendo aparecer en un **vocabulario del dominio de la aplicación (Léxico, Glosario)**.
7. Mantener un **uso consistente** de los términos del dominio de aplicación.

Contenido del Documento de Requisitos

- Segun la norma **IEEE 830 - 1998**, un documento de requisitos debe estar organizado de la siguiente forma:
 - 1 [Introducción](#)
 - [1.1 Propósito](#)
 - [1.2 Ámbito de Sistema](#)
 - [1.3 Definiciones](#)
 - [1.4 Referencias](#)
 - [1.5 Visión general del documento](#)
 - 2 [Descripción General](#)
 - [2.1 Perspectiva del Producto](#)
 - [2.2 Funciones del Producto](#)
 - [2.3 Características de los Usuarios](#)
 - [2.4 Restricciones](#)
 - [2.5 Suposiciones y Dependencias](#)
 - [2.6 Requisitos Futuros](#)
 - 3 [Requisitos Específicos](#)
 - [3.1 Interfaces Externas](#)
 - [3.2 Funciones](#)
 - [3.3 Requisitos de Rendimiento](#)
 - [3.4 Restricciones de Diseño](#)
 - [3.5 Atributos del Sistema](#)
 - [3.6 Otros Requisitos](#)
 - 4 [Apéndices](#)

Ver detalles en:

Especificación de Requisitos Software según el estándar IEEE 830.

https://wikis.fdi.ucm.es/ELP/Especificaci%C3%B3n_de_Requisitos_Software_seg%C3%BAn_el_est%C3%A1ndar_IEEE_830

<https://www.fdi.ucm.es/profesor/gmendez/docs/is0809/ieee830.pdf>

Ejemplo en:

<https://sistemasifescol.files.wordpress.com/2014/08/ejemplo-formato-ieee-830.doc>

Contenido del Documento de Requisitos

- Segun la norma **IEEE 29148:2011**, un documento de requisitos debe estar organizado de la siguiente forma:

1. Introduction
1.1 Purpose
1.2 Scope
1.3 Product overview
1.3.1 Product perspective
1.3.2 Product functions
1.3.3 User characteristics
1.3.4 Limitations
1.4 Definitions
2. References
3. Specific requirements
3.1 External interfaces
3.2 Functions
3.3 Usability Requirements
3.4 Performance requirements
3.5 Logical database requirements
3.6 Design constraints
3.7 Software system attributes
3.8 Supporting information
4. Verification
(parallel to subsections in Section 3)
5. Appendices
5.1 Assumptions and dependencies
5.2 Acronyms and abbreviations

Ejemplo en:

https://issuu.com/ivanjf/docs/srs_iso-iec-ieee_29148-2011

http://pegasus.javeriana.edu.co/~CIS1430IS11/documents/Requerimientos/SRS_MV%20LIFE%20Gym%20Mobile.pdf

Uso del Documento de Requisitos

- El documento de requisitos sirve como base para:
 - **Planning:** Based on the requirements document, concrete work packages and milestones for the implementation of the system can be defined.
 - **Architectural design:** The detailed documented requirements (along with constraints) serve as the basis for the design of the system architecture.
 - **Implementation:** Based on the architectural design, the system is implemented by making use of the requirements.
 - **Test:** On the basis of requirements that have been documented in the requirements document, test cases can be developed that can be used for system validation later on.
 - **Change management:** When requirements change, the requirements document can serve as the basis to analyze the extent to which other parts of the system are influenced. The change effort can thus be estimated.
 - **System usage and system maintenance:** After the system is developed, the requirements document is used for maintenance and support. This way, the requirements document can be used to analyze concrete defects and shortcomings that surface during system use. For example, one can deduct if a defect is a result of using the system incorrectly, a result of an error in requirements, or a result of an error in implementation.
 - **Contract management:** The requirements document is the prime subject of a contract between a client and a contractor in many cases.

Características de Calidad

- La norma **IEEE 29148:2011** define las siguientes Características de una buena
- **Especificación**

- Unambiguity and consistency
- Clear structure
- Modifiability and extendibility
- Completeness
- Traceability

● **Requisitos**

- Agreed:
- Unambiguous:
- Necessary:
- Consistent:
- Verifiable:
- Feasible:
- Traceable:
- Complete:
- Understandable:

Along with quality criteria for requirements, there are two fundamental rules that enhance the **readability** of requirements:

- *Short sentences and short paragraphs:* As human short-term memory is very limited, circumstances that belong together should be described in no more than seven sentences.
- *Formulate only one requirement per sentence:* Formulate requirements using active voice and use only one process verb. Long, complicated interlaced sentences must be avoided.

IREB, 2015

HERRAMIENTAS

3

Herramientas

- Las herramientas de Gestión de Requisitos se caracterizan por tener las siguientes propiedades:
 - Gestión de requisitos y atributos basados en los modelos de información
 - Organización de requisitos
 - Configuración y gestión de versión en los requisitos
 - Definición de baseline de los requisitos
 - Acceso y gestión multiusuario
 - Gestión de la trazabilidad
 - Consolidación de los requisitos obtenidos
 - Gestión de cambios
 - Análisis de impacto

Herramientas

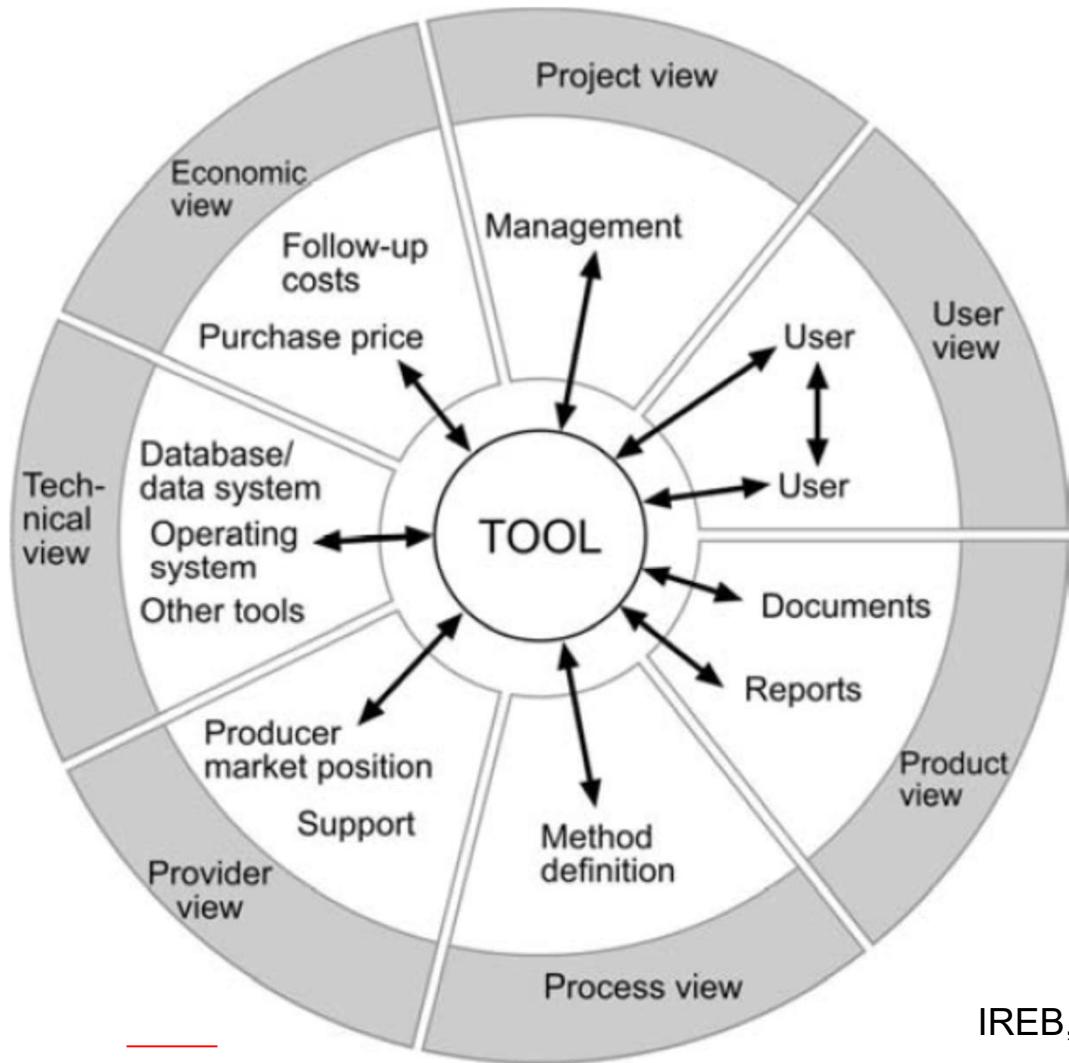
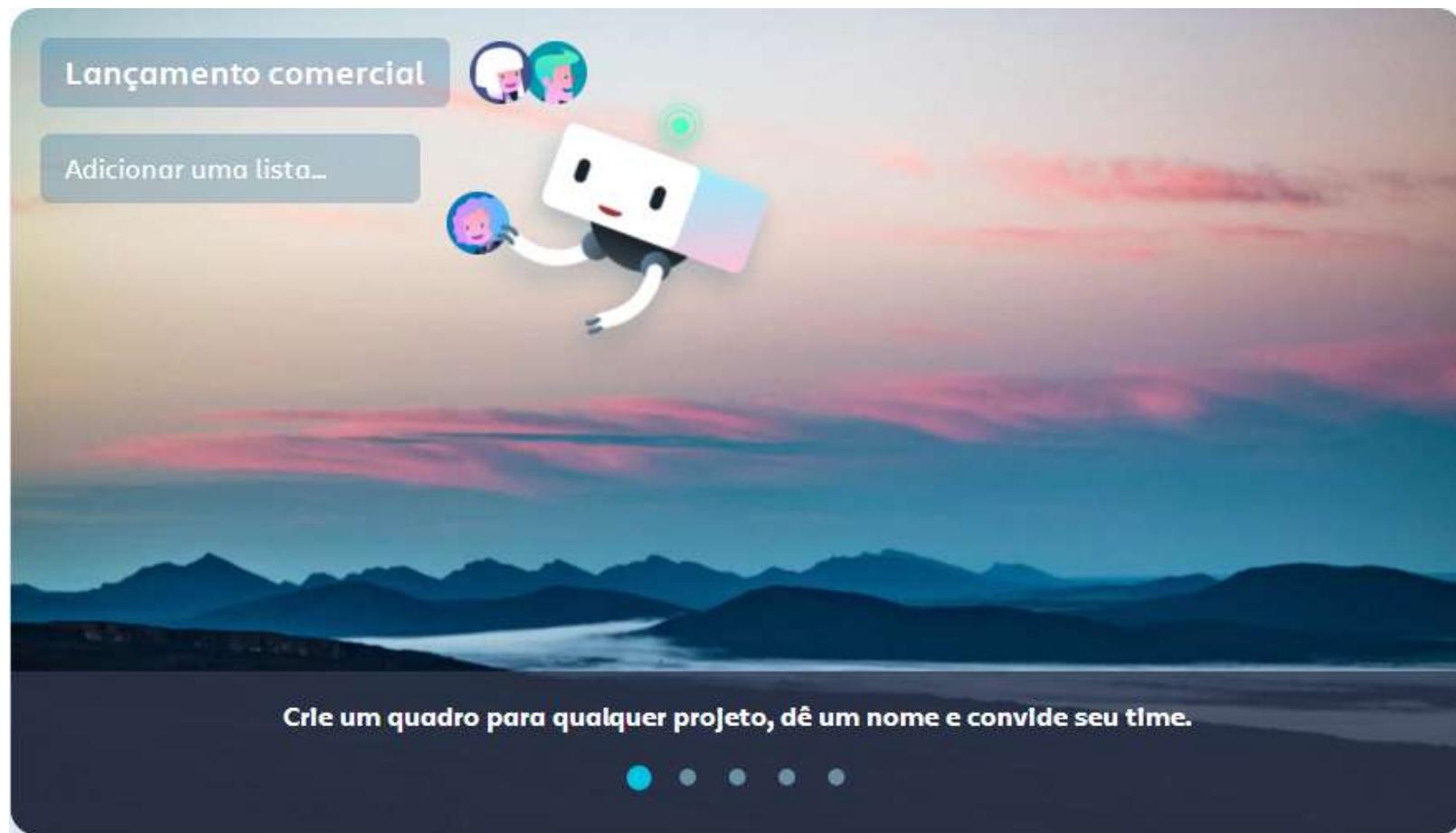


Figure 9-1 Views on a requirements engineering tool

Herramientas

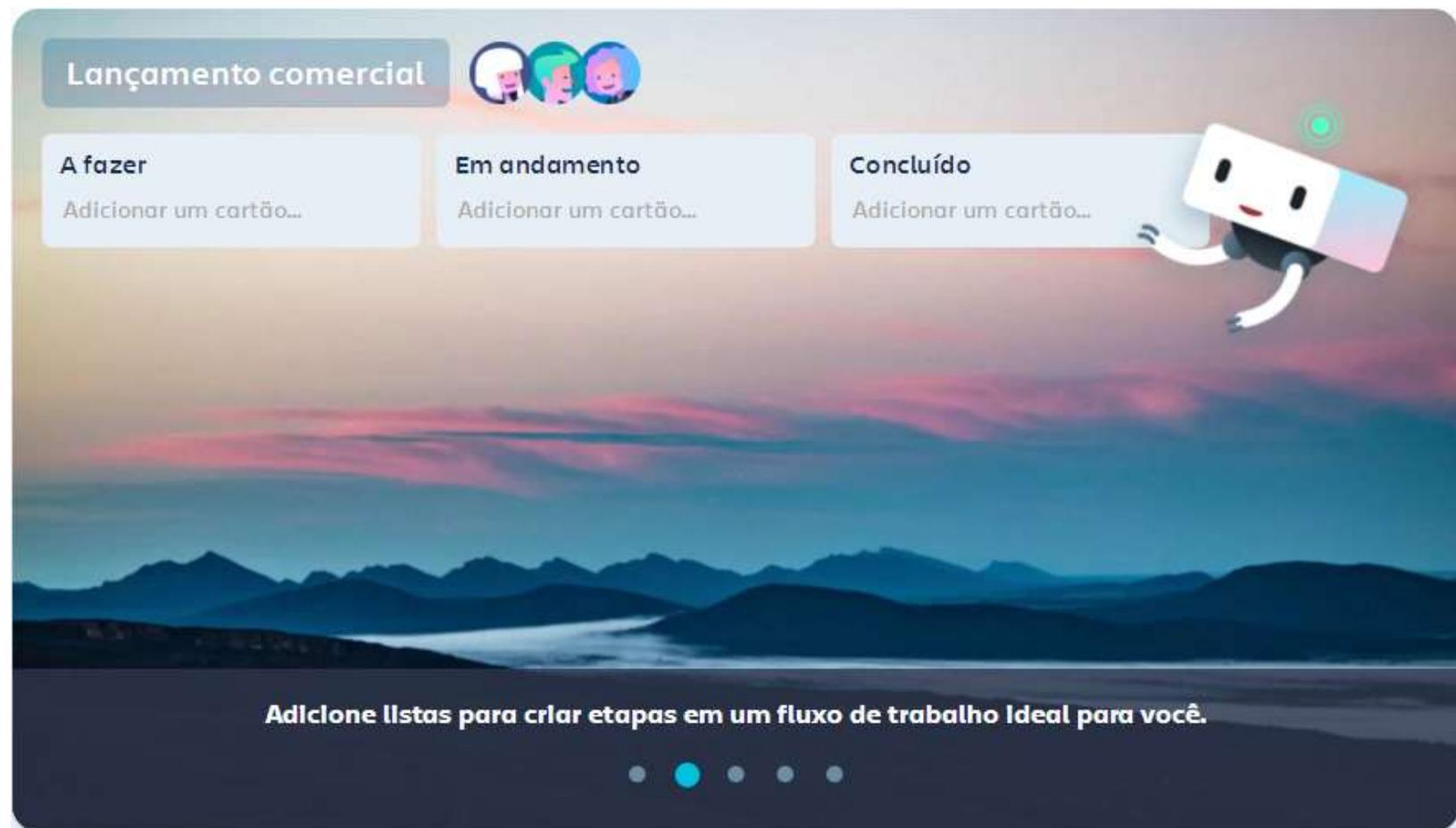


Trello



<https://trello.com>

Trello



<https://trello.com>

Trello

Lançamento comercial

A fazer

- Contratar contador
- Solicitar um empréstimo
- Adicionar um cartão...

Em andamento

- Criar página no Facebook
- Criar site
- Criar logotipo
- Assinar aluguel de escritório
- Adicionar um cartão...

Concluído

- Abrir conta bancária
- Elaborar plano de negócios
- Adicionar um cartão...

Cré cartões para tarefas a serem concluídas ou informações que deseja organizar.

<https://trello.com>

Trello

The screenshot shows a Trello card titled "Criar site". The card has the following details:

- Membros:** Two user icons (one pink, one teal).
- Data de Entrega:** A green button with a checkmark containing the text "24 de julho".
- Descrição:** "Crie um site moderno para nosso novo negócio."
- Checklist:** A progress bar at 100% concluído. The checklist items are:
 - Encontre um modelo do WordPress que atenda às suas necessidades
 - @stevethedev para configurar e personalizar o modelo
 - Publicar nosso site!
- Bottom text:** "Clique em um cartão para adicionar dados, datas de entrega, checklists, comentários e outros." and "Adicionar Comentário".

<https://trello.com>

Trello

Lançamento comercial

A fazer

- Contratar contador
- Solicitar um empréstimo
- Adicionar um cartão...

Em andamento

- Criar página no Facebook
- Criar logotipo
- Assinar aluguel de escritório
- Adicionar um cartão...

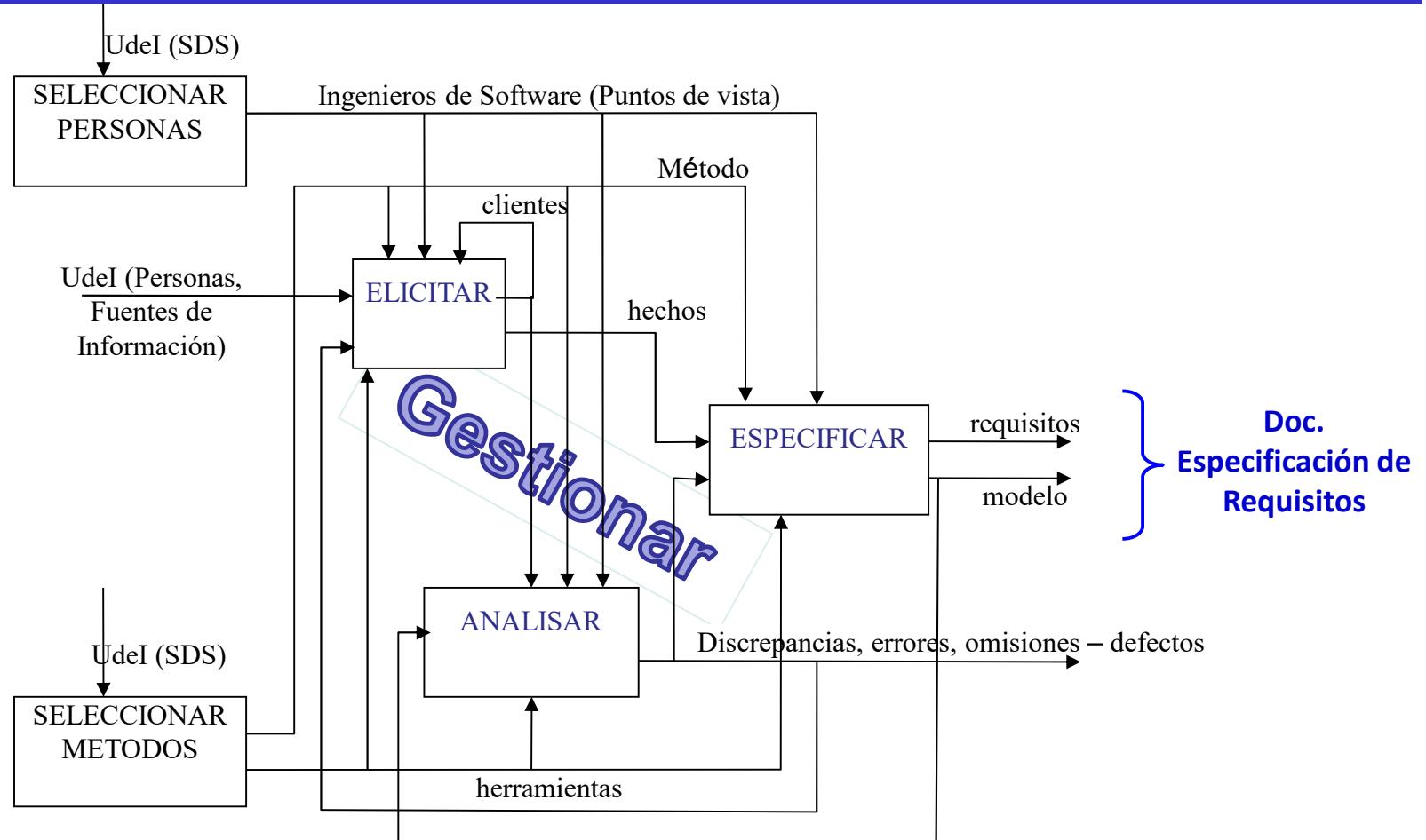
Concluído

- Criar site
- Abrir conta bancária
- Elaborar plano de negócios
- Adicionar um cartão...

Mova-os pelas listas para indicar progresso. Sólido "A fazer" para o "Concluído" em um piscar de olhos!

<https://trello.com>

Conclusión



Referencias

- **Basado en:**

- Leite, J.C.S.P. 2007. Livro Vivo : Engenharia de Requisitos, <http://livrodeengenhariaderequisitos.blogspot.com/>
- Rosana T. Vaccare Braga. 2017. Requisitos de Software. https://edisciplinas.usp.br/pluginfile.php/3142953/mod_resource/content/2/Aula09-Requisitos.pdf
- Elisa Yumi Nakagawa. ENGENHARIA DE REQUISITOS. https://edisciplinas.usp.br/pluginfile.php/58062/mod_resource/content/1/Aula08_Engenharia_Requisitos.pdf
- Pohl, K. and Rupp, C. 2015. Requirements Engineering Fundamentals. IREB
- DO PRADO LEITE, Julio Cesar Sampaio et al. A scenario construction process. **Requirements Engineering**, v. 5, n. 1, p. 38-61, 2000.
- MAVIN, Alistair et al. Easy approach to requirements syntax (EARS). In: **2009 17th IEEE International Requirements Engineering Conference**. IEEE, 2009. p. 317-322.
- Un ejemplo de casos de uso. Sokoban http://www.lsi.us.es/~javierj/cursos_ficheros/03.%20Sokoban.%20Un%20ejemplo%20de%20plantillas.pdf