UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Departamento de Electrónica

Ingeniería de Software
Algunas herramientas de apoyo para:
Definición de requerimientos, análisis y diseño

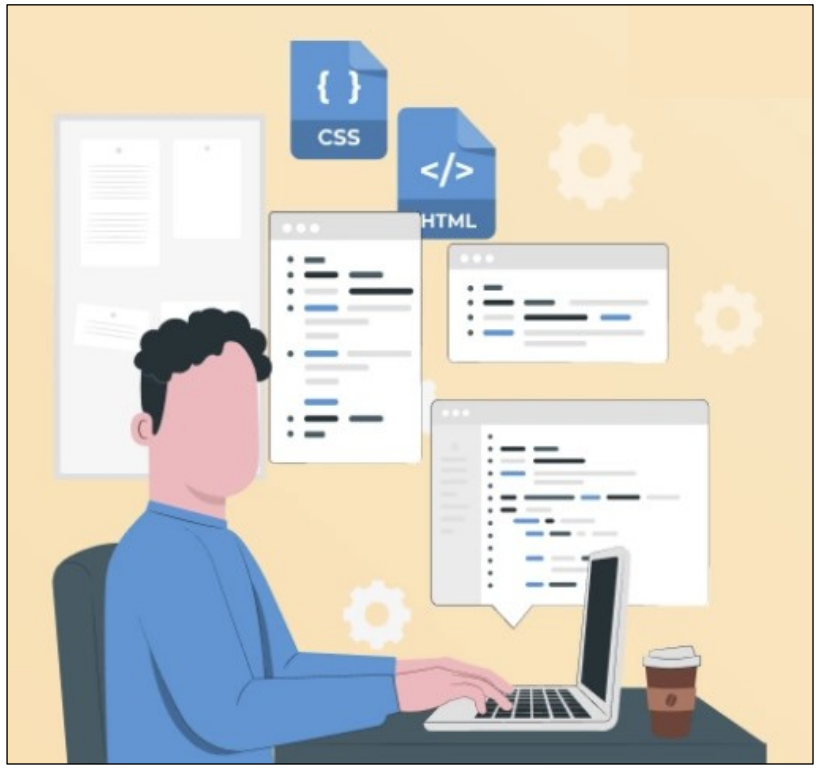
ELO329: Diseño y Programación Orientados a Objetos

Departamento de Electrónica

Universidad Técnica Federico Santa María

Diseño y Arquitectura del Software

El diseño y arquitectura del software se construyen durante la fase de diseño del **SDLC**



SDLC

Arquitectura de Software

- Representa las decisiones de diseño (estructura y comportamiento general del sistema).
- Ayuda a la comprensión y análisis de cómo el sistema logrará objetivos esenciales (modificabilidad, estabilidad y seguridad).
- Define/Describe las estructuras fundamentales y comportamiento de un software.
- Define/Describe interacciones de los componentes entre sí y principios utilizados para diseñar el software.

Diseño y Arquitectura del Software

El diseño y arquitectura del software se construyen durante la fase de diseño del **SDLC**

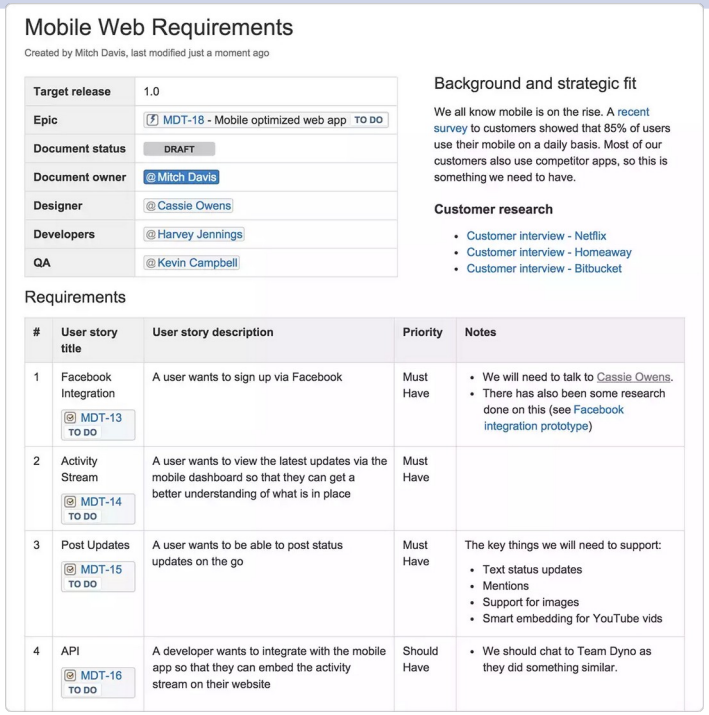
Analistas o Arquitectos de Software

- Diseña y describe la arquitectura de un proyecto.
- Comunica la arquitectura a los miembros del equipo.
- Guía la selección de herramientas tecnológicas:
  - Software / Lenguajes de programación.
  - Bibliotecas / Librerías.
  - Frameworks.
- Evalúan de manera anticipada las ventajas y desventajas de la selección de las herramientas.
- Documenta la arquitectura del sistema, elabora diagramas, especificaciones técnicas y otras formas de documentación.

Diseño y Arquitectura del Software. Artefactos

Diseño Arquitectónico → **Artefactos** = elementos del diseño del software

SDD	Diagramas Arquitectónicos	Diagramas (UML) Unified Modeling Language
❑ Documentan aspectos técnicos de la implementación del diseño: dependencias, requerimientos, restricciones, objetivos, otros.	❑ Describen los componentes, interacciones, restricciones, límites, patrones de arquitectura.	❑ Visualizan la estructura, diseño, arquitectura e implementación del software mediante un lenguaje común.



Mobile Web Requirements

Created by Mitch Davis, last modified just a moment ago

Target release	1.0
Epic	MDT-18 - Mobile optimized web app TO DO
Document status	DRAFT
Document owner	@Mitch Davis
Designer	@Cassie Owens
Developers	@Harvey Jennings
QA	@Kevin Campbell

Background and strategic fit

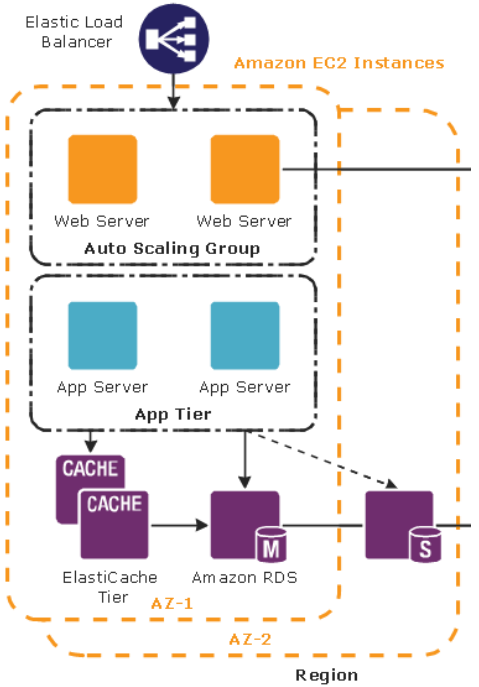
We all know mobile is on the rise. A recent survey to customers showed that 85% of users use their mobile on a daily basis. Most of our customers also use competitor apps, so this is something we need to have.

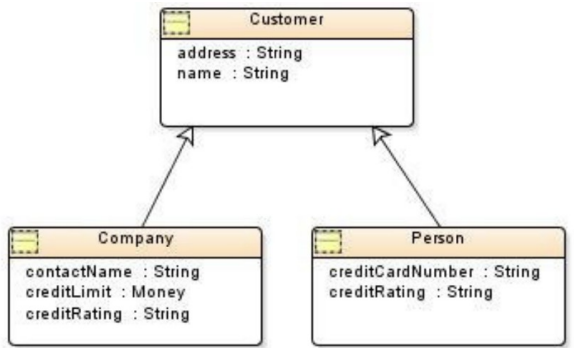
Customer research

- Customer interview - Netflix
- Customer interview - Homeaway
- Customer interview - Bitbucket

Requirements

#	User story title	User story description	Priority	Notes
1	Facebook Integration MDT-13 TO DO	A user wants to sign up via Facebook	Must Have	• We will need to talk to Cassie Owens. • There has also been some research done on this (see Facebook integration prototype)
2	Activity Stream MDT-14 TO DO	A user wants to view the latest updates via the mobile dashboard so that they can get a better understanding of what is in place	Must Have	
3	Post Updates MDT-15 TO DO	A user wants to be able to post status updates on the go	Must Have	The key things we will need to support: • Text status updates • Mentions • Support for images • Smart embedding for YouTube vids
4	API MDT-16 TO DO	A developer wants to integrate with the mobile app so that they can embed the activity stream on their website	Should Have	• We should chat to Team Dyno as they did something similar.





Ejemplos Referenciales

Diseño y Arquitectura del Software. Artefactos

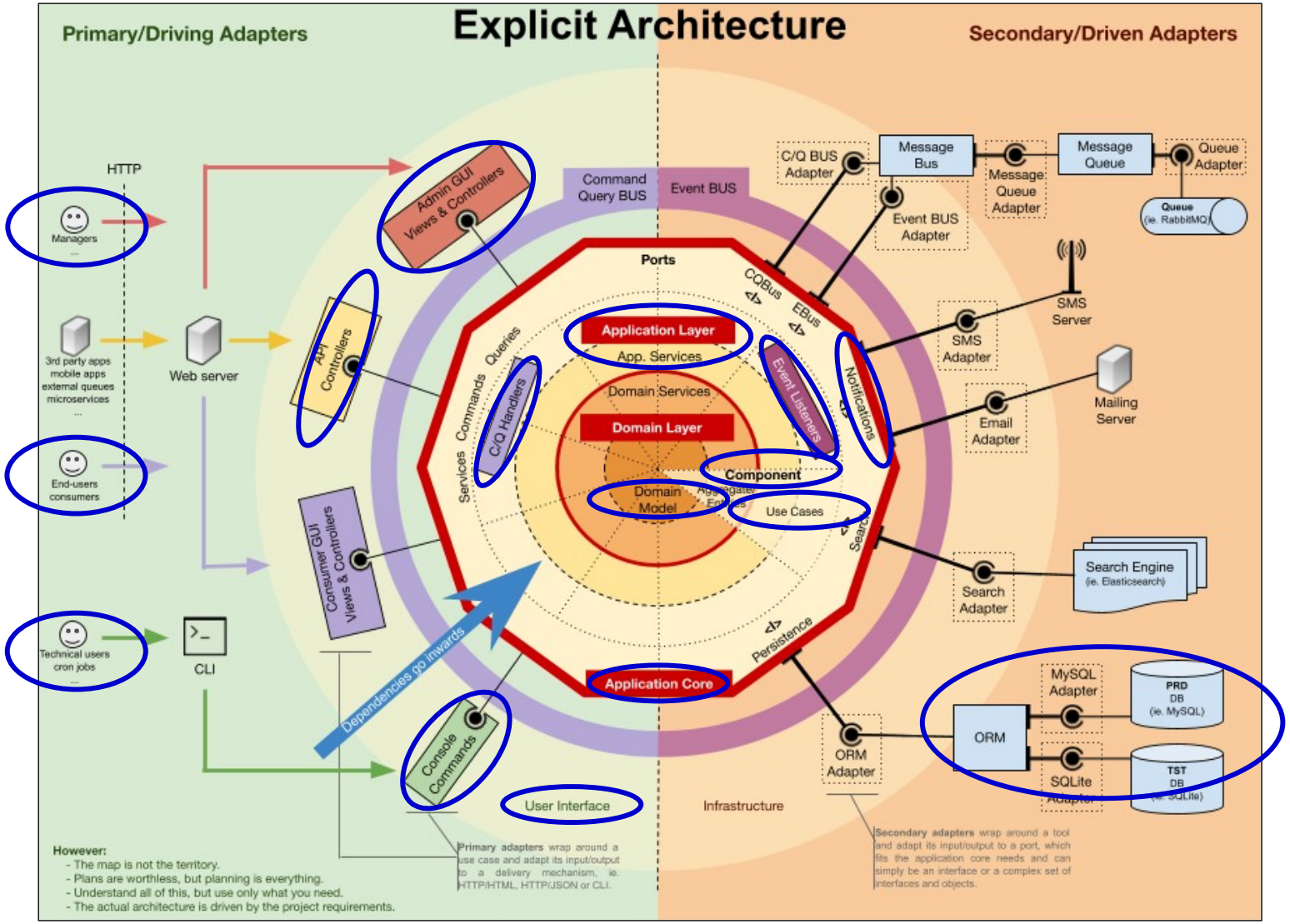
Ejemplo:
Diagrama
Arquitectónico
o
Diseño
Estructurado

Módulos

Sub-módulos

Comunicación

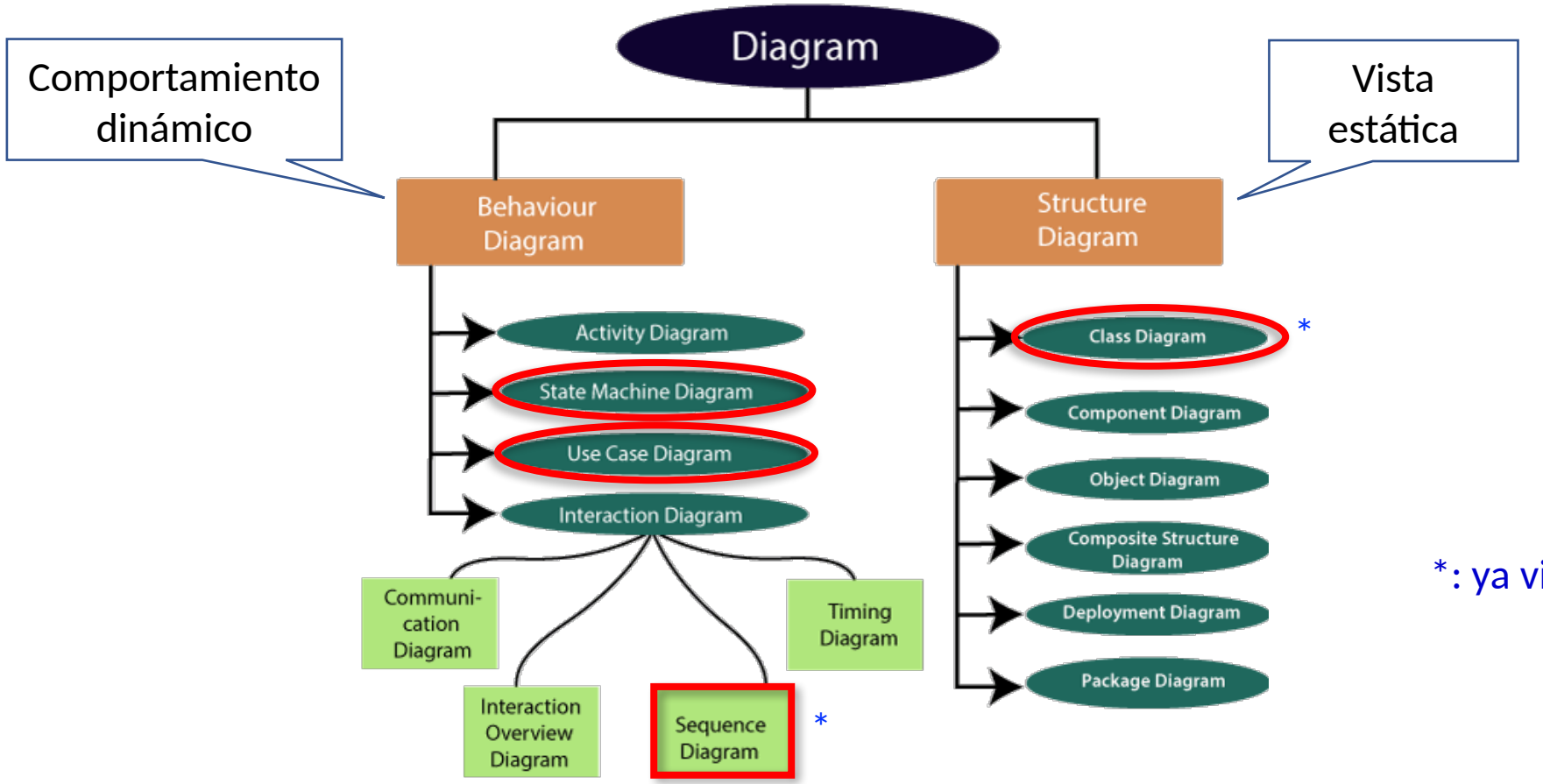
Flujo



Fuente: The Software Architecture Chronicles, a series of posts about Software Architecture (https://herbertograca.com/tag/software-architecture/)
Relatives of Onion Architecture: Hexagonal and Clean Architecture Annotations #70 (https://github.com/xmolecules/jmolecules/discussions/70)

Diseño y Arquitectura del Software. Artefactos

Diagramas UML (Unified Modeling Language)



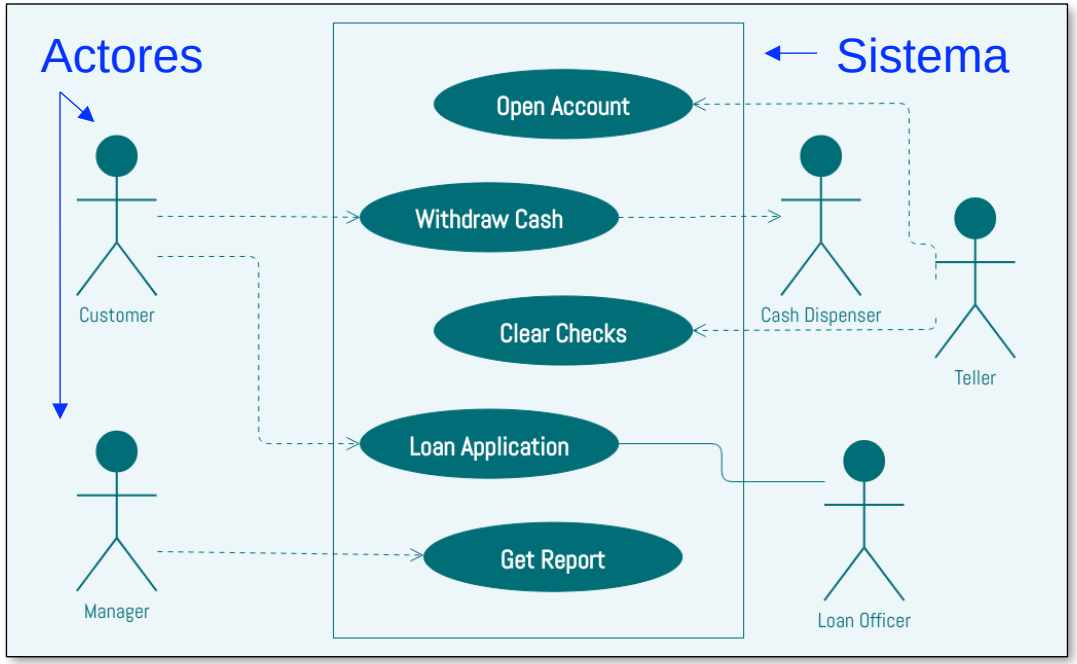
*: ya vistos en clases

Fuente: UML Tutorials - Herong's Tutorial Examples (http://www.herongyang.com/UML/Introduction-UML-Diagrams.html)
UML-Diagrams (https://www.javatpoint.com/uml-diagrams)

UML: Diagrama de Casos de Uso del Sistema (1/6)

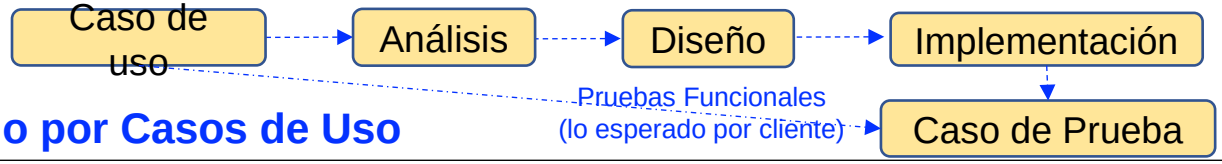
Casos de uso: **Es una técnica para levantar requisitos de un sistema**

- Define **requerimientos** y **análisis**.
- Utilidad durante las pruebas.
- Cada caso de uso se concentra en un escenario específico y describe la interacción entre un **actor principal** y el **sistema**.



- Actores: usuarios, sistemas, hardware.
- Cada resultado (salida) tiene un valor para cada uno de los actores.
- Existen variaciones para situaciones excepcionales.
- Enfatiza en qué hace el sistema (no en cómo lo hace).

Cada iteración puede desarrollarse según:



Proceso dirigido por Casos de Uso

Fuente: Use Case Diagram: Banking Systems (https://online.visual-paradigm.com/diagrams/templates/use-case-diagram/use-case-diagram-banking-system/)

Casos de Uso del Sistema (2/6)

- Se construye en base a levantamiento inicial de requerimientos y en reuniones de análisis con usuarios.
- Proporcionan un lenguaje común entre los desarrolladores, usuarios y otras partes interesadas involucradas en el proyecto.

- Un caso de uso debe ser simple, claro y conciso.
- Se deben identificar los actores involucrados (entidades externas al sistema que interactúan con él).

- Cada caso de uso debe ser entendible para el cliente.

- Son buenos para capturar requerimientos reflejados en comportamientos del sistema ante determinadas entradas, éstos son los requerimientos funcionales.

- No son buenos para requerimientos no funcionales. Ej., plataforma, desempeño, seguridad, tiempo de respuesta.
- Pueden volverse muy detallados y complejos, especialmente para sistemas grandes y complejos.
- Requiere un esfuerzo significativo y puede llevar mucho tiempo.

Casos de Uso del Sistema (3/6)

Ejemplo de Plantilla para documentar/registrar Casos de Uso

1	**Nombre:** Nombre del caso de uso, usar verbo y sustantivo, debe sintetizar el objetivo deseado. Ej., Cambiar nota.
2	**Propósito:** Resumen simplificado de lo que se desea lograr con este caso de uso.
3	**Actores:** Entes externos que participan en el caso de uso. Ej. Profesor
4	**Pre-condiciones:** Pre-requisitos existentes (que se prevén) para el correcto funcionamiento de la funcionalidad especificada en el caso de uso. Ej. La nota ya existe, el profesor ya se autenticó.
5	**Evento:** Situación que gatilla el inicio del caso de uso. Ej. Profesor terminó re-corrección.
5	**Curso normal de eventos (o detalle):** aquí se describe una secuencia numerada de pasos relatando el flujo básico o feliz del caso de uso. Se sugiere separar en dos columnas.
6	**Curso alternativo de eventos:** Funcionalidad que se requiere en caso de error.
7	**Requerimientos no funcionales:** Especificación narrativa de solicitudes no funcionales del usuario que especifican situaciones de rendimiento, volúmenes de información, seguridad, tiempos de respuesta, etc.
8	**Autor:** Persona(s) responsable del análisis y redacción del caso de uso.

Actor	Sistema
1)	
	2)

Lectura de ejemplo: (https://lsi2.ugr.es/~mvega/docis/casos%20de%20uso.pdf)

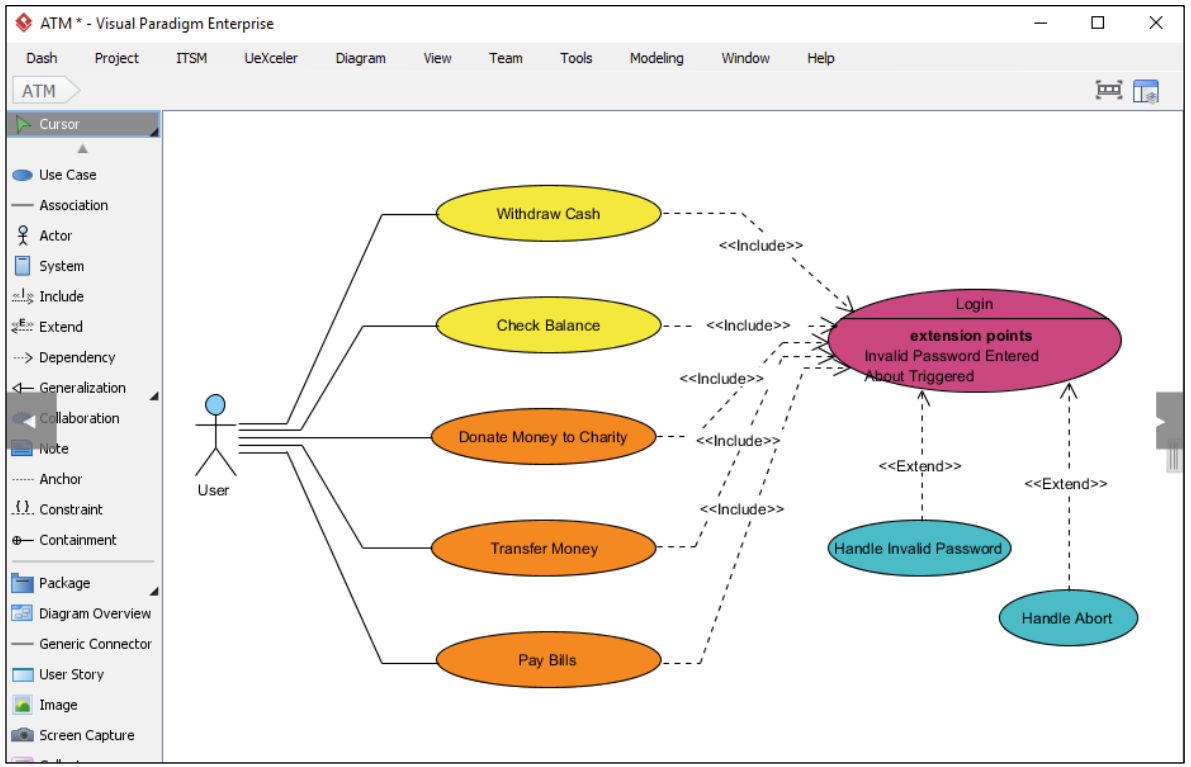
Ejemplo de plantilla (otra) en formato .docx [10]

[10] Fuente: Functional Requirements and Specifications: A Quick Tutorial. By Laura Brandenburg (CEO of Bridging the Gap) (https://www.bridging-the-gap.com/uctemplate/)

UML: Diagrama de Casos de Uso del Sistema (4/6)

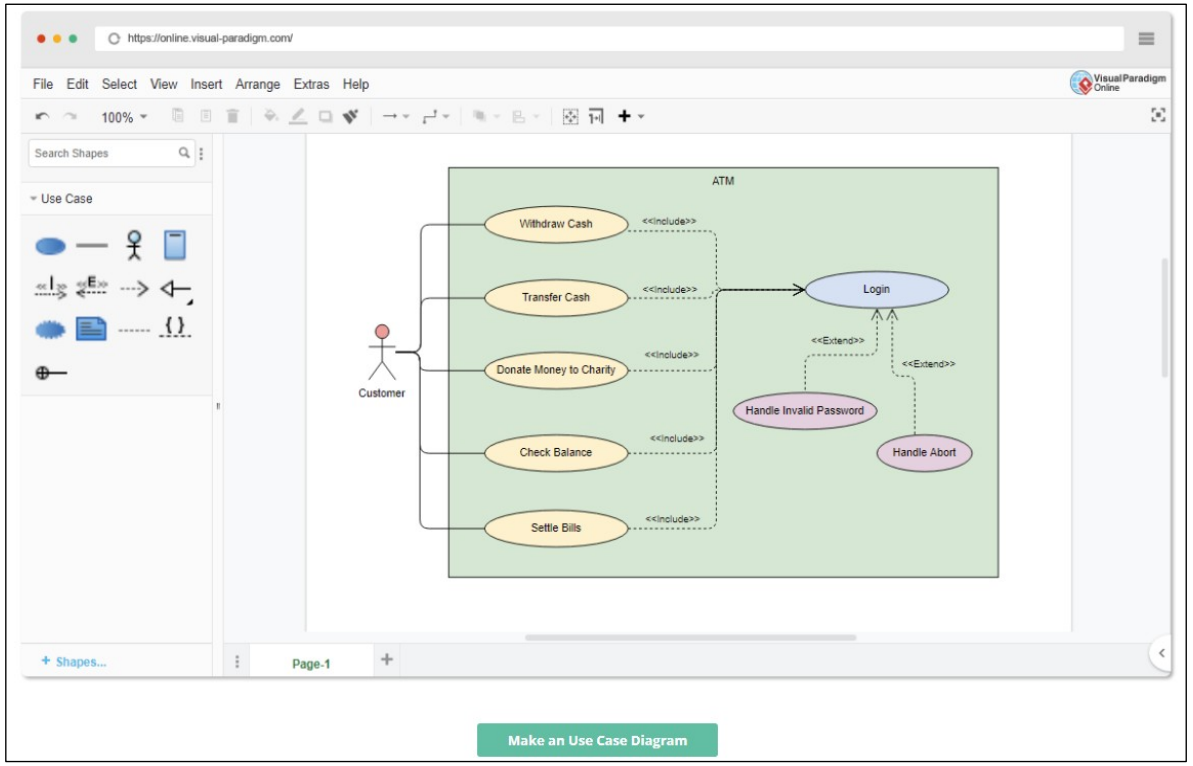
Ejemplo de Herramienta para crear diagramas de Casos de Uso

Visual Paradigm



Visual Paradigm (desktop version)
(https://www.visual-paradigm.com/download/)

Visual Paradigm Online



Visual Paradigm (online version)
(https://online.visual-paradigm.com/diagrams/features/use-case-diagram-software/)

Casos de Uso del Sistema (5/6)

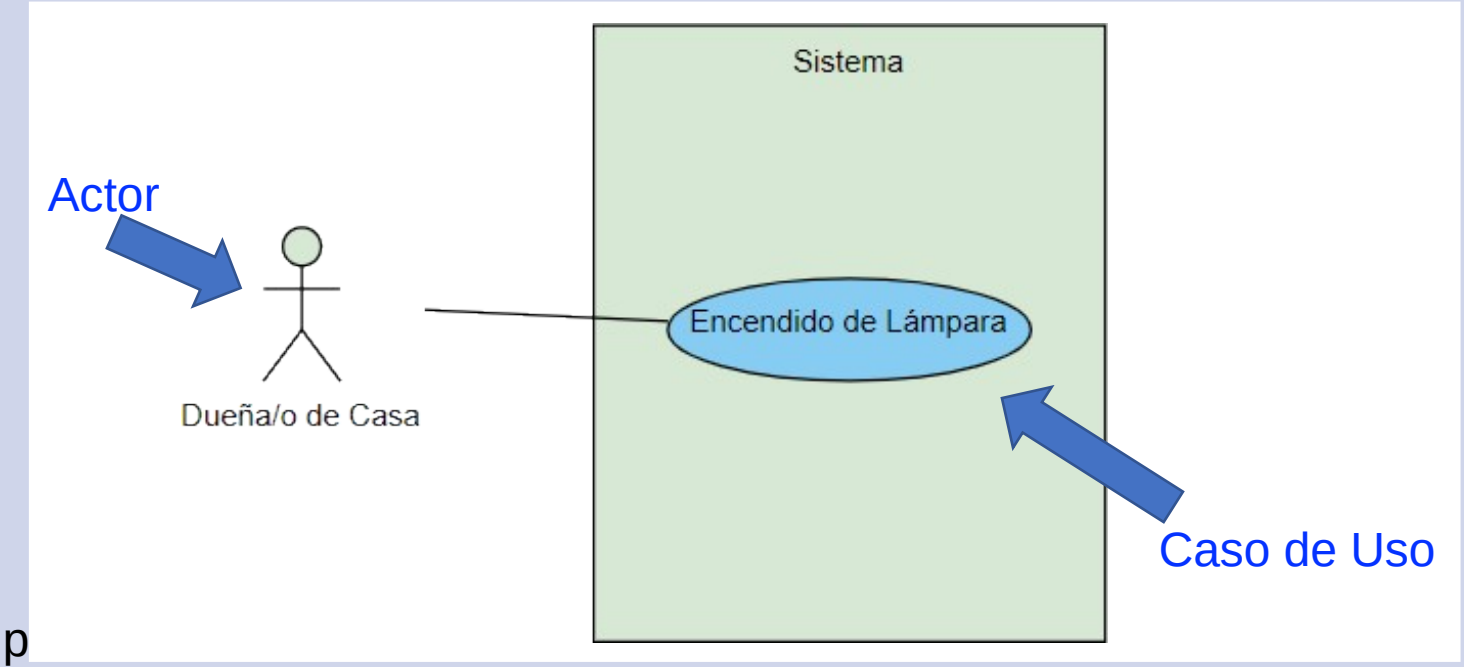
Ejemplo de caso de uso: Sistema de “Lámpara Domótica”

<table>
<tr><td>1</td><td>Nombre: Encendido de lámpara.</td></tr>
<tr><td>2</td><td>Propósito: El usuario remoto desea encender una luz de su casa.</td></tr>
<tr><td>3</td><td>Actores: Dueña de casa.</td></tr>
<tr><td>4</td><td>Pre-condición: La lámpara está configurada, registrada en la nube.</td></tr>
<tr><td>5</td><td>Evento: La dueña presiona el botón power del control remoto.</td></tr>
<tr><td>5</td><td>Curso normal de eventos (o detalle): aquí se describe una secuencia numerada de pasos relatando el flujo básico o feliz del caso de uso. Se sugiere separar en dos columnas.
<table>
<tr><th>Actor</th><th>Sistema</th></tr>
<tr><td>1. La dueña de casa selecciona un canal y presiona el botón power del control remoto.</td><td>2. El sistema cambia a azul el botón power.</td></tr>
<tr><td></td><td>3. El sistema enciende las lámparas configuradas en el canal del control remoto.</td></tr>
</table></td></tr>
<tr><td>6</td><td>Curso alternativo de eventos: Es común especificar variantes de un caso de uso:
❑ Variante 1:
▪ 1A1: El usuario selecciona un canal sin lámparas y presiona botón power.
▪ 2A1: El sistema cambia a azul el botón power.
▪ 3A1. No se encienden lámparas por no haberlas en ese canal.
❑ Variante 2:
• 3A2: La luz ya estaba encendida, permanece encendida.</td></tr>
<tr><td>7</td><td>Requerimientos no funcionales: Se debe poder manejar hasta 10 lámparas en un mismo canal.</td></tr>
<tr><td>8</td><td>Autor: Juan Pérez.</td></tr>
</table>

UML: Diagrama de Casos de Uso del Sistema (6/6)

Ejemplo de caso de uso: Sistema de “Lámpara Domótica”

- Su objetivo es representar gráficamente una funcionalidad provista por el sistema.



- Son buenos p
  - Identificar nuevos requerimientos (Ej., para apagar luz, se debe poder cambiar el canal).
  - Comunicarse con el cliente.
  - Generar pruebas.

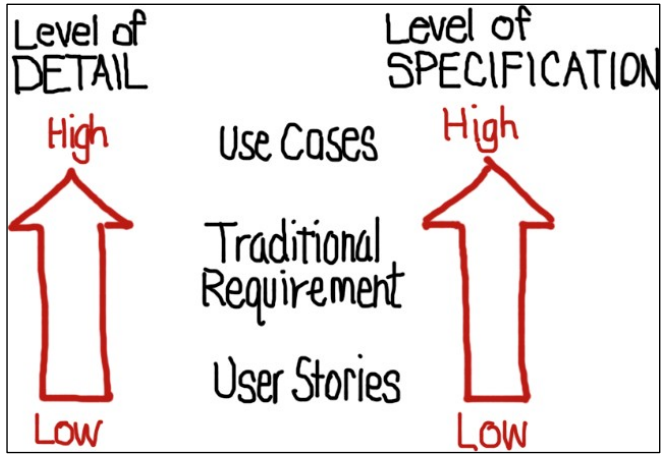
Historias de Usuario y Casos de Uso (1/4)

Historias de Usuario

- Funcionalidad computacional resumida en una tarjeta para **también levantar requisitos** de un sistema. Es una conversación.
- No existe certeza absoluta de que una funcionalidad se va a programar.
- Situaciones no tan definidas, no tan claras, se hace una exploración.
- Descripción relativamente genérica, no tan precisa, mucho menos detallada.
- Menos esfuerzo y tiempo.
- Una buena cantidad de historias de usuario no se implementan, porque, se dan cuenta de que hay otras historias que aportan más valor al negocio.

Casos de Uso

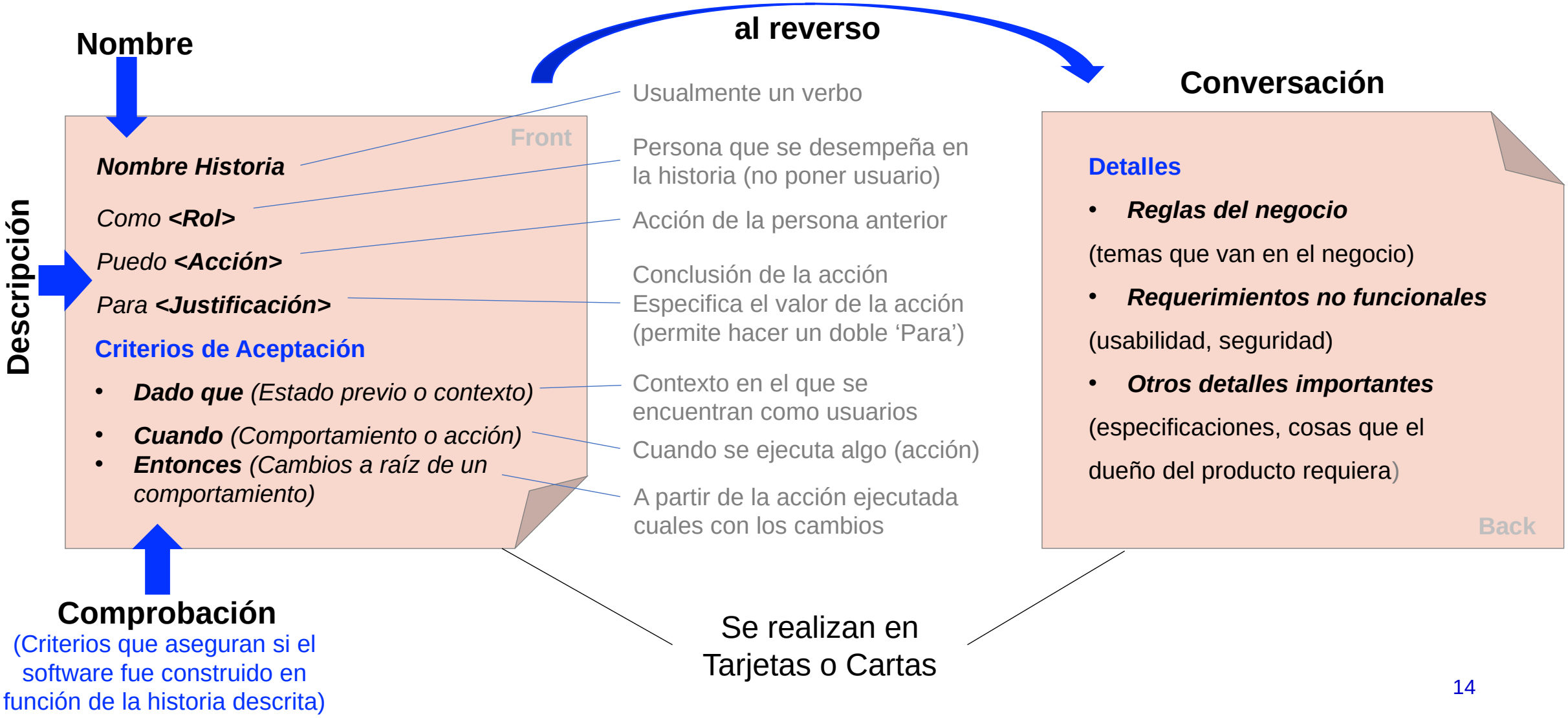
- Especificación de requerimientos muy precisa, muy detallada, muy paso a paso.
- Buenas cuando los requerimientos seguramente se van a implementar.
- Requiere esfuerzo y tiempo.



¿Cuál utilizar? HU cuando hay incertidumbre, si hay claridad casos de uso.

Historias de Usuario y Casos de Uso (2/4)

Historias de Usuario: Estructura



Historias de Usuario y Casos de Uso (3/4)

Ejemplo de Historia de Usuario: Sistema de “Encendido de Lámpara”

Front

Encender lámpara desde sistema domótico

Como ***<cuidador de museo>***

Puedo ***<encender lámpara a distancia desde aplicación en celular>***

Para ***<no tener que caminar y encender la lámpara manualmente>***

Criterios de Aceptación

- *Dado que (**quiero encender la lámpara con mi celular**)*
- *Cuando (**en mi celular elijo la aplicación con el sistema domótico y enciendo la lámpara**)*
- *Entonces (**se solicitará que proceda con el encendido utilizando la aplicación**)*

Detalles

- *Reglas del negocio* ***(El listado de botones en el sistema domótico deberá contener nombre, estado y figura)***
- *Requerimientos no funcionales* ***(El encendido de la lámpara deberá tomar menos de 5 segundos)***
- *Otros detalles importantes* ***(Previo a solicitar el encendido de la lámpara procede a una validación de sensores disponibles en la zona)***

Back

Historias de Usuario y Casos de Uso (4/4)

Ejemplo de Mapping/Board de Historias de Usuario en un Sprint

Gestión de Proyectos con JIRA (1/3)

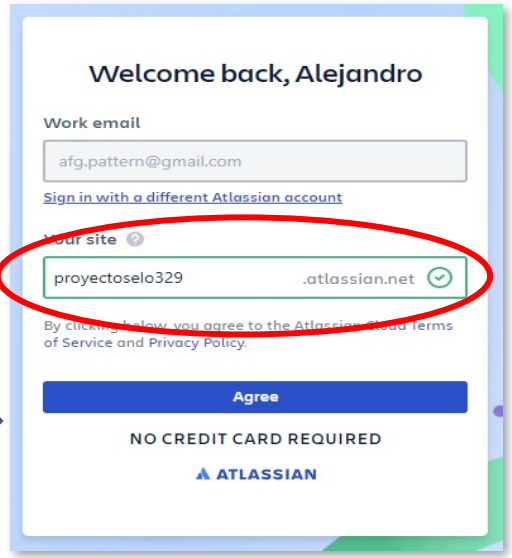
Atlassian
Jira Software

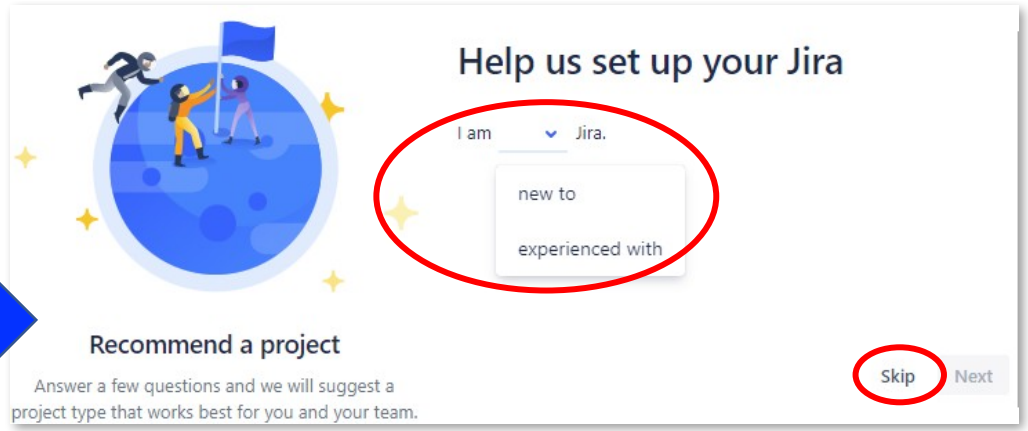
- JIRA: Herramienta de gestión de **Proyectos Ágiles** utilizada para diferentes tipos de equipos de trabajo.
- Muy personalizable y adaptable a las necesidades del Proyecto.
- Dispone de add-ons para potenciar sus funcionalidades base.
- Herramienta propietaria de Atlassian.

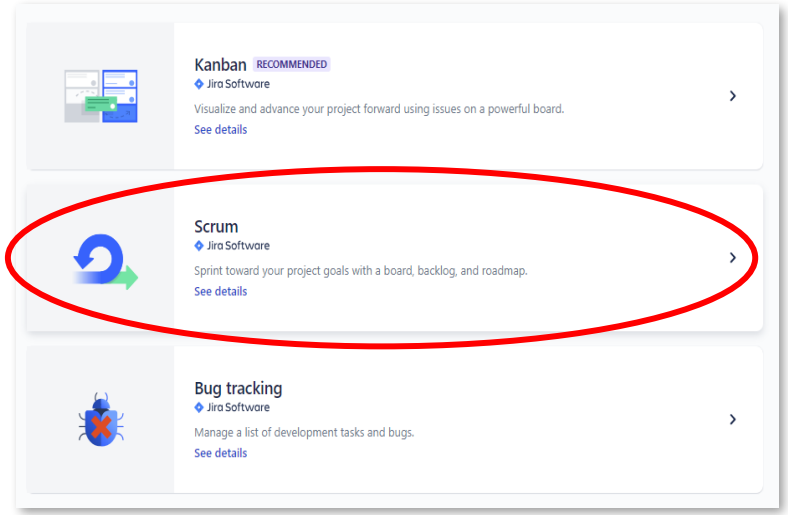
Enlace: (https://www.atlassian.com/es/software/jira/free)
Recomendación: Registrar con cuenta de google.

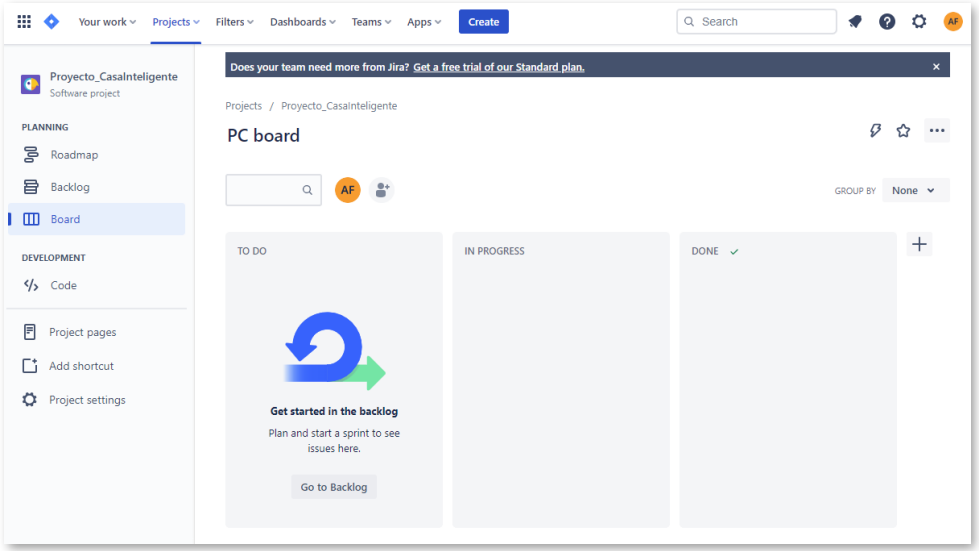
Gestión de Proyectos con JIRA (1/3)

Creando un Proyecto con JIRA







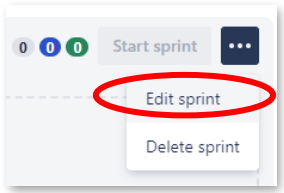


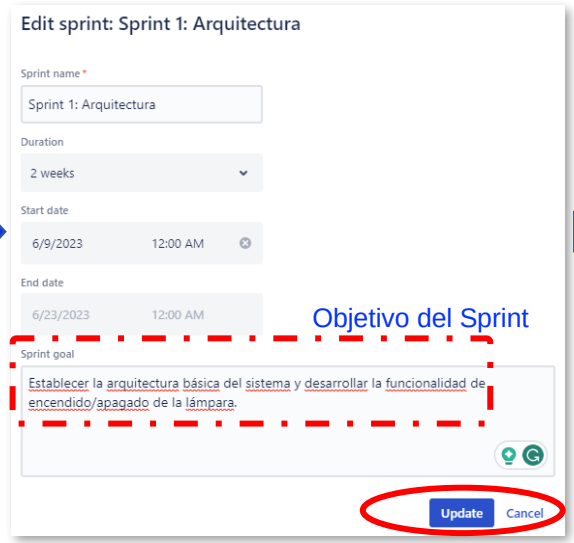
Gestión de Proyectos con JIRA (2/3)

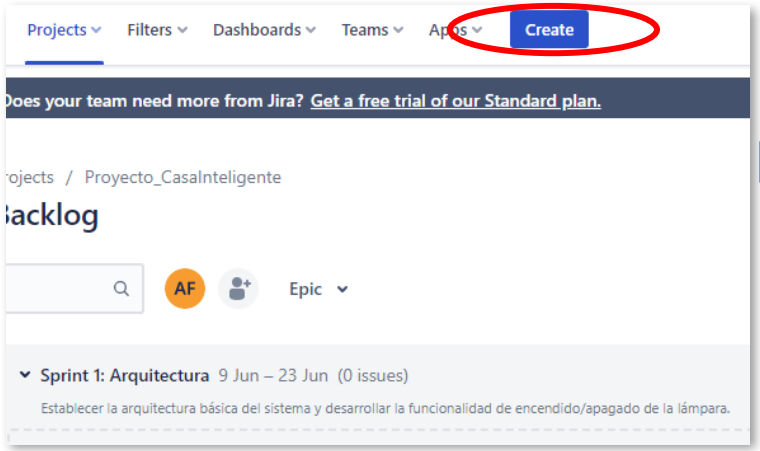
Incidencias en JIRA

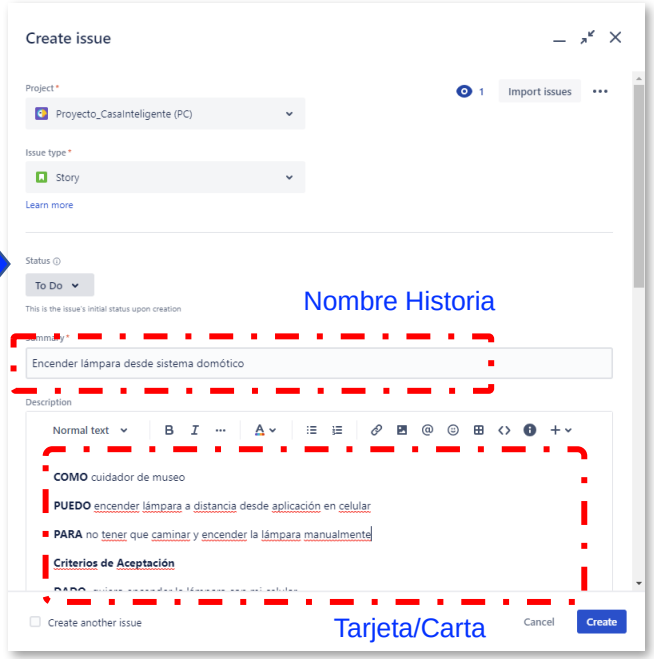
Gestión de Proyectos con JIRA (3/3)

Creando Historia de Usuario en JIRA

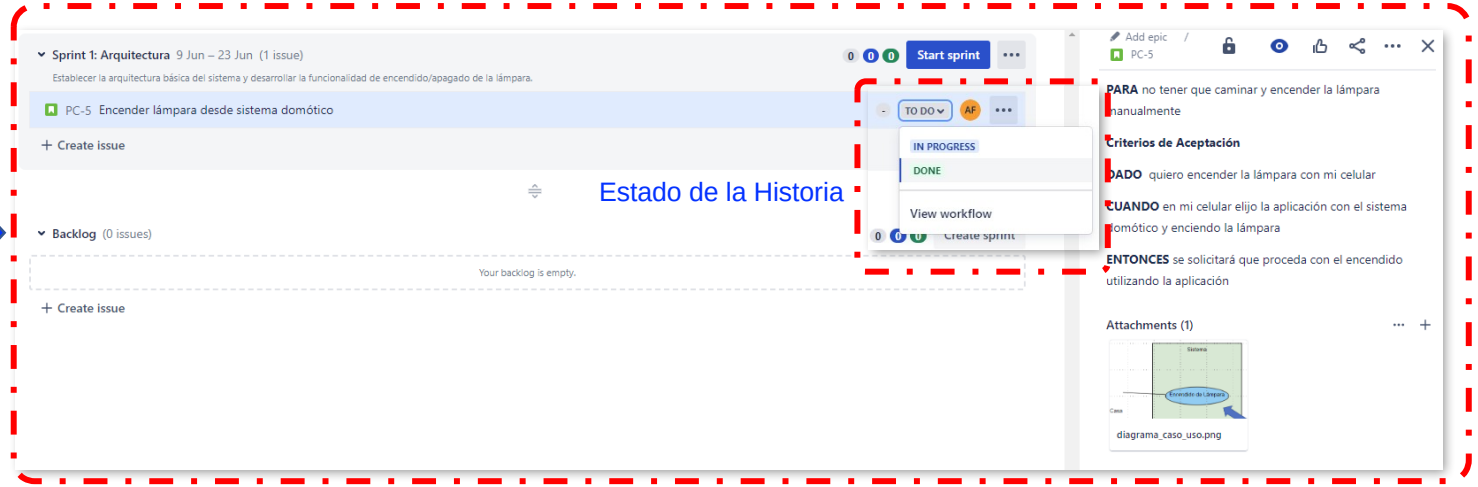








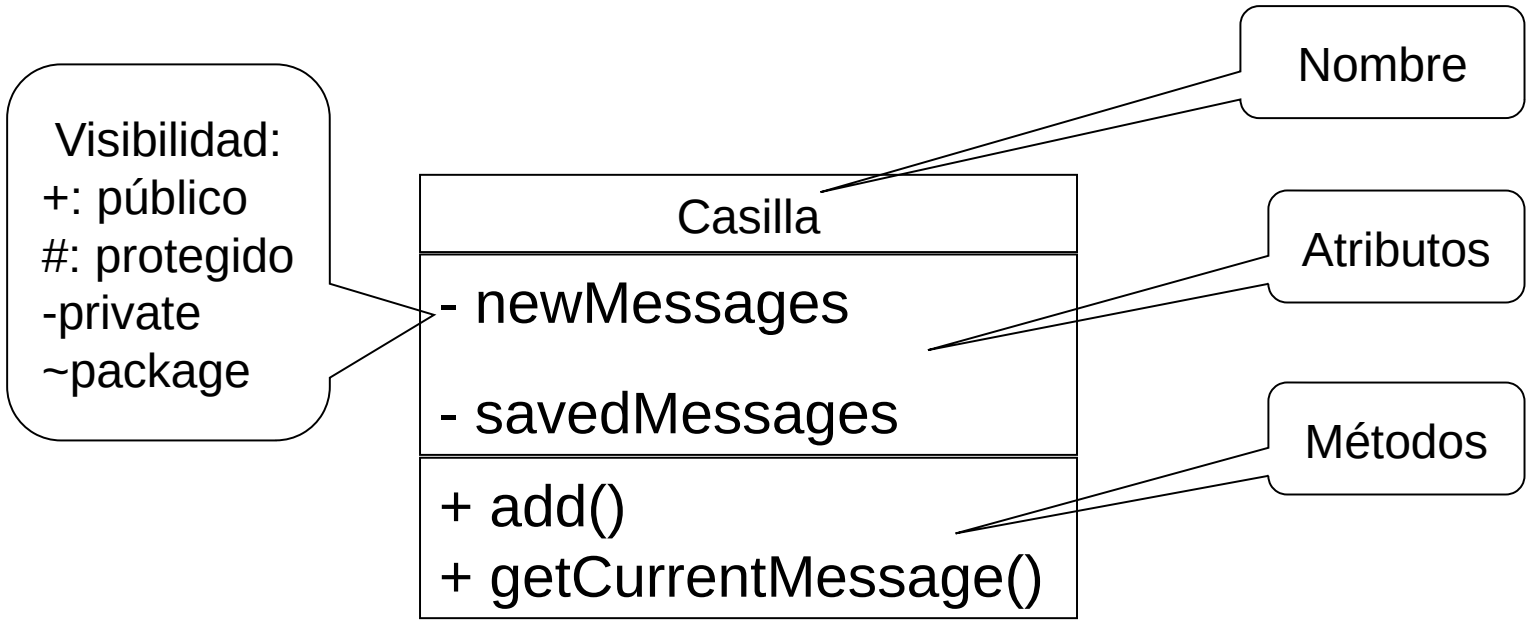
Listado de Historias de Usuarios y más…



UML: Diagrama de Clases del Sistema (1/5)

Comunican la estructura de un sistema de software en **object-oriented analysis and design** (**OOAD**)

- Cada clase es representada por sus atributos y métodos:

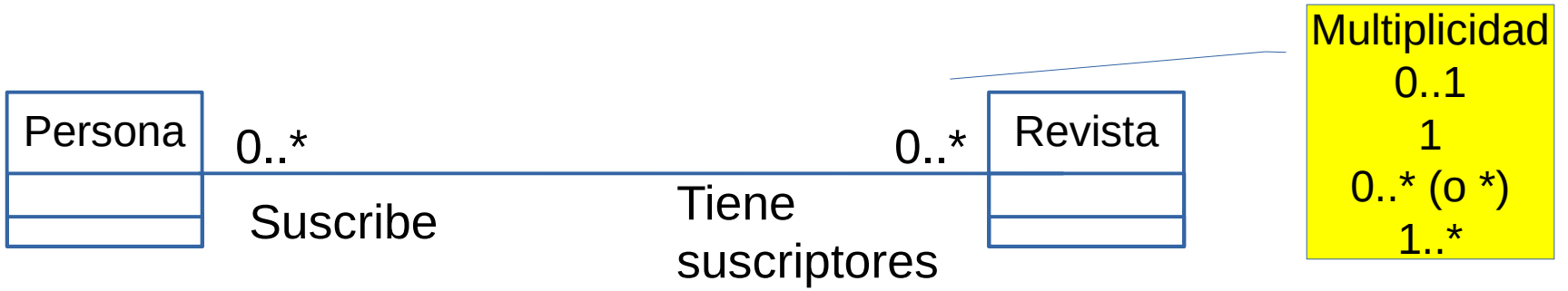


- Muestra cómo las clases en un OOD se relacionan entre sí.
- La representación varía según la herramienta usada.

UML: Diagrama de Clases del Sistema (2/5)

Tipos de relaciones entre clases

- **Asociación**: es la relación más general. Representa una familia de relaciones, la asociación puede ser unidireccional o bidireccional. Puede tener roles, algunos la usan a cambio de agregación.



- Para indicar direccionalidad se usan flechas:

- Por ejemplo, un mensaje no sabe en qué cola de mensajes está.

UML: Diagrama de Clases del Sistema (3/5)

Tipos de relaciones entre clases

- **Agregación**: Relación “tiene” o “contiene”, la parte puede existir fuera del todo.

  Típica relación entre la clase y sus atributos.

(Ver Relación Composición)[11]

- **Herencia**: Cuando se cumple la relación es-un y además hay una relación de sub-tipo válida.

[11]Fuente: Class Diagram Example: GUI (https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/)

UML: Diagrama de Clases del Sistema (4/5)

Tipos de relaciones entre clases

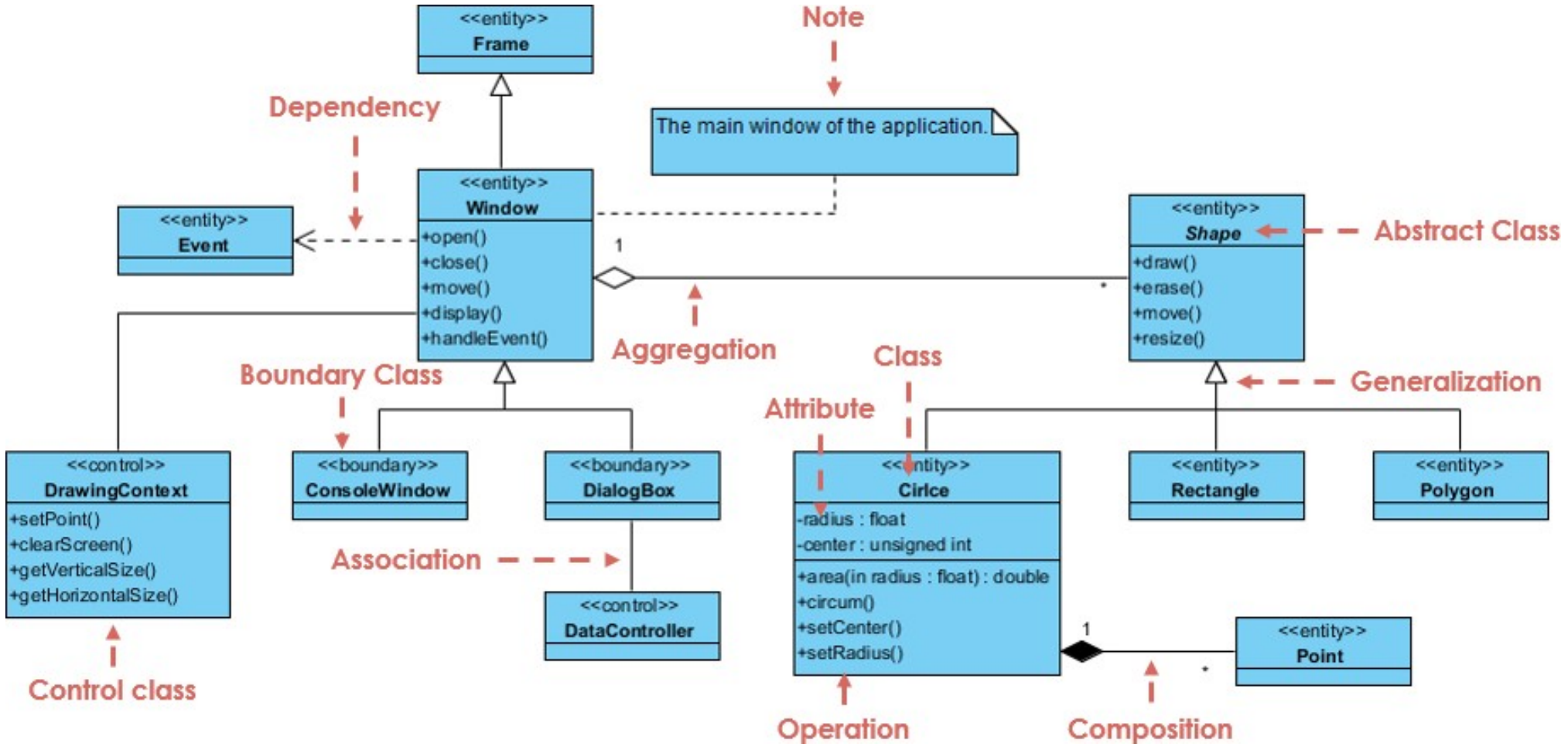
- **Interfaces**: Describe un conjunto de métodos.

- **Dependencia**: Es la más débil de las asociaciones. Indica que una clase usa otra en algún momento. Existe dependencia si una clase aparece en un parámetro o variable local de un método de la otra.

- Usted puede ocultar (u omitir) detalles no esenciales.

UML: Diagrama de Clases del Sistema (5/5)

Ejemplo de tipos de relaciones entre clases

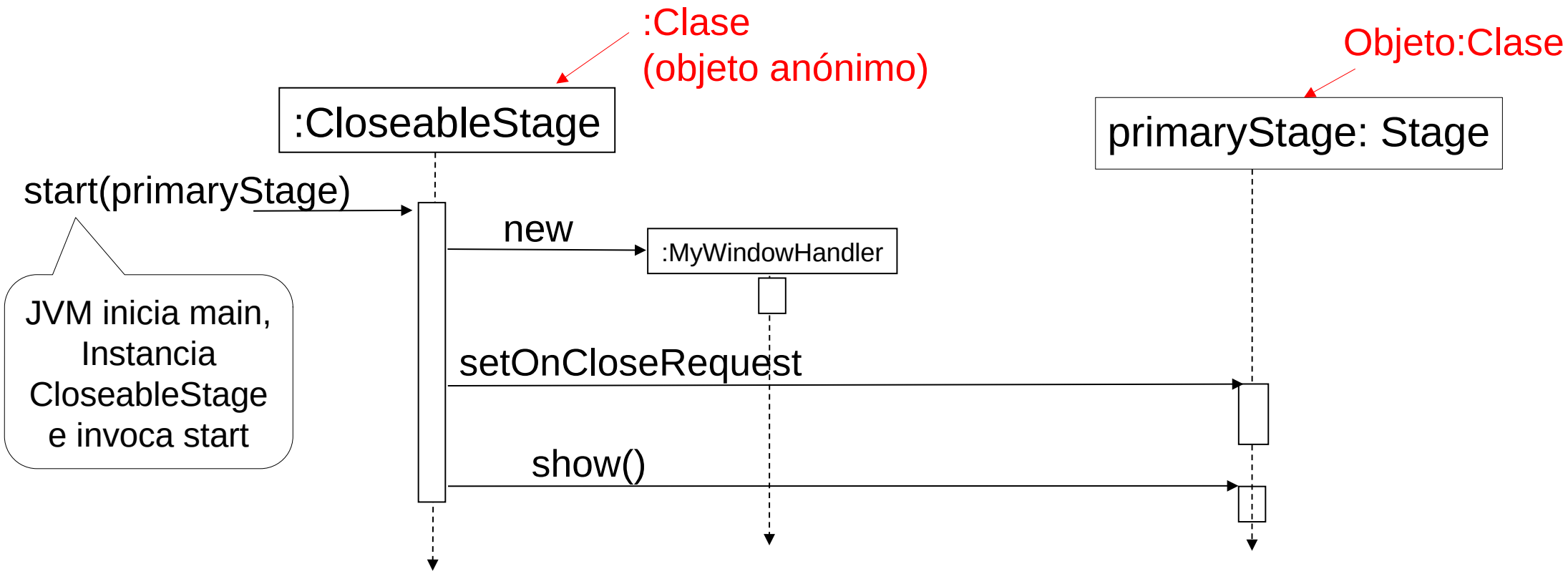


Fuente: Class Diagram Example: GUI (https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/)

UML: Diagrama de Secuencia (1/2)

Modelan la naturaleza dinámica de un sistema de software

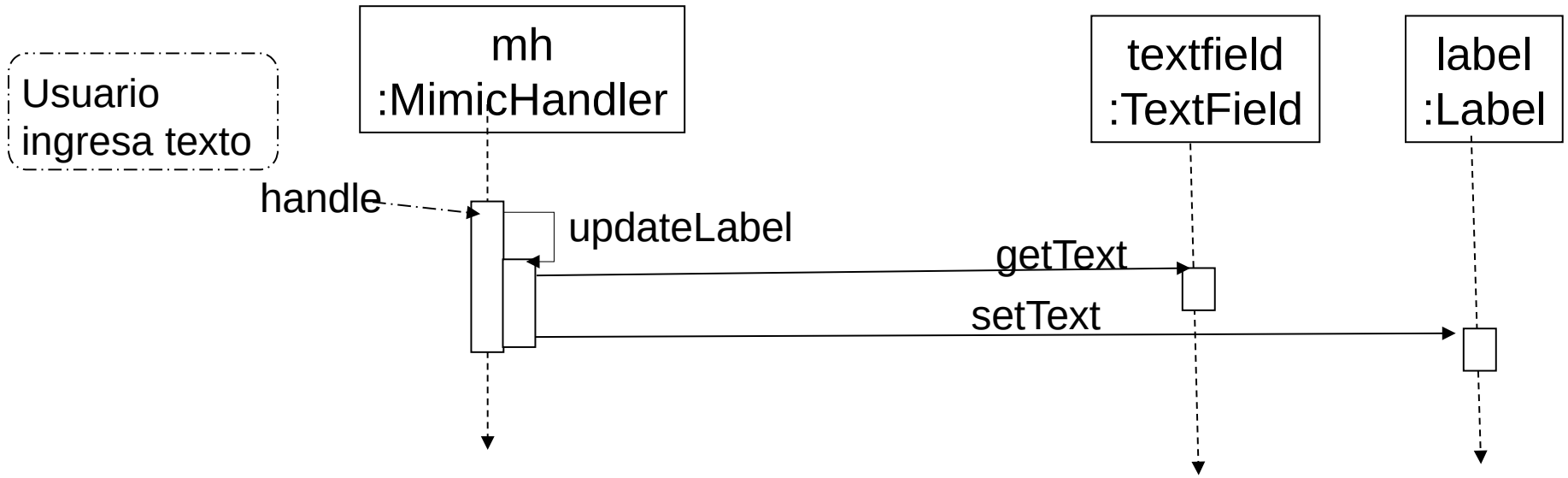
- Cada diagrama muestra la dinámica de un escenario:



- Ayuda a visualizar objetos y sus relaciones.

UML: Diagrama de Secuencia (2/2)

- Caso de uso: Ingreso de nuevo texto. Esto gatilla el evento esperado.



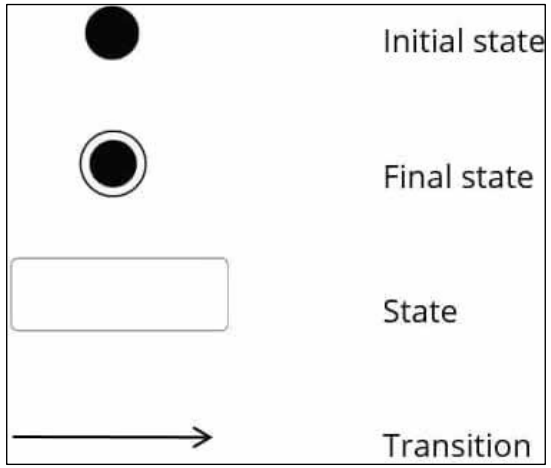
- Es un tipo de diagrama de interacción que muestra la comunicación entre objetos con respecto al tiempo.

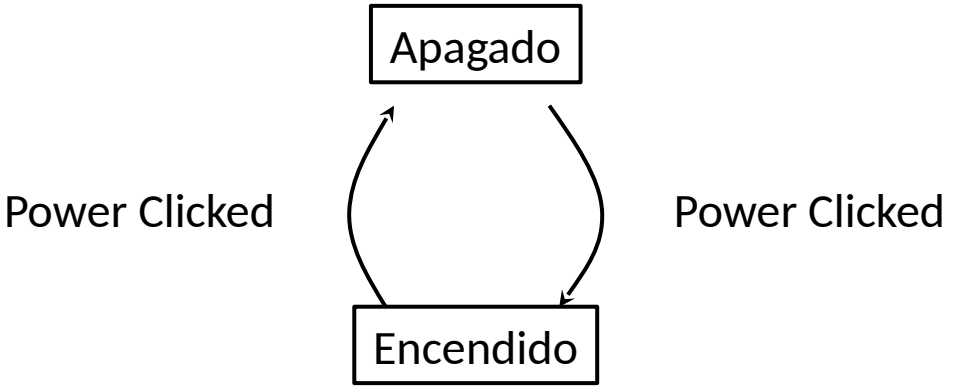
UML: Diagrama de Estado (1/2)

Modelar los estados de un objeto y los eventos que provocan cambios en esos estados

- Muestra los estados por los cuales puede pasar una entidad u objeto en el transcurso del tiempo.
- Son utilizados en las clases cuyos objetos tiene estados de interés.
- Similares a los diagrama de estados que verán o vieron en “Sistemas Digitales”.

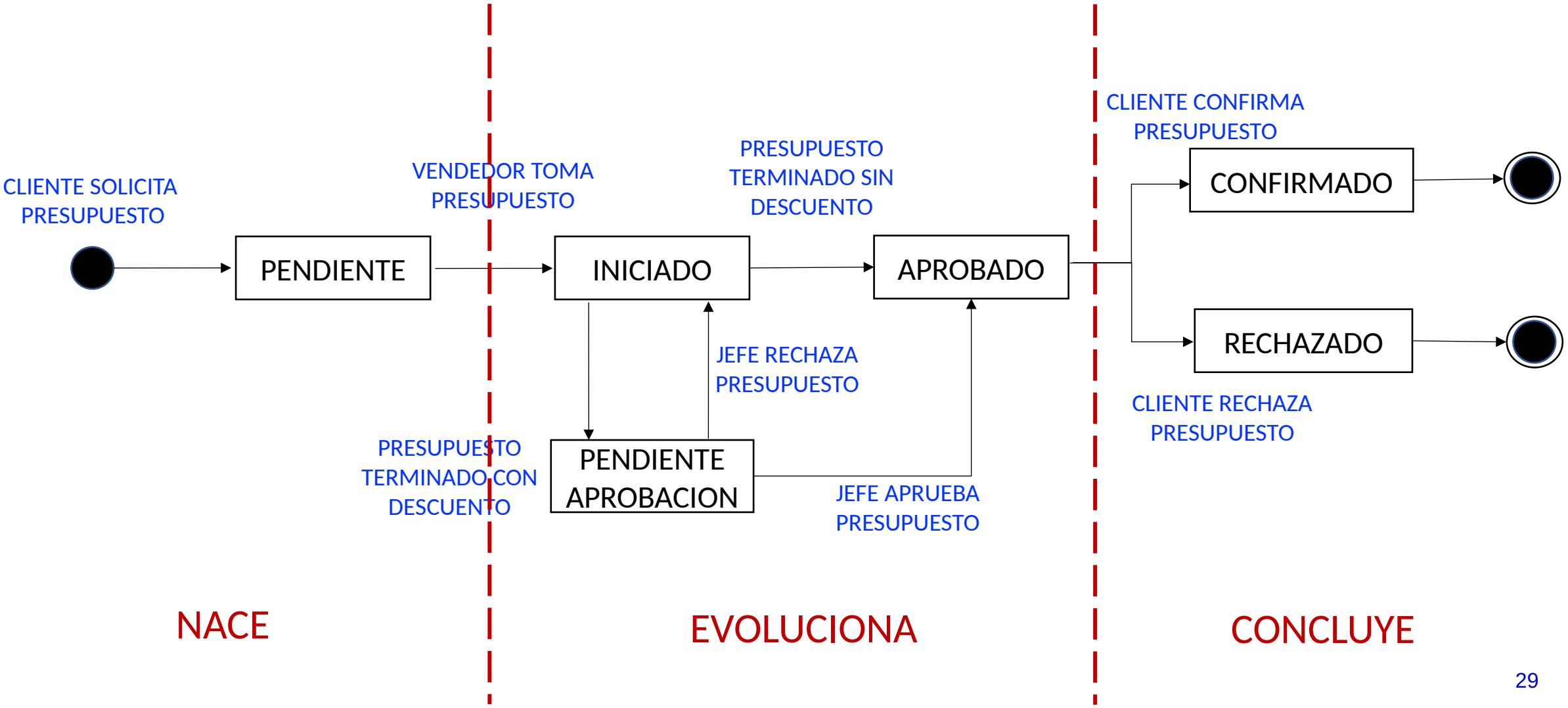
Notación





UML: Diagrama de Estado (2/2)

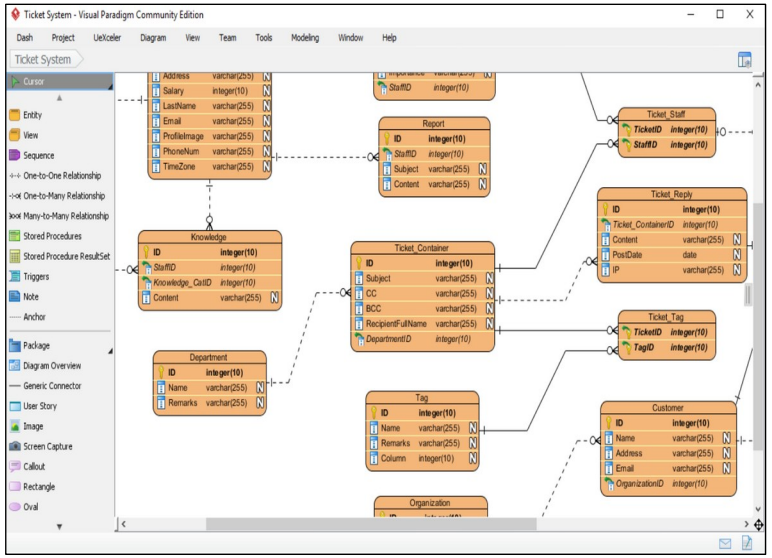
Ejemplo: Sistema online de solicitud de presupuesto para mantenimiento de edificios



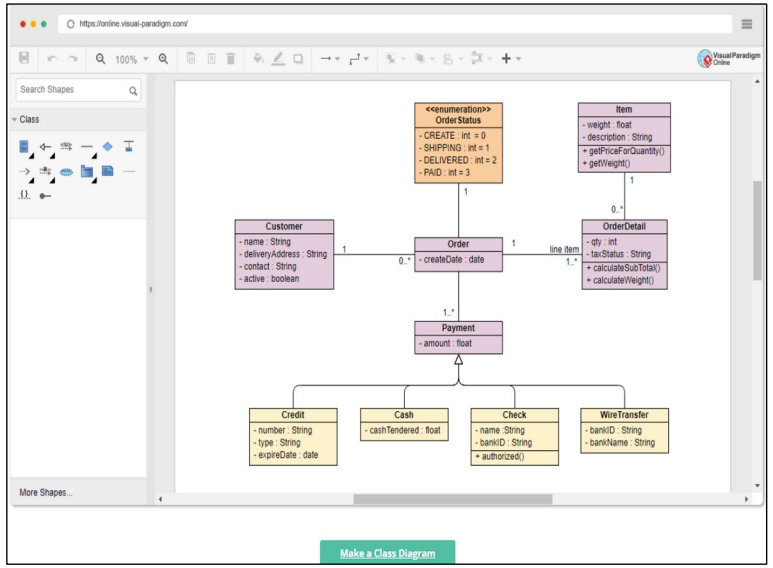
UML: Herramientas para SDLC

Ejemplo de Herramienta para crear diagramas de Clases

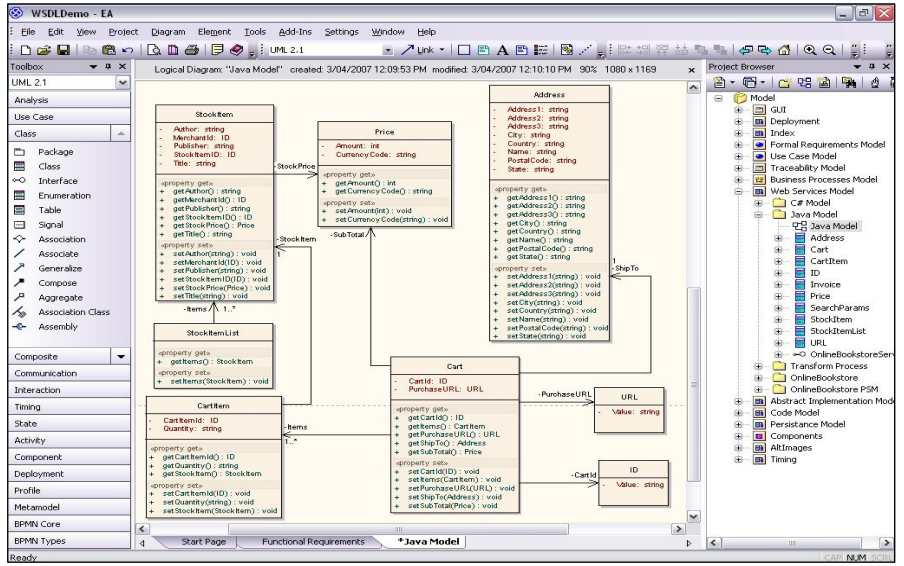
Visual Paradigm



Visual Paradigm Online



ENTERPRISE ARCHITECT



Visual Paradigm (desktop version) (https://www.visual-paradigm.com/download/)

Visual Paradigm (online version) (https://online.visual-paradigm.com/diagrams/solutions/free-class-diagram-tool/)

Enterprise Architect (desktop version) (https://sparxsystems.com/products/ea/)

Microsoft Project (https://www.microsoft.com/en-ww/microsoft-365/project/agile-methodology)

StarUML (https://staruml.io/)

IntelliJ (https://www.jetbrains.com/help/idea/class-diagram.html)

Jgrasp (https://www.jgrasp.org/)

Qt creator (https://wiki.qt.io/ModelEditor)

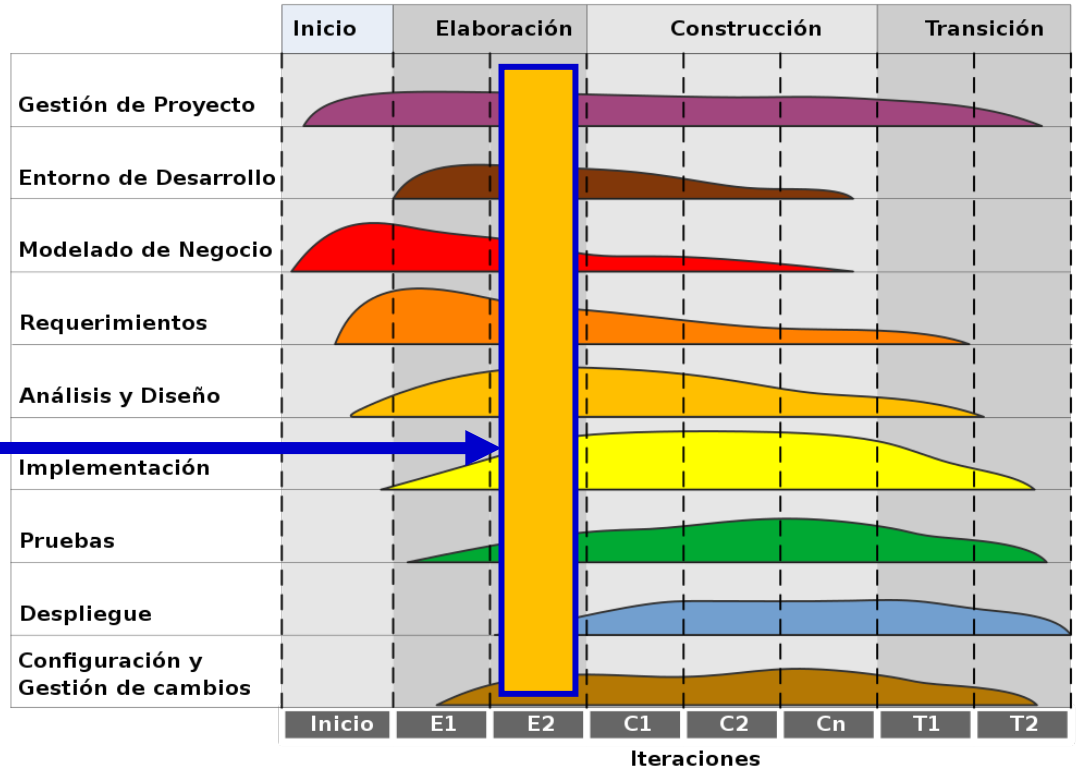
PlantUML (https://plantuml.com/)

Procedimiento en una Metodología Tradicional (1/2)

SDLC y Rational Unified Process Methodology (RUP)

Una iteración en la fase de elaboración

Una iteración en la fase de elaboración



15%
10%
10%
10%
10%
15%
10%
10%
10%

Duración:	5%	20%	65%		10%
Esfuerzo:	10%	30%	50%		10%

Procedimiento en una Metodología Tradicional (2/2)

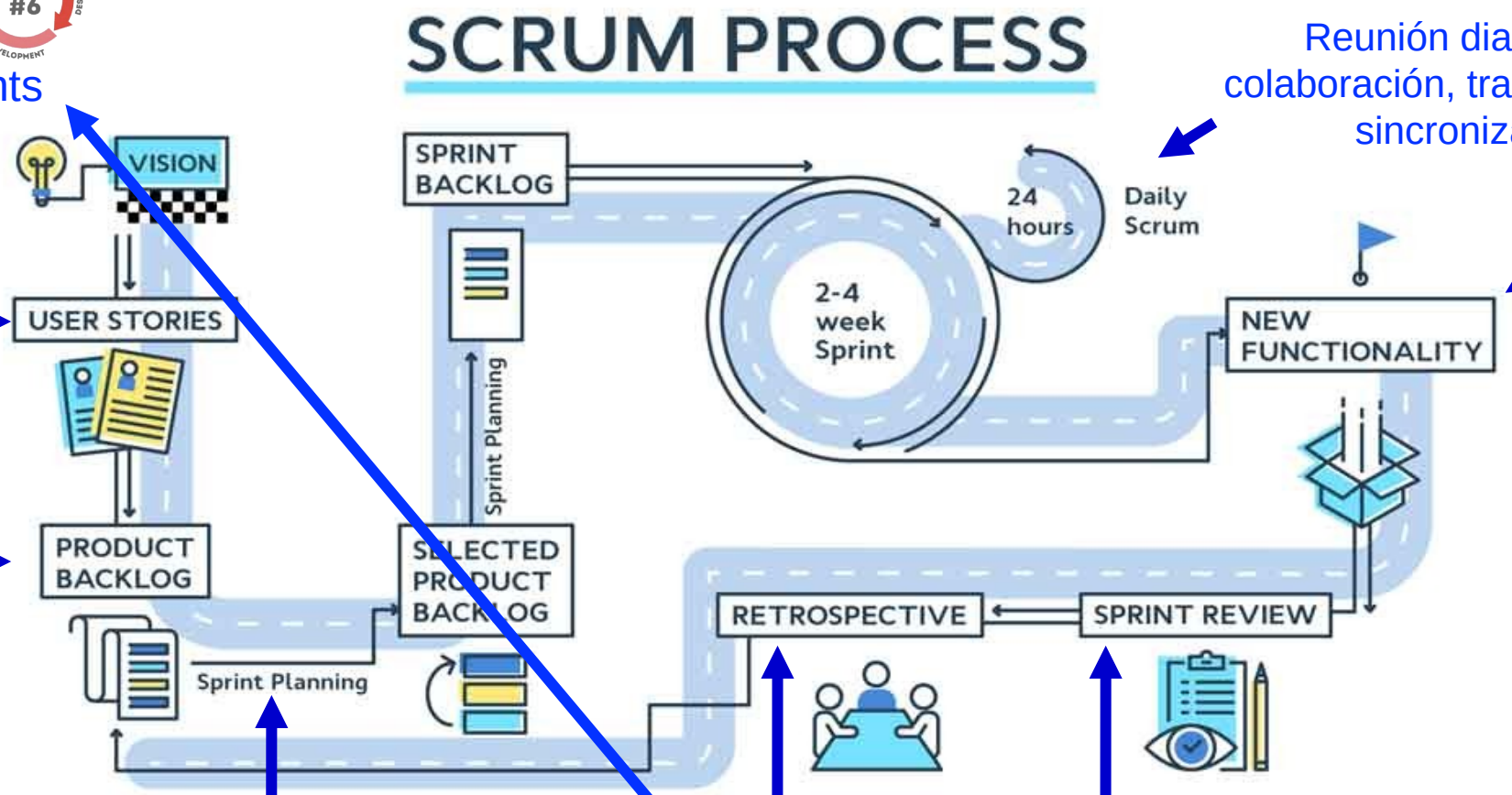
Ejemplo: Rational Unified Proces Methodology (RUP)

Fuente: https://www.phinet.cl/

Procedimiento en una Metodología Ágil (1/2)

Siguientes Sprints

SDLC y SCRUM



Capturar y describir los requisitos funcionales

Lista prioritizada de elementos de trabajo pendientes para un proyecto o producto

Reunión diaria corta colaboración, transparencia y sincronización

Características adicionales para el producto

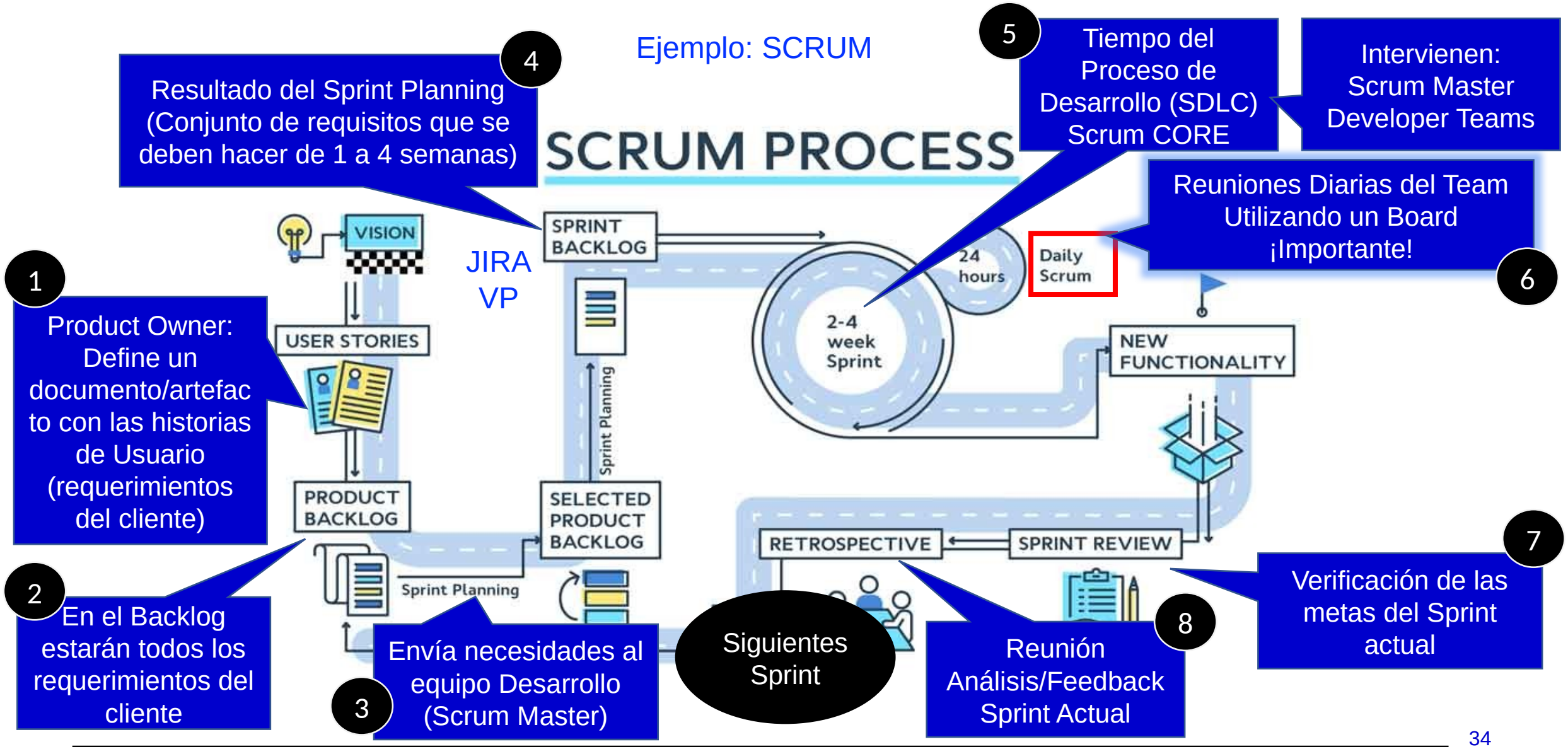
Selección de elementos del backlog que serán desarrollados

Reunión que se lleva a cabo al final de cada Sprint

Fuente: https://www.scrum.org/resources/what-scrum-module

Procedimiento en una Metodología Ágil (2/2)

Ejemplo: SCRUM



1 Product Owner: Define un documento/artefacto con las historias de Usuario (requerimientos del cliente)

2 En el Backlog estarán todos los requerimientos del cliente

3 Envía necesidades al equipo Desarrollo (Scrum Master)

4 Resultado del Sprint Planning (Conjunto de requisitos que se deben hacer de 1 a 4 semanas)

5 Tiempo del Proceso de Desarrollo (SDLC) Scrum CORE

Intervienen: Scrum Master Developer Teams

6 Reuniones Diarias del Team Utilizando un Board ¡Importante!

7 Verificación de las metas del Sprint actual

8 Reunión Análisis/Feedback Sprint Actual

Siguientes Sprint

Fuente: https://www.scrum.org/resources/what-scrum-module

Resumen

- Definición de Arquitectura y Arquitecto de Software.
- Artefactos: Diagramas UML del sistema (Casos de uso, Clases, Secuencia, Estado).
- Definición de Historias de Usuario. Diferencias vs Casos de uso.
- Herramientas de Gestión de Proyectos de Software: JIRA, Visual Paradigm, JetBrains, otras.

Lección Importante

- El tiempo es independiente del contexto: “ahorrar una semana la comienzo de un proyecto es tan bueno como ahorrarla al final”. “Una semana es una semana”.
- Es mucho más fácil ahorrar tiempo al inicio del proyecto (cuando los “entregables” son menos claros).

¿Cómo usted aplica esto al proyecto de la asignatura?

¿Cuánto desearía tener algunos días “libres” antes del plazo?

Fin