Ingeniería de Requisitos: Elicitación y Especificación

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

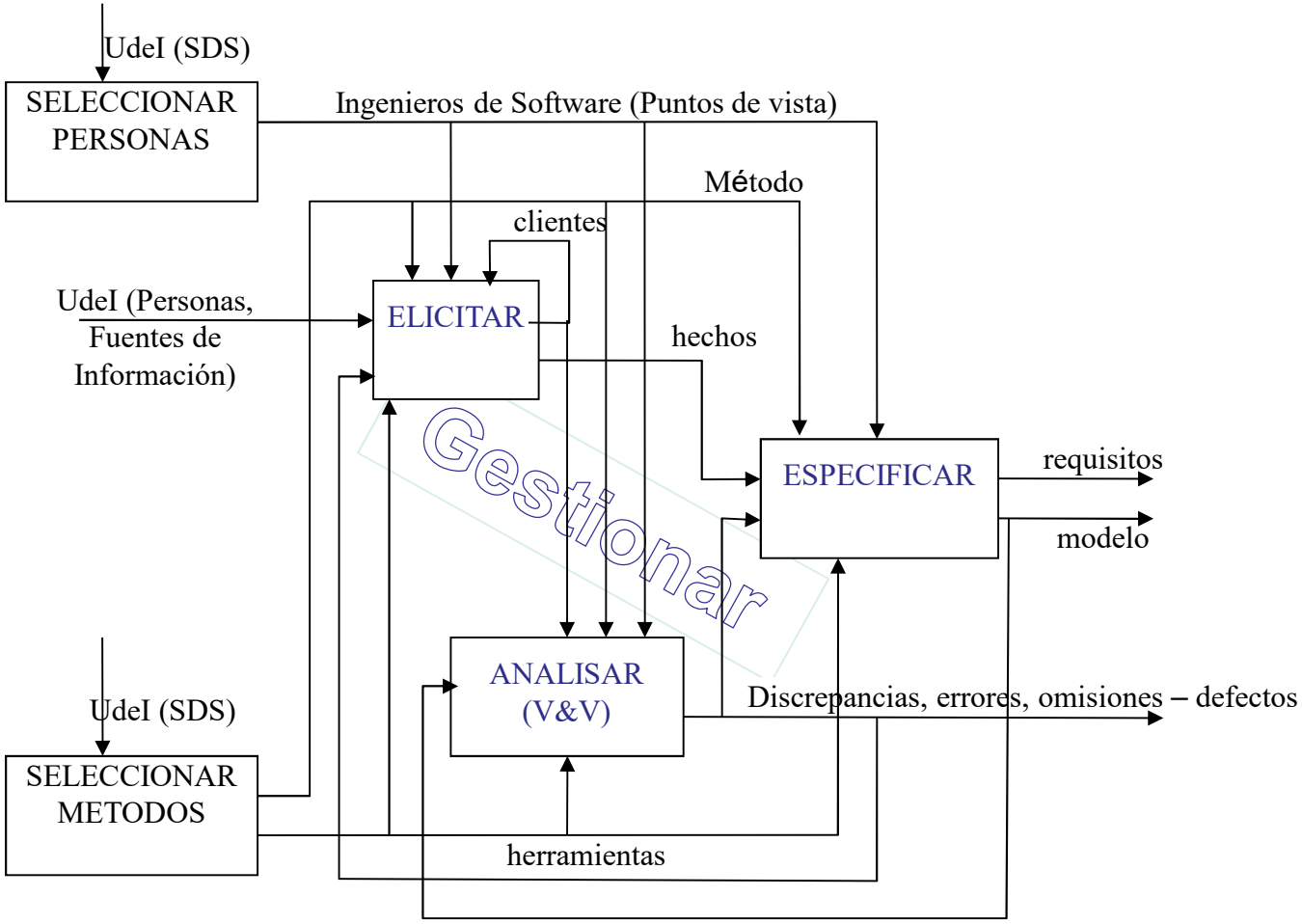
Los servicios prestados por el sistema, sus cualidades específicas y sus restricciones operativas.

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

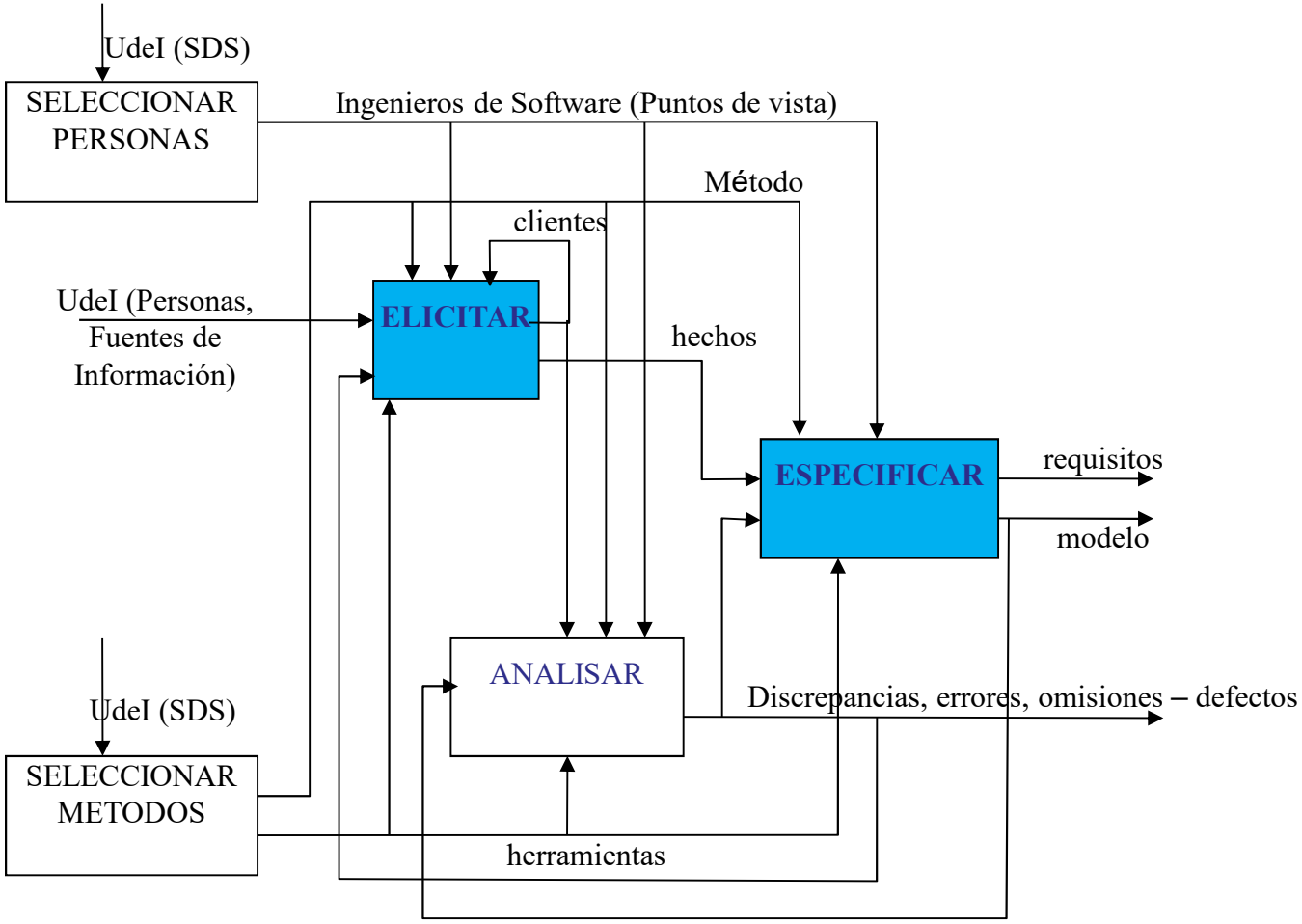
La I.R. establece el proceso de definición de Requisitos como un proceso en el cual lo que debe hacer es elicitar, modelar y analizar. Este proceso debe manejar diferentes puntos de vista, y utilizar una combinación de métodos, herramientas y personal. El producto de este proceso es un modelo, del que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

(Júlio Leite, 1994)

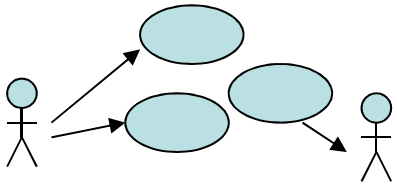
Universo de Información

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



Principales Actividades de la Ingeniería de Requisitos



- Cliente realiza transacciones
  - Cliente deposita dinero
  - Cliente retira dinero
    - Saldo suficiente
    - Saldo insuficiente

Elicitación

Ejemplos/
Escenarios/

Historias

Especificación

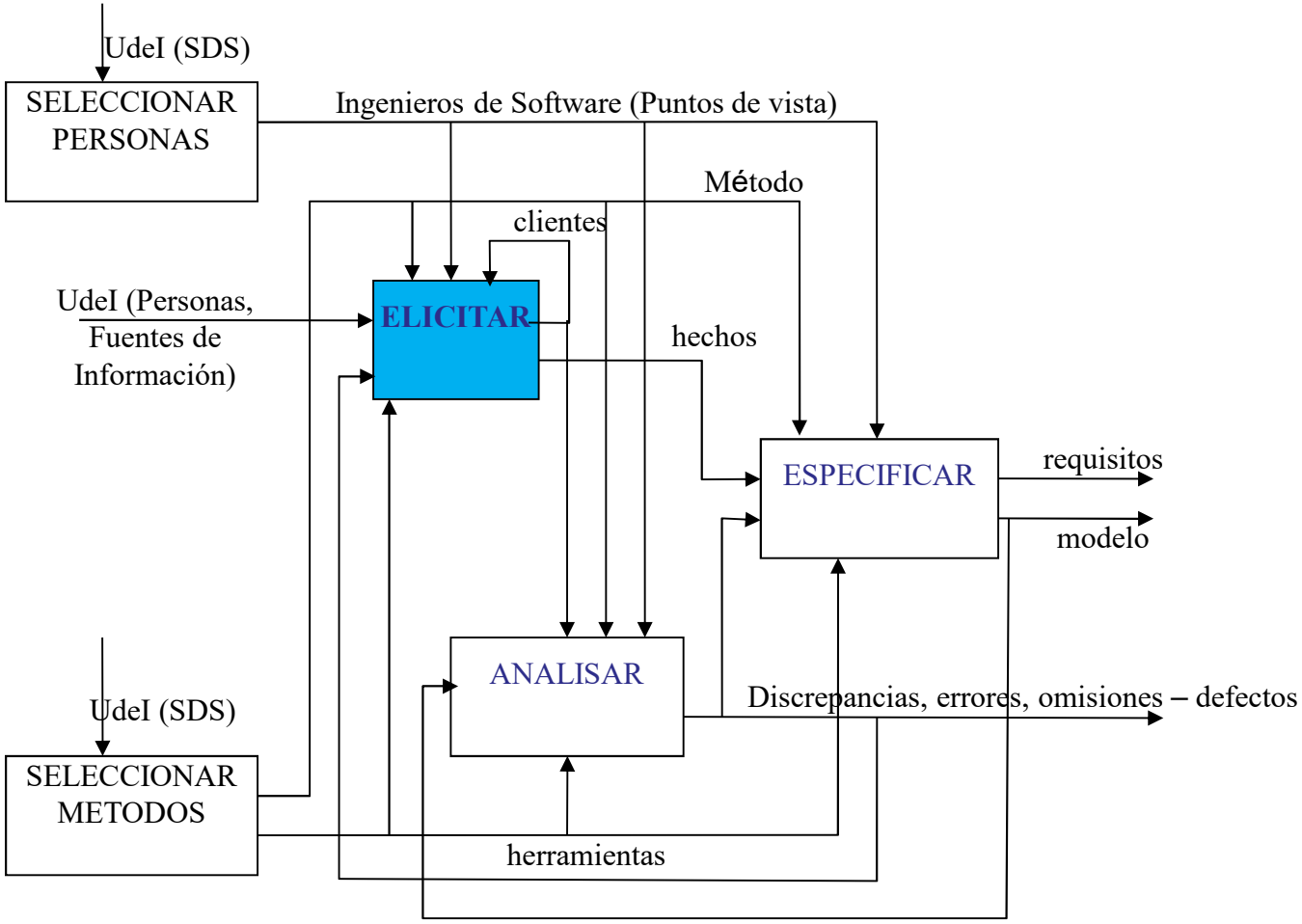
Historias/

Descripciones de
Casos de Uso/
Especificación BDD

Retirar Dinero	
Actores	Cliente
Flujo Principal	1. 2.
Flujo Alternativo	2.1 Saldo insuficiente

Retirar Dinero	
Escenario	Saldo suficiente
Dado Cuando Entonces	
Escenario	Saldo insuficiente
Dado Cuando Entonces	

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 CORRECCIÓN 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
  - Otros softwares de sistemas similares
  - Google....

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 UdeI
- Reuniones
- Brainstorming
- Reutilización
- joint application design
- Etnografía
- 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 imagem 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 UdeI.
- 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 UdeI. Estos tipos son: Sujeto, Objeto, Verbo y Estado.

Léxico Extendido del Lenguaje

Objetivo	Consecuencia
CONOCER EL VOCABULARIO DEL USUARIO	• 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	• 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:

…

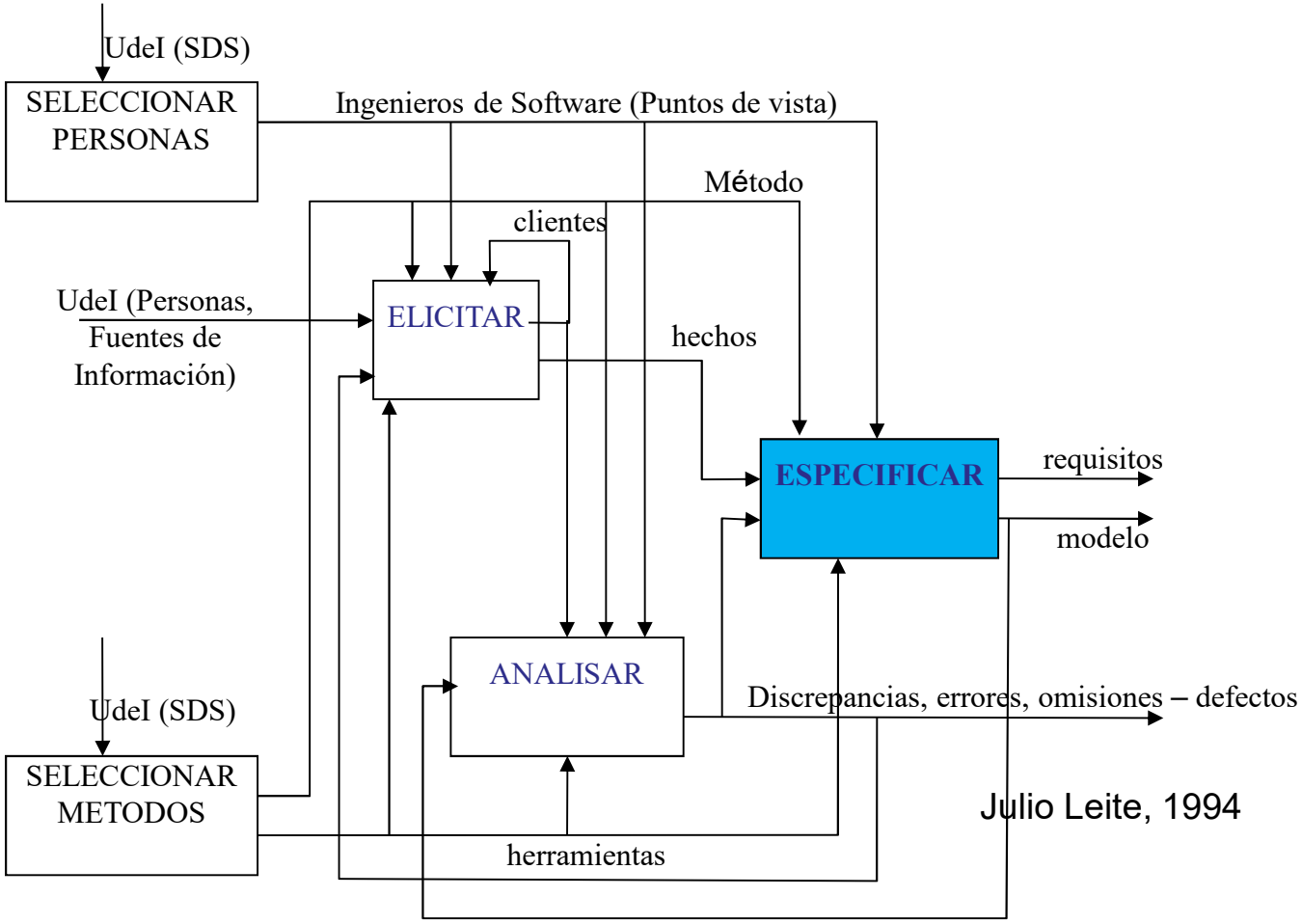
Léxico Extendido del Lenguaje

- 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



Julio Leite, 1994

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

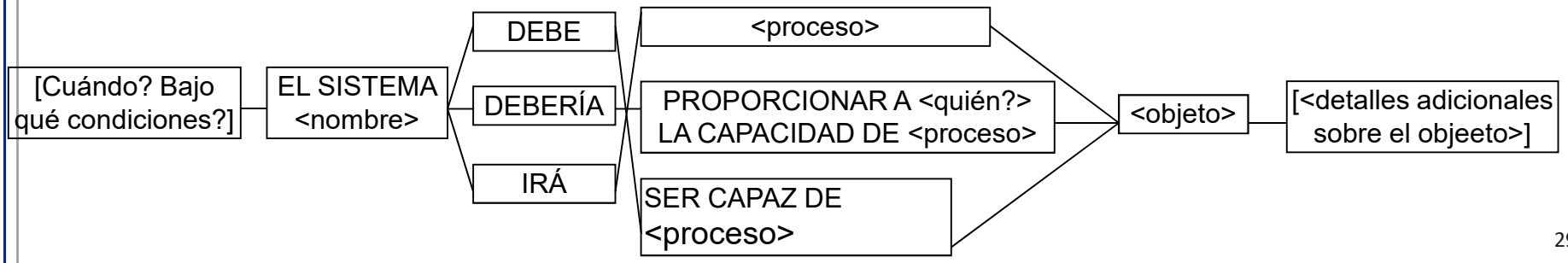
- Una vez que los requisitos son elicitados necesitan ser descritos en un documento de Especificación de Requisitos de Software: *Requisitos de Usuario o Requisitos de Sistema*
- **Métodos / Técnicas :**
  - Especificación Usando Lenguaje Natural
  - Especificación Usando Modelos

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

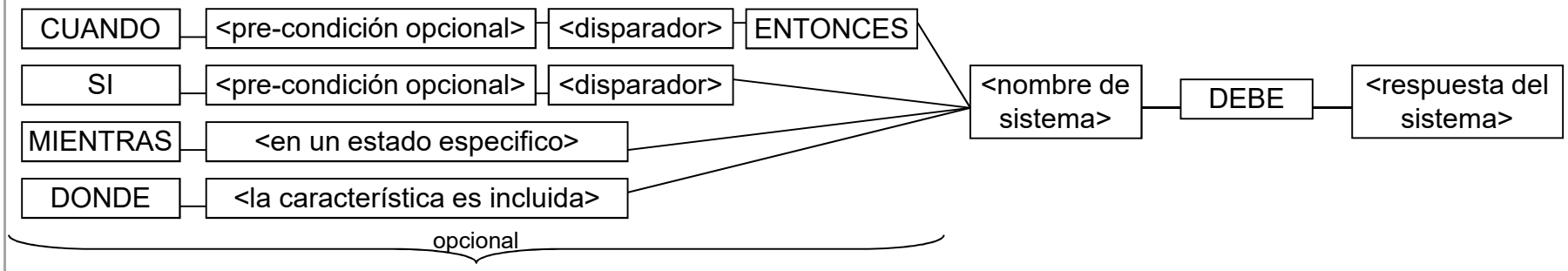
- **Rupp's template**:

Tipo	Template/Ejemplo
Requisito Autonomo	**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.
Requisito de Interacción del Usuario	**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.
Requisito de Interface	**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.

30

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, desvios). 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.

32

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

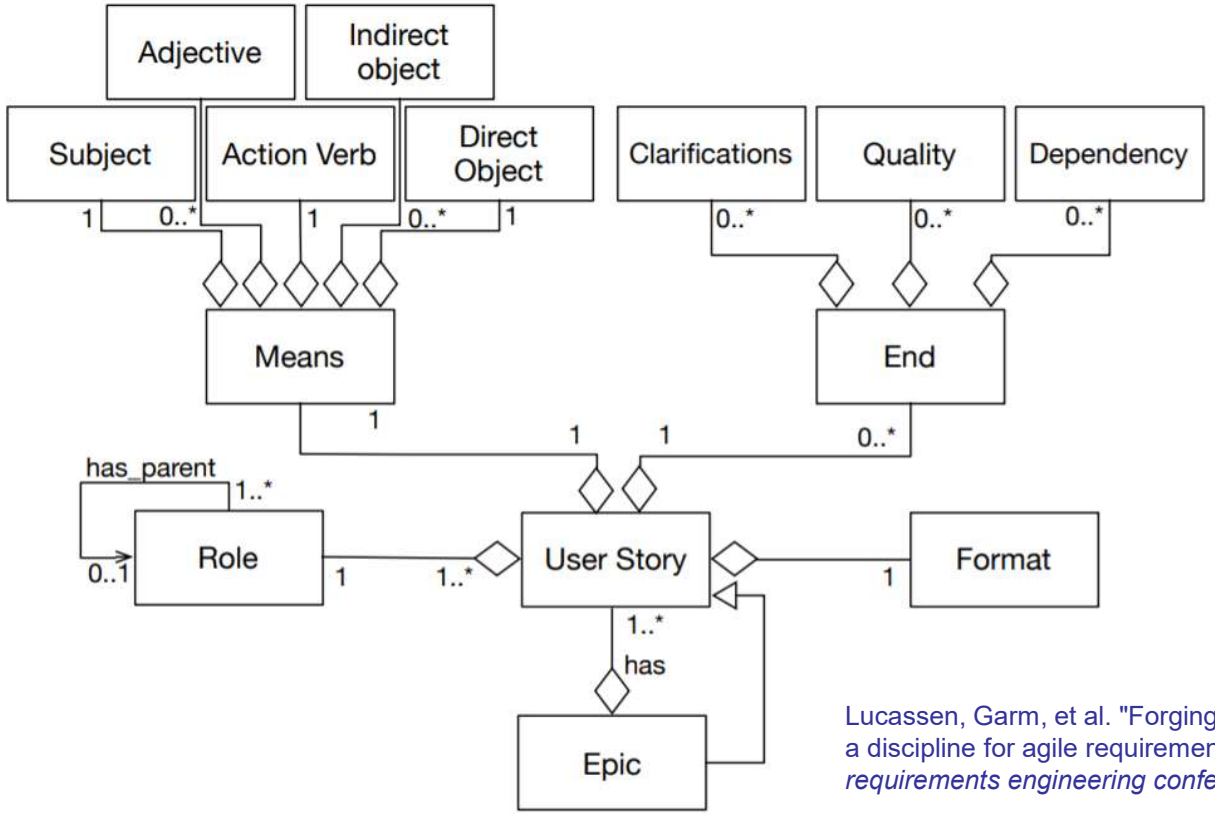
Especificación Usando Lenguaje Natural

- **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.



Lucassen, Garm, et al. "Forging high-quality user stories: towards a discipline for agile requirements." *2015 IEEE 23rd international requirements engineering conference (RE)*. IEEE, 2015.

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

p.e. para diseñar software con cierta autonomía, requisitos no funcionales

- **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

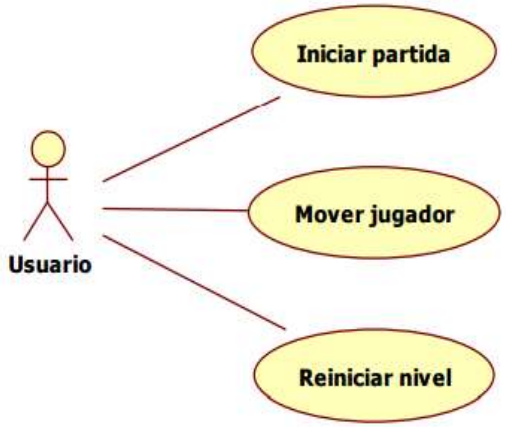
Casos de Uso

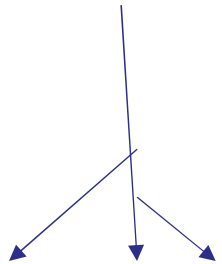
Casos de Uso

- **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 juga
- Para mover una caja, el jug
  empujarla. Si la casilla haci
  está libre la caja se moverá
- Si el jugador se queda bloq
  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.

http://www.lsi.us.es/~javierj/cursos_ficheros/03.%20Sokoban.%20Un%20ejemplo%20de%20plantillas.pdf

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** son: "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.

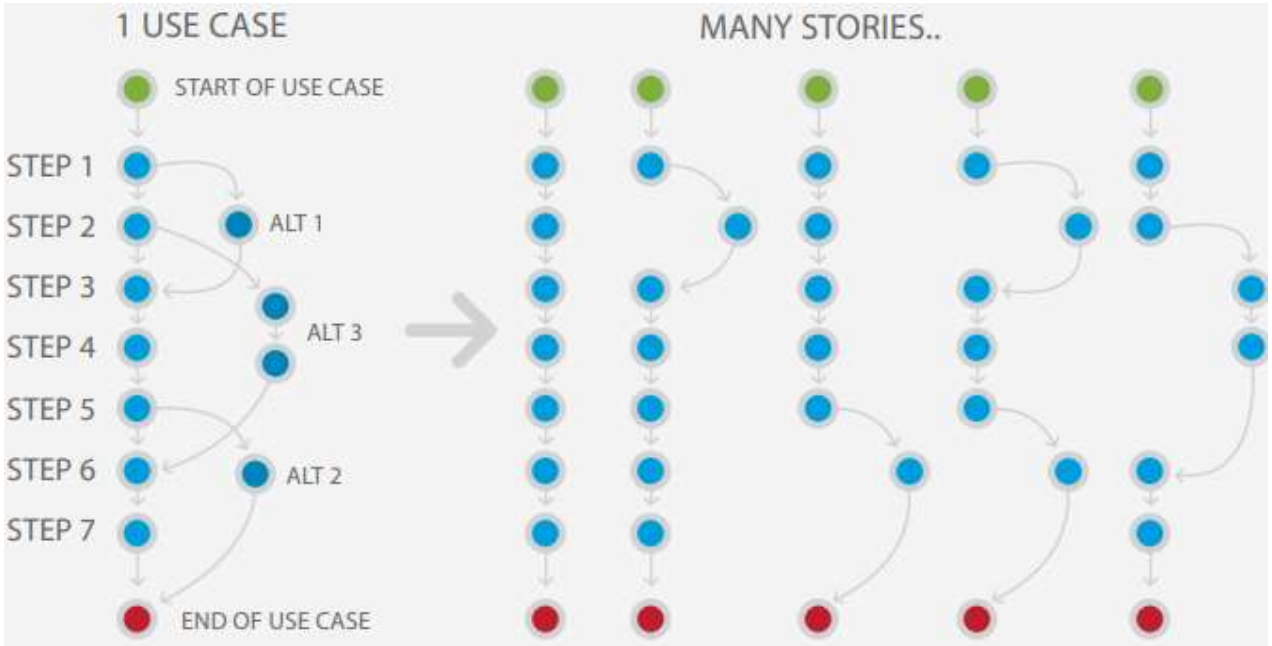
Descripción de Casos de Uso

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	El flujo de eventos más común y exitoso. Acciones o eventos - se numeran secuencialmente.		
	<Paso>	<Acción o nombre de otro caso de uso>	
	…	…	
Alternativas/Excepciones	Describe una desviación en otro flujo de eventos. Un flujo alternativo siempre depende de una condición que ocurra en un paso específico en un flujo de referencia. Un flujo alternativo se compone de una secuencia de pasos numerados.		
	<Paso> <Referencia>	<Condición que causa la ramificación>	
		<Paso><Referencia><Ref>	<Acción o nombre de otro caso de uso>
		...	

Descripción de Casos de Uso

- **Descripción de Caso de Uso**: Un conjunto ordenado de interacciones entre un sistema y un conjunto de actores externos al sistema
- **Escenario de Uso (Historia o Slice)**: Descripción parcial del comportamiento de una aplicación que ocurre en un momento dado y un contexto determinado – una **situación**

BASIC FLOW	ALTERNATIVE FLOWS
1. Insert Card	A1 Invalid Card
2. Validate Card	A2 Non-Standard Amount
3. Select Cash Withdrawal	A3 Receipt Required
4. Select Account	A4 Insufficient Funds in ATM
5. Confirm Availability of Funds	A5 Insufficient Funds in Acct
	A6 Would Cause Overdraft
6. Return Card	A7 Card Stuck
7. Dispense Cash	A8 Cash Left Behind
	etc..

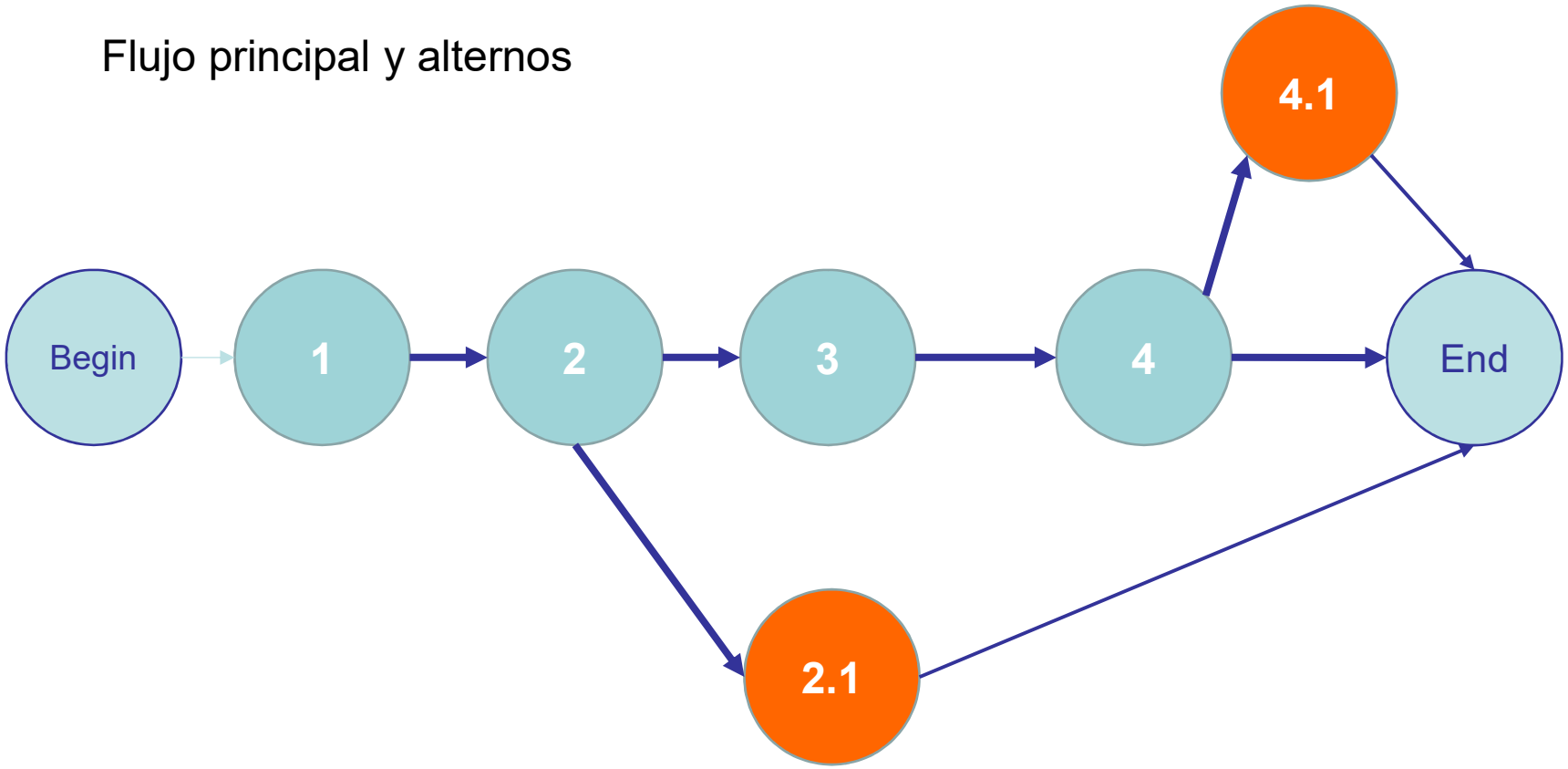


Descripción de Casos de Uso

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	**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 cajas **4** El sistema comprueba si el usuario ha completado el nivel, muestra la pantalla de juego y espera un nuevo movimiento
Alternativas	**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

Flujo principal y alternos

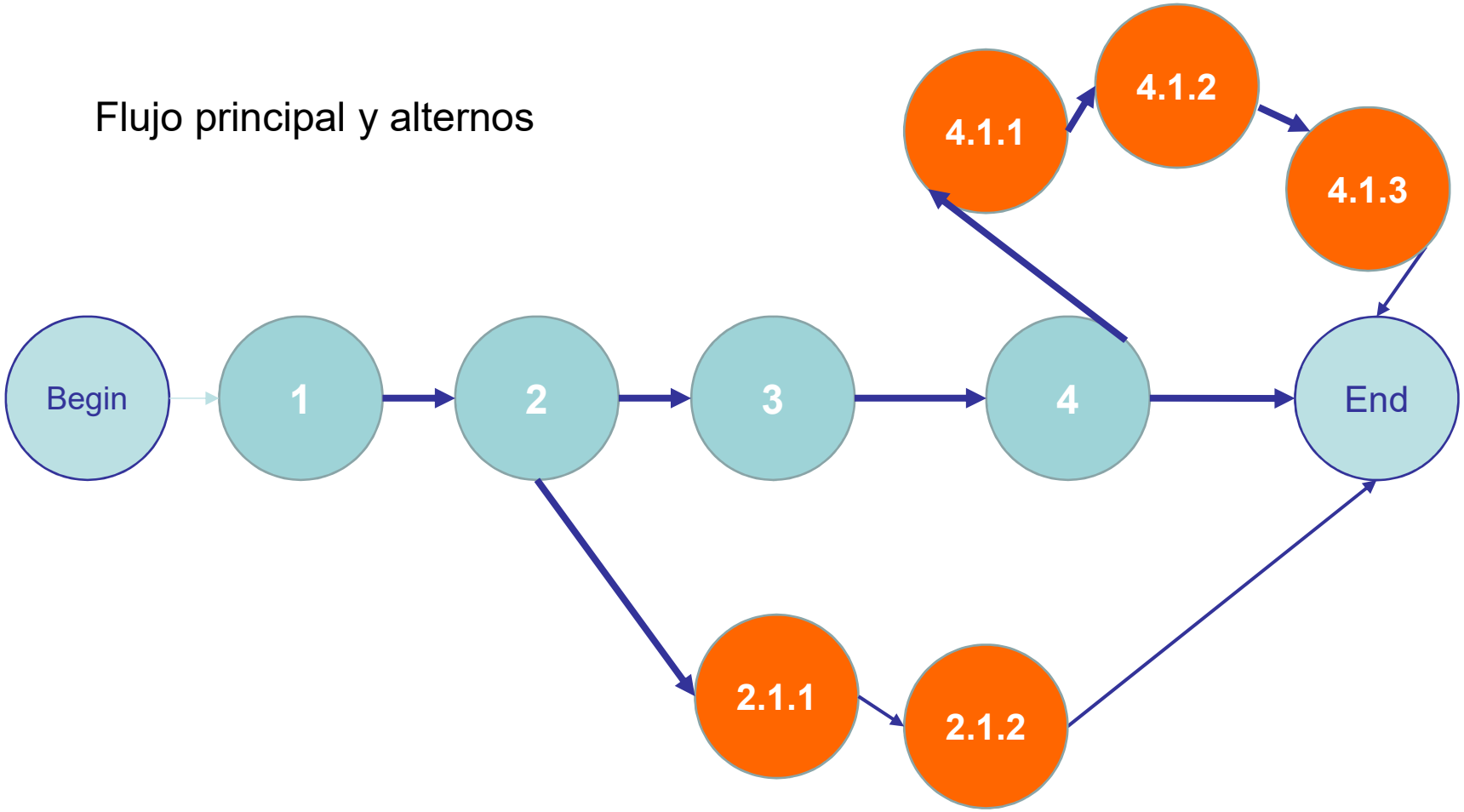


Descripción de Casos de Uso

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	**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 cajas **4** El sistema comprueba si el usuario ha completado el nivel, muestra la pantalla de juego y espera un nuevo movimiento
Alternativas	**2.1** movimiento no es válido: **2.1.1** El sistema no hace nada **2.1.2** El sistema espera un nuevo movimiento. **4.1** El usuario ha completado el nivel: **4.1.1** el sistema carga el siguiente nivel, **4.1.2** el sistema muestra la pantalla de juego, **4.1.3** el sistema espera un nuevo movimiento.

Descripción de Casos de Uso

Flujo principal y alternos



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

- Mediante **BDD**, Behavior-Driven Development (Desarrollo guiado por comportamiento), son especificados **escenarios** de funcionamiento para la aplicación y las **pruebas** de esta son desarrolladas en base a dichos **escenarios**.
  - BDD se enfoca en el **comportamiento** de la aplicación
  - De esta forma queda más claro para todos cuáles son los **objetivos** que una **funcionalidad** desea alcanzar y es de mucho más fácil entendimiento
  - Los **escenarios** acompañan a la **funcionalidad** de forma que muestran **comportamientos** para atenderla.
  - Una **funcionalidad** y sus **escenarios** son descritos usando el lenguaje Gherkin [y la herramienta Cucumber]

Behavior-driven Development – BDD

La funcionalidad es escrita siguiendo un patrón:

Elemento	Descripción	
Funcionalidad/Feature	<Nombre de la funcionalidad o Caso de Uso>	
[Descripción – historia de usario]	Como un <tipo de usuario>, quiero/deseo <objetivo>, [de manera que <alguna razón>]	
Escenario	<Serie de pasos involucrados en la ejecución de la funcionalidad y prueba>	
	Dado	<Estado inicial del sistema – precondiciones, estado y los parámetros relevantes para este escenario en particular>
	Cuando	<Acciones a ser realizadas en el sistema, un cambio de estado, o lo que estamos probando>
	Entonces	<El resultado esperado, estado resultante – postcondiciones>

Behavior-driven Development – BDD

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

Scenario: **…**

Given …

When …

Then …

Behavior-driven Development – BDD

```
Feature: Login

  I want to login on Keepfy

  Background:
    Given I go to '/login'
    And the field 'email' is empty
    And the field 'password' is empty

  Scenario: Error on empty fields
    When I click on 'enter'
    Then field 'email' should be with error
    And field 'password' should be with error

  Scenario: Wrong password
    When I type 'john.test@keepfy.com' in 'email'
    And I type '123456' in 'password'
    And I click on 'enter'
    Then I should see 'E-mail or password is incorrect'

  Scenario: Login successfully
    Given I have users:
      | name           | email             | password |
      | Vitor Batista  | vitor@keepfy.com  | abcdef   |
    When I type 'vitor@keepfy.com' in 'email'
    And I type 'abcdef' in 'password'
    And I click on 'enter'
    Then I shouldn't see 'Access your account'
```

```
Feature: Student registration
As a student I want to register for a course so that I can reserve a place
in the course with a limited capacity.

Scenario: Successful registration to a course
  Given I am a student
  And a lecture "PA042" with limited capacity of 20 students
  When I am on a registration page
  And I fill in "PA042"
  And I press "Register"
  Then I should see "You are successfuly registered"
  And I should be registered for the course

Scenario: Unsuccessful registration due to full course
  Given I am a student
  And a lecture "PA042" with limited capacity of 20 students
  But the capacity of this course is full
  When I am on a registration page
  And I fill in "PA042"
  And I press "Register"
  Then I should see "The course is full"
  And I should not be registered for the course
```

https://gist.github.com/vitorebatista/d979332bb57c1ad17b031f7e3e0cd677

Behavior-driven Development – BDD

- **Automation**

Feature: Login

Scenario: Valid User

Given a user with valid credentials

When they log in

Then they will have access to secure portions of the site

Scenario: Invalid User

Given a user with invalid credentials

When they log in

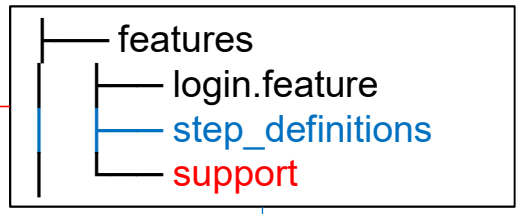
Then they will not gain access to secure portions of the site

How To Add Visual Testing To Your BDD Tests: How scenarios you've specified (using the Gherkin syntax) translates into test automation. 2015
https://applitools.com/blog/how-to-add-visual-testing-to-your-bdd-tests/

A Solution
By using the built-in functionality of our BDD tool of choice, we can easily generate step definitions for our scenarios and add in test execution with a third-party provider like **Applitools** (for visual testing) and Sauce Labs (for access to additional browsers/devices)

```
Before do |scenario|
  @eyes = Applitools::Eyes.new
  @eyes.api_key = 'your Applitools API key'
  caps = Selenium::WebDriver::Remote::Capabilities.internet_explorer
  caps.version  = '8'
  caps.platform = 'Windows XP'
  caps['name'] = scenario.title
  browser = Selenium::WebDriver.for(
    :remote,
    url: "http://your-sauce-username:your-sauce-access-
key@ondemand.saucelabs.com:80/wd/hub",
    desired_capabilities: caps)
  @driver = @eyes.open(app_name: 'the-internet', test_name: scenario.title, driver:
browser)
end

After do
  @eyes.abort_if_not_closed
  @driver.quit
end
```




```
When(/^they log in$/) do
  @driver.get 'http://the-internet.herokuapp.com/login'
  @driver.find_element(id: 'username').send_keys(@user[:username])
  @driver.find_element(id: 'password').send_keys(@user[:password])
  @driver.find_element(id: 'login').submit
end
```

DOCUMENTO DE REQUISITOS

2

2. Documento de requisitos

- Como resultado del proceso de elicitación se desarrolla el documento de especificacion 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. Es importante no utilizar términos relacionados con la implementación, como "archivo" y "menú“, “boton”, ...

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.

IREB, 2015

Características de Calidad

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

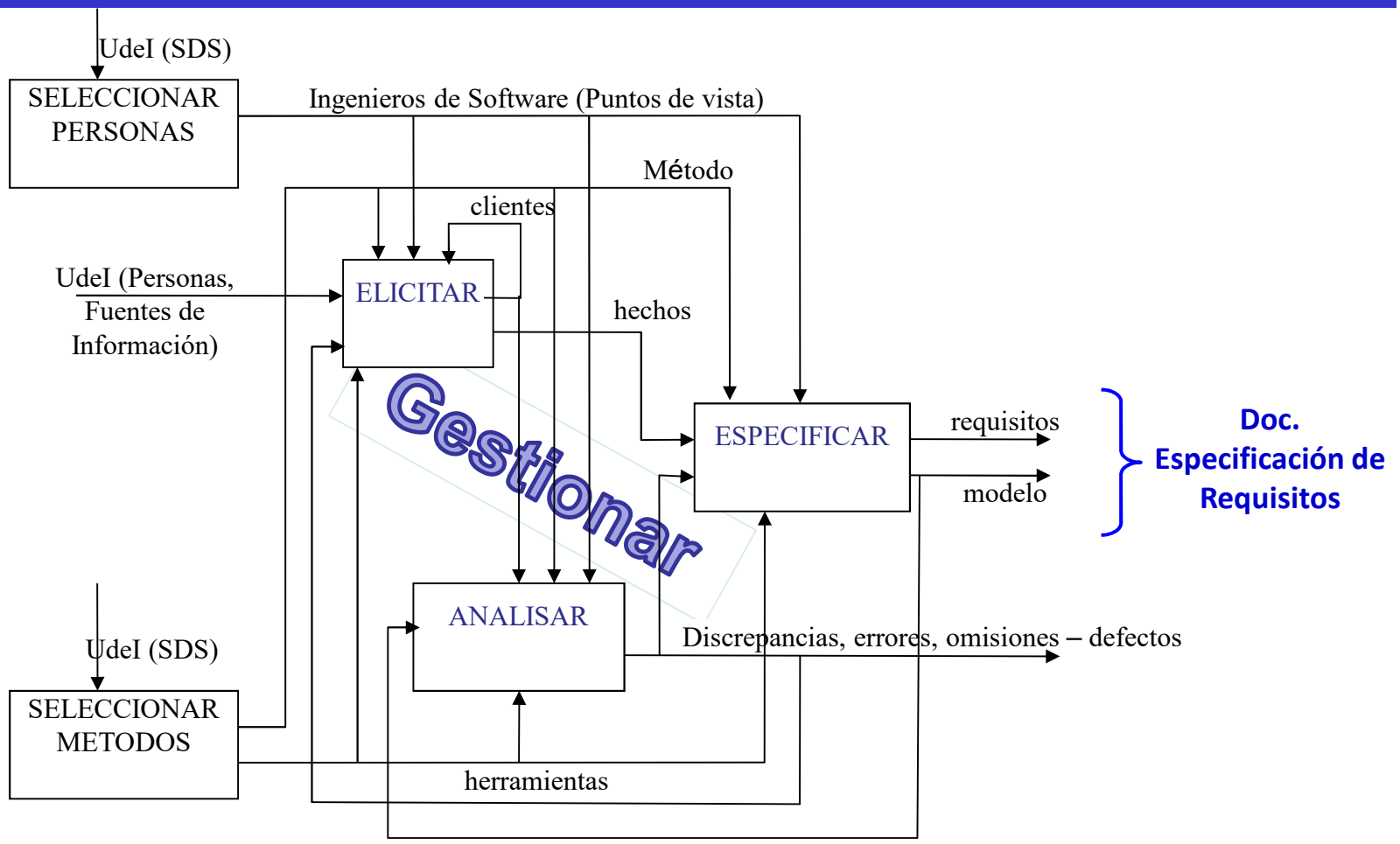
- **Especificación**
  - Unambiguity and consistency
  - Clear structure
  - Modifiability and extendibility
  - Completeness
  - Traceability
- **Requisitos**
  - *Agreed*:
  - *Unambiguous*:
  - *Necessary*:
  - *Consistent*:
  - *Verifiable*:
  - *Feasible*:
  - *Traceable*:
  - *Complete*:
  - *Understandable*:

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.

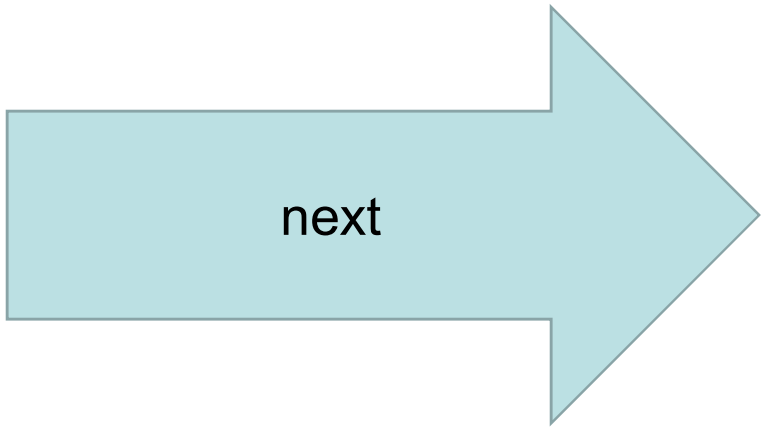
IREB, 2015

Conclusión



4

Continua...



Referencias

- **Basado en:**

  - Leite, J.C.S.P. 2007. Livro Vivo : Engenharia de Requisitos, http://livrodeengenhariaderequisitos.blogspot.com/
  - **Pohl, K. and Rupp, C. 2015. Requirements Engineering Fundamentals. IREB**
  - **Software Requirements - K Wiegers - 3Ed - Microsoft – 2013. http://wdz.eng.br/LivroRequirements3Ed.pdf**
  - **Jacobson, I., Spence, I., & Bittner, K. (2011). Use Case 2.0: The Guide to Succeeding with Use Cases. Ivar Jacobson International.** https://www.ivarjacobson.com/sites/default/files/field_iji_file/article/use-case_2_0_jan11.pdf
  - 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
  - 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. Sokobanhttp://www.lsi.us.es/~javierj/cursos_ficheros/03.%20Sokoban.%20Un%20ejemplo%20de%20plantillas.pdf