

Curso de Ingeniería de Software

## Unidad 2

¿Cómo y con qué vamos a trabajar?

Guadalupe Ibargüengoitia G.

Hanna Oktaba

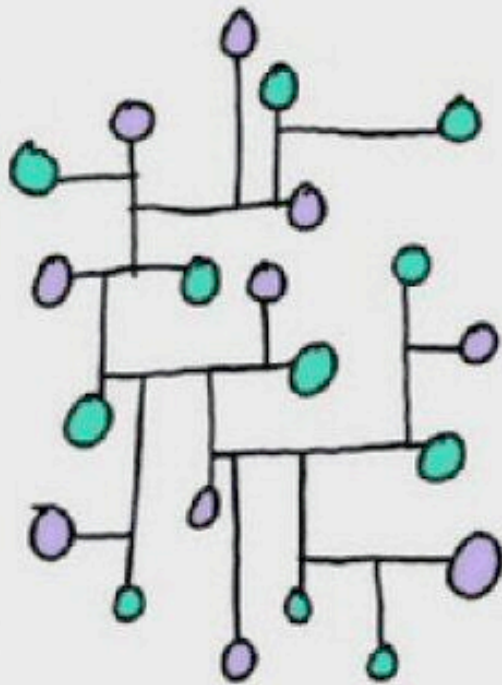
# Objetivos

- Conocer los **estándares** de Ingeniería de Software, que **fundamentan** el curso, así como su papel en el **diseño del método de desarrollo de software**.
- Conocer y aplicar **las prácticas básicas** de **Ingeniería de Software** mediante desarrollo de un **proyecto de software**, trabajando en **equipos**.

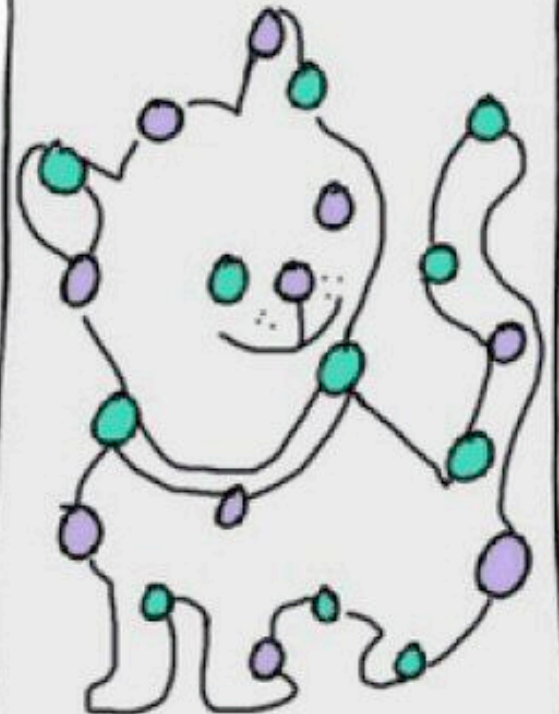
## Knowledge



## Experience



## Creativity



# ¿En qué nos basamos?

- Este curso está basado en 6 estándares de Ingeniería de Software.



## ISO/IEC 29110

Para los  
procesos de  
administración y  
desarrollo



## SCRUM

Para las  
prácticas  
ágiles



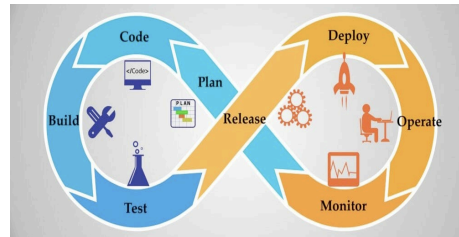
## SWEBOOK

Para las  
definiciones



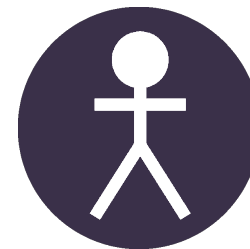
## KUALI-BEH

Para expresar el  
método y las  
prácticas del curso



## DevOps

Para integrar  
Desarrollo y  
Operación



## UML

Para  
modelar  
productos

# ¿Qué son los estándares?

- Los estándares se crean como **acuerdos** entre **grupos de personas, empresas, organismos o países** para **resolver o aminorar** algún **problema** en común.

# ¿De qué sirven los estándares?

- Inducir **reglas de comportamiento** para el **bien de las comunidades**, como por ejemplo, el reglamento de tránsito (semáforos, pasos peatonales, multas, ...), recomendaciones sanitarias en pandemia.

# ¿De qué sirven los estándares?





# ¿De qué sirven los estándares?

- **Lograr** ciertos **objetivos siguiendo formas de trabajar sugeridas (buenas prácticas)**, como, por ejemplo, las recetas de cocina.

# Sándwich de jamón y queso

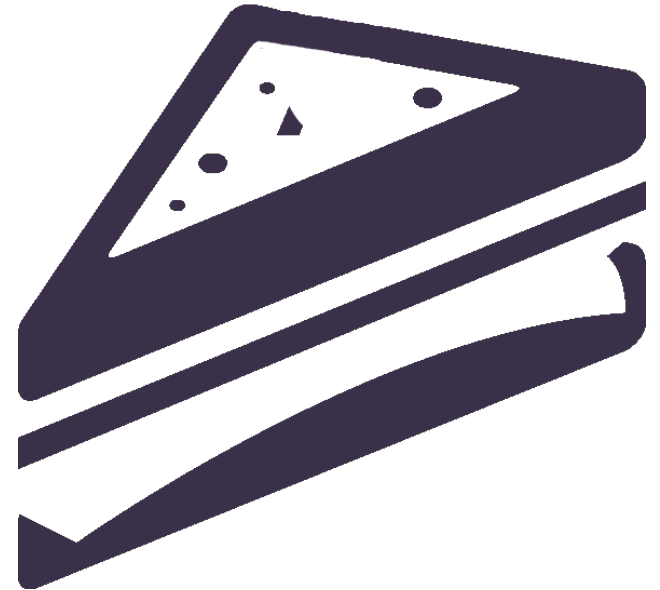
Porciones:1

Tiempo de preparación: 2 minutos

## Ingredientes:

- 2 hojas de lechuga
- 3 rebanadas de pan de caja
- 3 cucharadas de mayonesa
- 1 cucharada de mostaza
- 2 rebanadas de jamó ahumado
- 2 rebanadas de queso amarillo
- 2 cucharadas de aguacate machacado
- 2 rebanadas de tomate

- Pasos**
1. Revuelva las cucharadas de mayonesa con la mostaza.
  2. Unte las rebanadas de pan con la mayonesa y mostaza preparada.
  3. Acomode una rebanada de jamón, una de queso, una de lechuga, una de tomate y una cucharada de aguacate,
  4. Repita con la otra tapa de pan y finalice con la última rebanada de pan.



# ¿De qué sirven los estándares?

- **Generar productos**, como por ejemplo el estándar USB para conectar y alimentar los dispositivos electrónicos



# ¿Cómo se nombran?

- Los estándares tienen distintos nombres:  
**normas, procesos, modelos, reglamentos o protocolos ...**



## ISO/IEC 29110

Para los  
procesos de  
administración y  
desarrollo



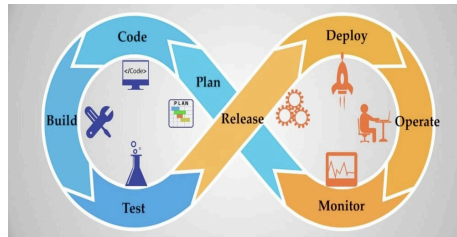
## SCRUM

Para las  
prácticas  
ágiles



## SWEBOOK

Para las  
definiciones



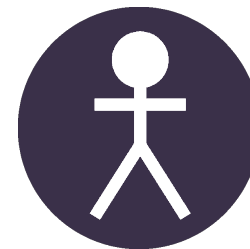
## DevOps

Para integrar  
Desarrollo y  
Operación



## KUALI-BEH

Para expresar el  
método y las  
prácticas del curso



## UML

Para  
modelar  
productos

# Qué es SWEBOOK 3.0



- SWEBOOK (Software Engineering Body of Knowledge) (SWEBOOK 3.0, 2014) es un cuerpo de conocimiento sobre la Ingeniería de Software recopilado por la comunidad de los profesionales y académicos, con el objetivo principal de integrar y sistematizar los contenidos de la Ingeniería del Software y hacerlos disponibles a la comunidad mundial.



*Guide to the Software  
Engineering Body of Knowledge*

**Editors**

Pierre Bourque  
Richard E. (Dick) Fairley



IEEE  computer society

# Para qué lo vamos a usar

- **SWEBOK** va a ser nuestra fuente básica de **definiciones de conceptos** de Ingeniería de Software.

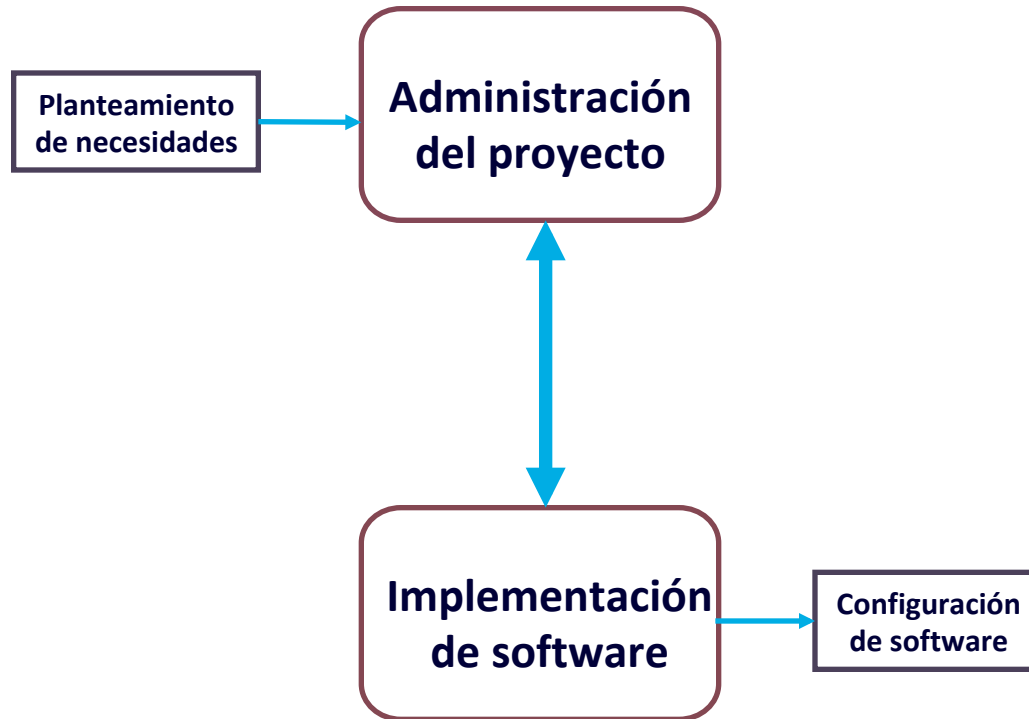


# Qué es ISO/IEC 29110 Perfil Básico



- ISO/IEC 29110 Perfil Básico (ISO/IEC 29110, 2011) es un estándar internacional, que contiene la descripción de **dos procesos: Administración de Proyecto e Implementación de Software**.
- Estos procesos recogen las **buenas prácticas** para el desarrollo de proyectos de software por organizaciones pequeñas de hasta 25 personas.
- Está basado en los procesos correspondientes de la **norma mexicana NMX-059-NYCE** que es el modelo de procesos **MoProSoft** (MoProSoft, 2005).

# Los dos procesos del Perfil Básico



# Administración de Proyecto

- Tiene el propósito de **organizar el trabajo** del equipo de desarrollo y vigilar el cumplimiento de las **necesidades del cliente**.
- Contiene las **actividades** que permiten **guiar la colaboración** del equipo involucrado en el proceso de Implementación.

# Implementación de Software (1)

- Se empieza por **entender las necesidades** del cliente, **analizar** estas necesidades y **expresarlas** en forma de **requerimientos** para el software.
- Una vez entendidos los **requerimientos** hay que idear el **diseño arquitectónico** de los **componentes** del sistema, sus **relaciones** y, luego **detallarlos** asegurándose que se cumplan los **requerimientos**.

# Implementación de Software (2)

- A partir del **diseño** se **construyen** (programan) los **componentes de software** y se **prueban** de manera **individual**.
- Las siguientes actividades son la **integración** de los **componentes** y la **prueba del sistema integrado**, además se realiza la **corrección de defectos** encontrados.
- Finalmente, **se entrega** el **software** funcional acompañado de la **documentación**, que ayuda al **cliente a operarlo y a comprenderlo** para las futuras **modificaciones**.

# Para qué lo vamos a usar

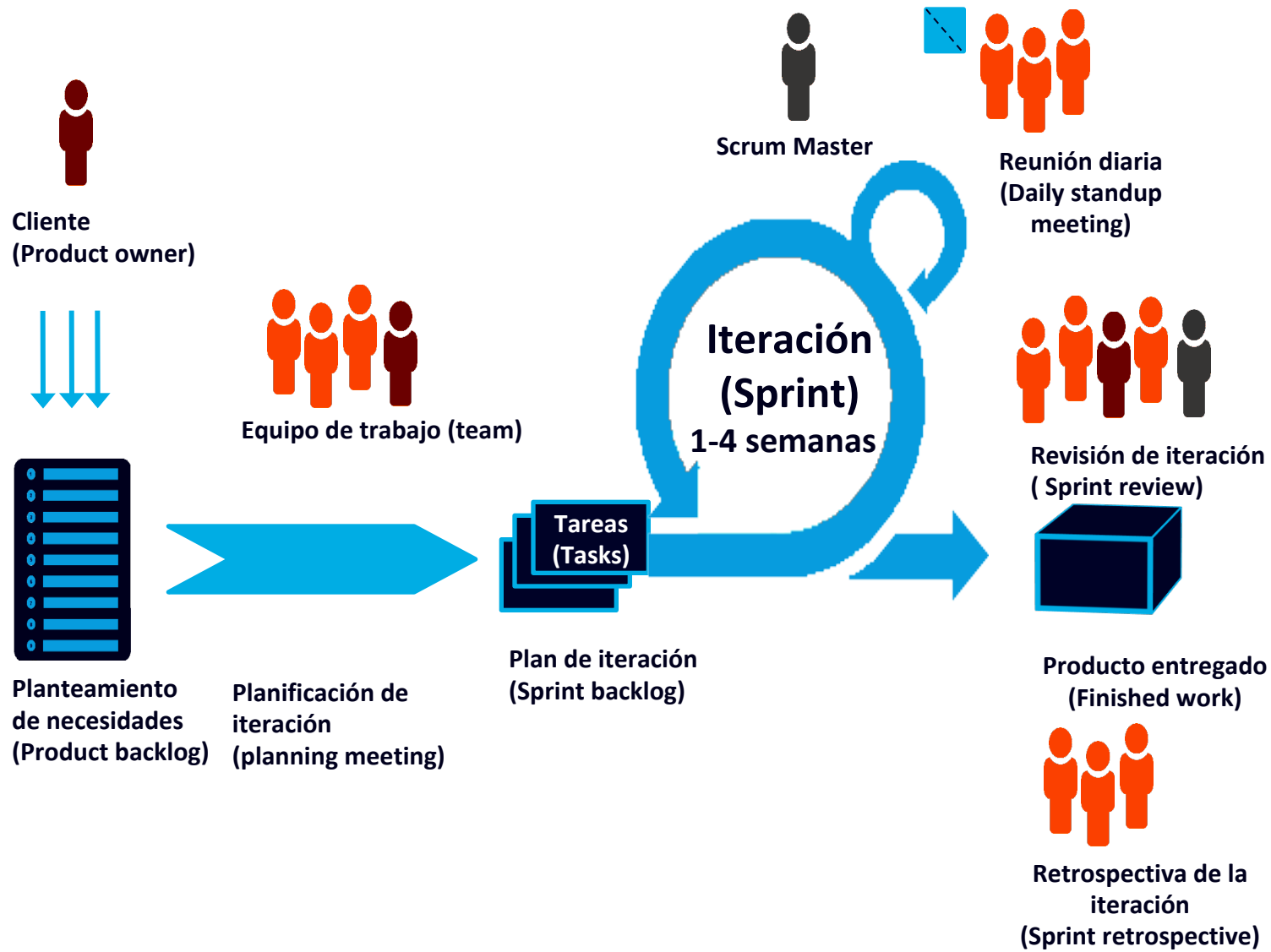
- Los **procesos** del **Perfil Básico** nos servirán para la **definición** de las **prácticas administrativas y de desarrollo del método** del curso.

# Qué es SCRUM



- Es un **marco de trabajo ágil** que define un conjunto de prácticas y roles para **organizar** el trabajo de desarrollo de software en **equipos**.

# Conceptos más importantes de SCRUM

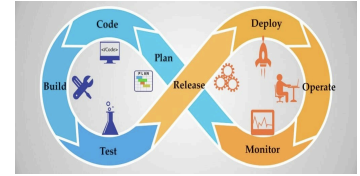




# Para qué lo vamos a usar

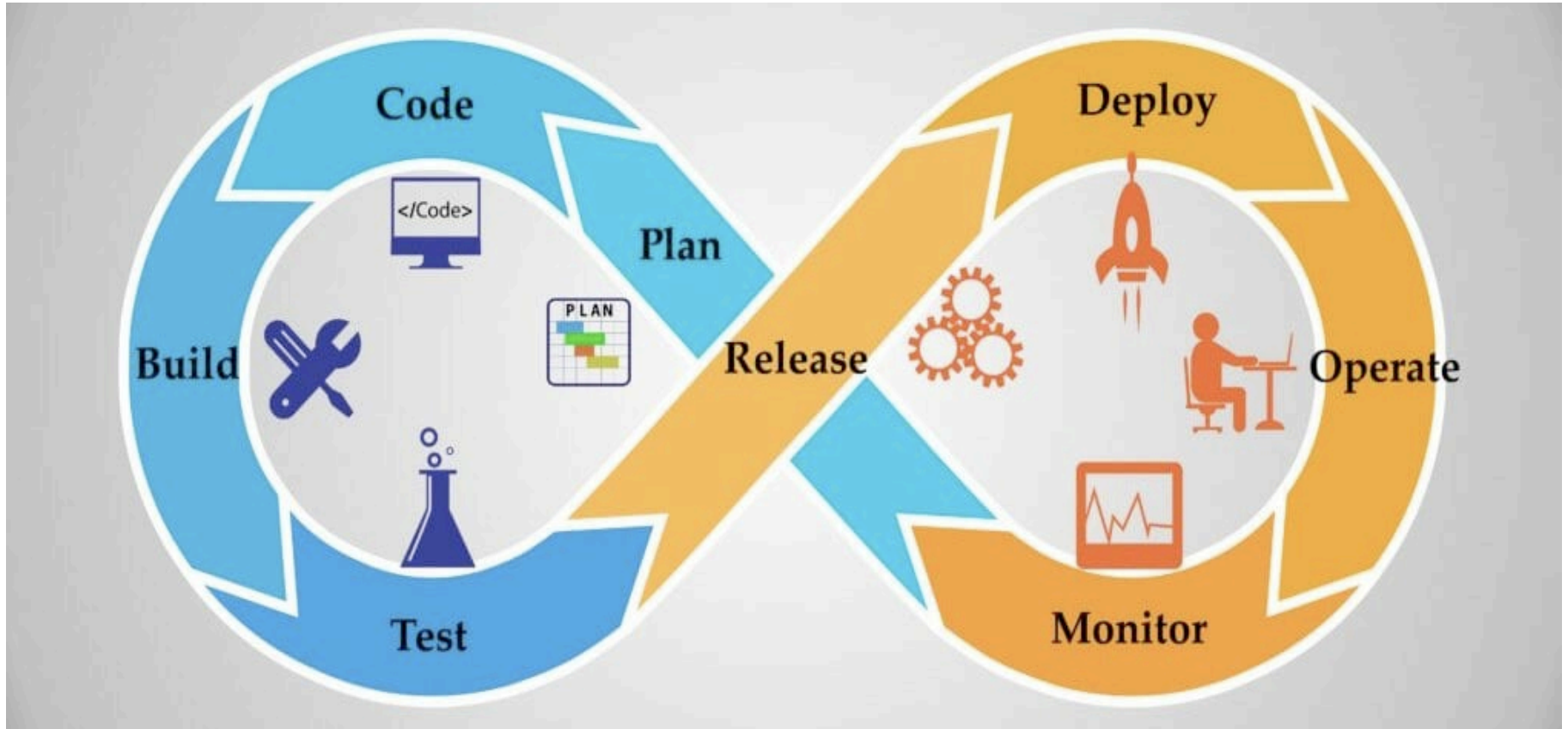
- El **SCRUM** lo usaremos para la definición de algunas técnicas sociales y administrativas del curso.

# DevOps



- Agrupa un conjunto de **prácticas de codificación, integración, prueba, despliegue, monitoreo y recuperación** de software, apoyadas con **herramientas que permiten su automatización**, con el objetivo de **optimizar** y hacer mas **eficientes** las actividades de **desarrollo y operación** de software.

# DevOps



<https://www.information-age.com/automating-six-cs-devops-123488662/>

# Para qué lo vamos a usar

- El **DevOps** lo usaremos para **refinar las prácticas de desarrollo** e introducir **herramientas para la automatización** de ciertas actividades.

# Qué es KUALI-BEH



- **KUALI-BEH** es una propuesta de **definición del concepto de método y práctica** para Ingeniería de Software, desarrollada **en México**, que forma parte del estándar **Essence – Kernel and Language for Software Engineering Methods** (OMG, 2013).

# KUALI-BEH (1)



- Un **método** es “*la articulación de un **conjunto** coherente, consistente y completo de **prácticas** que tiene el **propósito** de satisfacer las **necesidades del cliente**, bajo condiciones específicas*” (OMG, 2013).

# KUALI-BEH (2)



- Una **práctica** “proporciona una *guía del trabajo* a realizar, tiene un **objetivo** específico y proporciona una orientación de cómo producir un **resultado** a partir de una **entrada**. La guía ofrece un conjunto de **actividades** sistemáticas y repetibles, enfocadas a lograr el objetivo de la práctica y su resultado. Se requieren **habilidades y conocimientos** particulares para seguir la guía de la práctica y las **herramientas** pueden facilitar su realización” (OMG, 2013).

# Para qué lo vamos a usar

- El **KUALI-BEH** lo usaremos para la documentación del método y de las prácticas del curso.

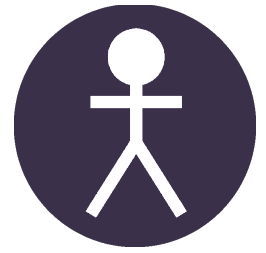




# Qué es UML (1)

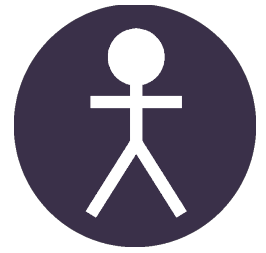
- Lenguaje de Modelado Unificado **UML** tiene como propósito **visualizar, especificar, documentar** y facilitar la **construcción y mantenimiento** de **software**. (Rumbaugh J. et al, 1998)

# UML (2)



- UML usa **símbolos gráficos** para representar los **conceptos** y **sus relaciones** siguiendo una sintaxis de lenguaje gráfico.
- Ofrece a los desarrolladores **varios tipos de diagramas** para modelar **diferentes aspectos** y etapas del desarrollo de software .
- Estos diagramas están clasificados en diagramas de **estructura** y del **comportamiento**.

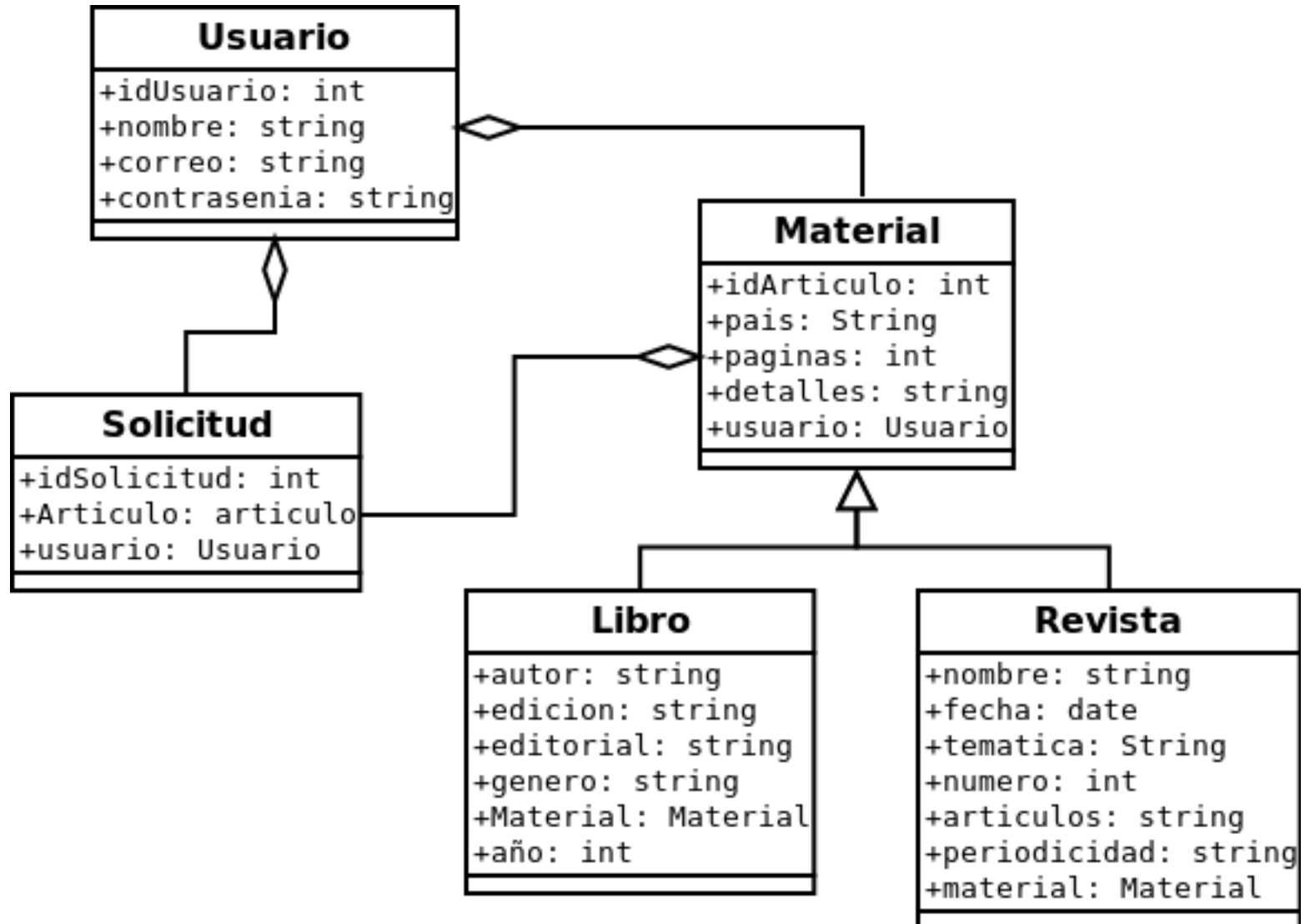
# UML (3)



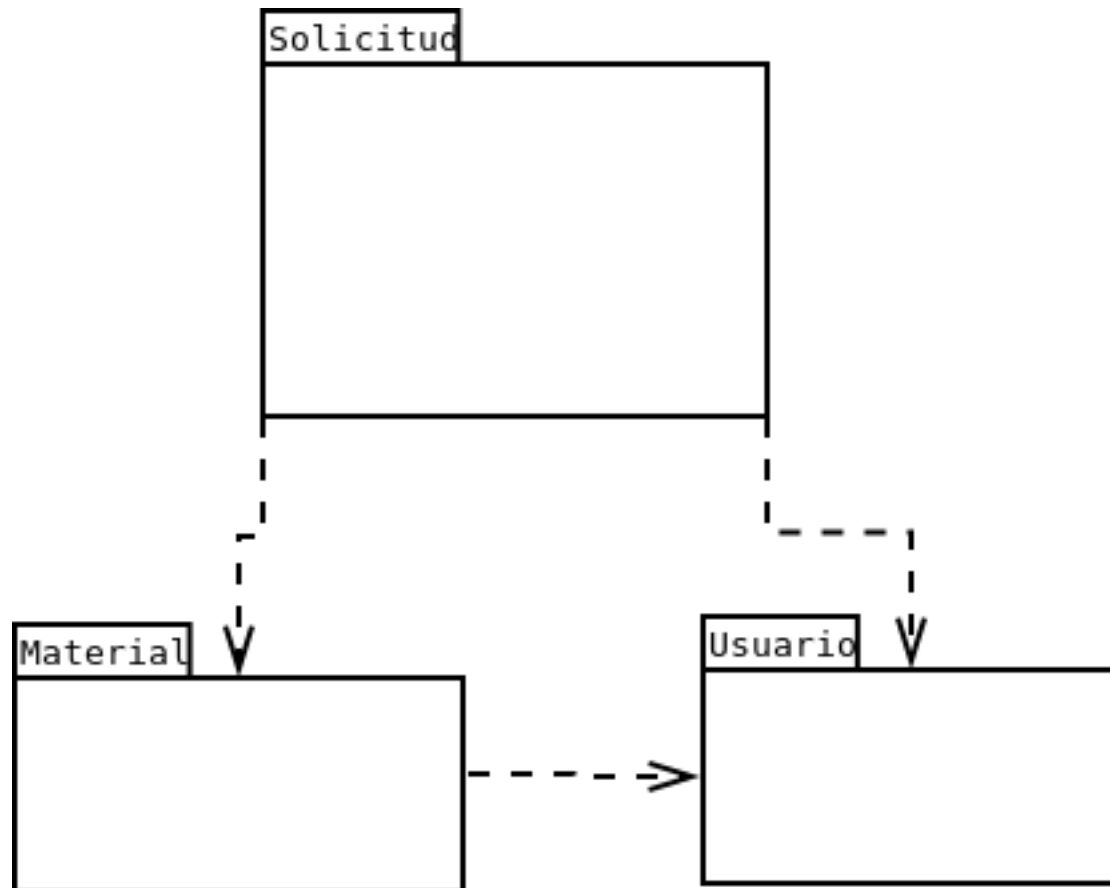
**Diagramas de estructura** muestran los elementos de los que se **compone el software** y sus **relaciones** en diferentes niveles de abstracción:

- **Clases** - son bloques de construcción básicos (Diagrama de clases)
- **Paquetes** - es un mecanismo para agrupar a los diagramas de clases en una estructura más abstracta. (Diagrama de paquetes)
- **Distribución** - sirven para visualizar como el software va a ser **distribuido entre los elementos físicos** (hardware) del sistema de software. (Diagrama de distribución)

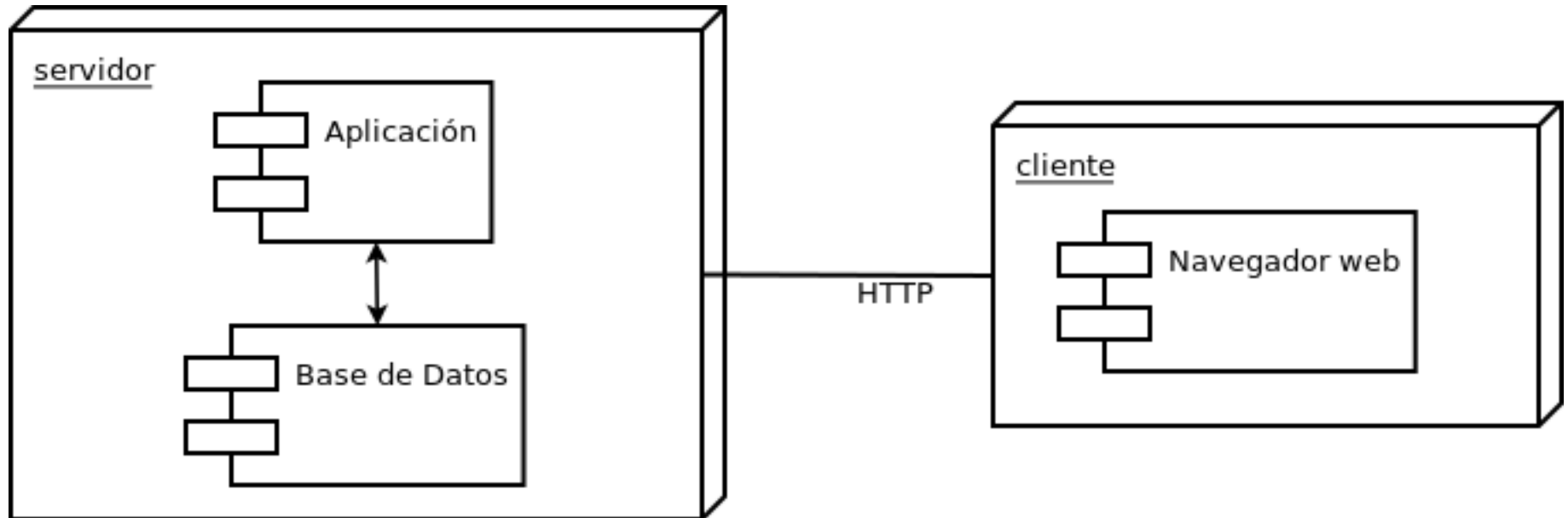
# Diagrama de clases



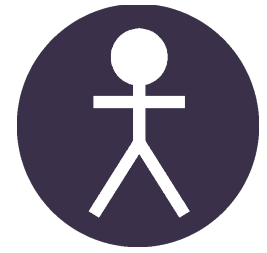
# Diagrama de paquetes



# Diagrama de distribución



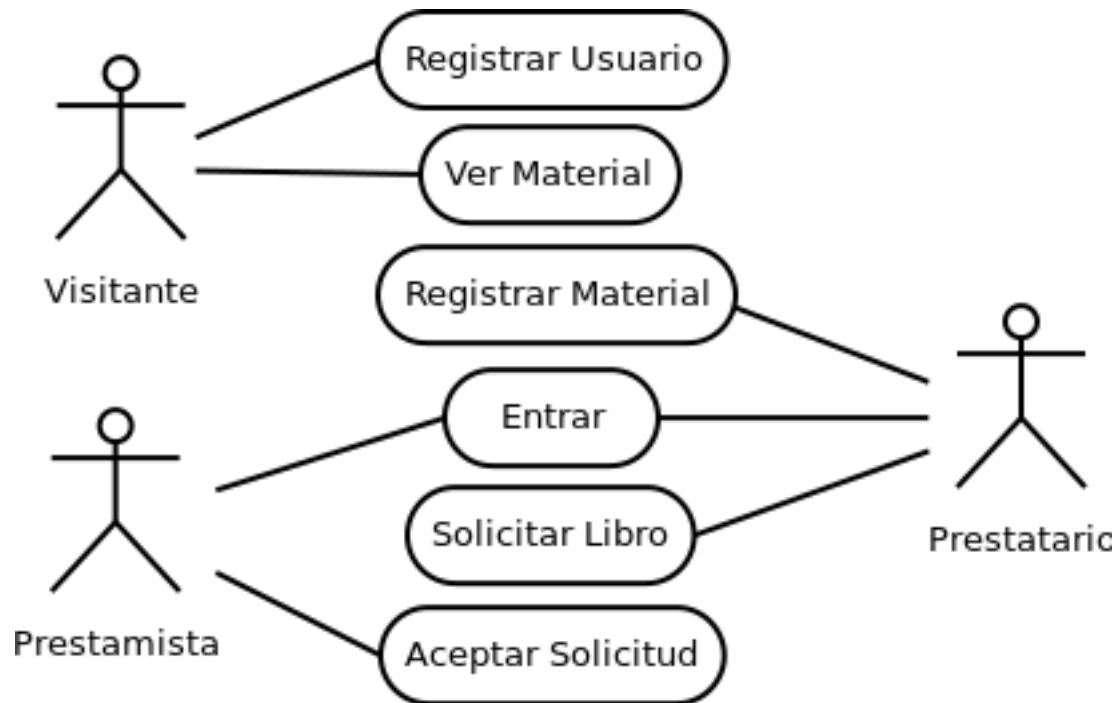
# UML (4)



**Diagramas del comportamiento** muestran como interaccionan estos elementos a lo largo del tiempo para proporcionar una funcionalidad.

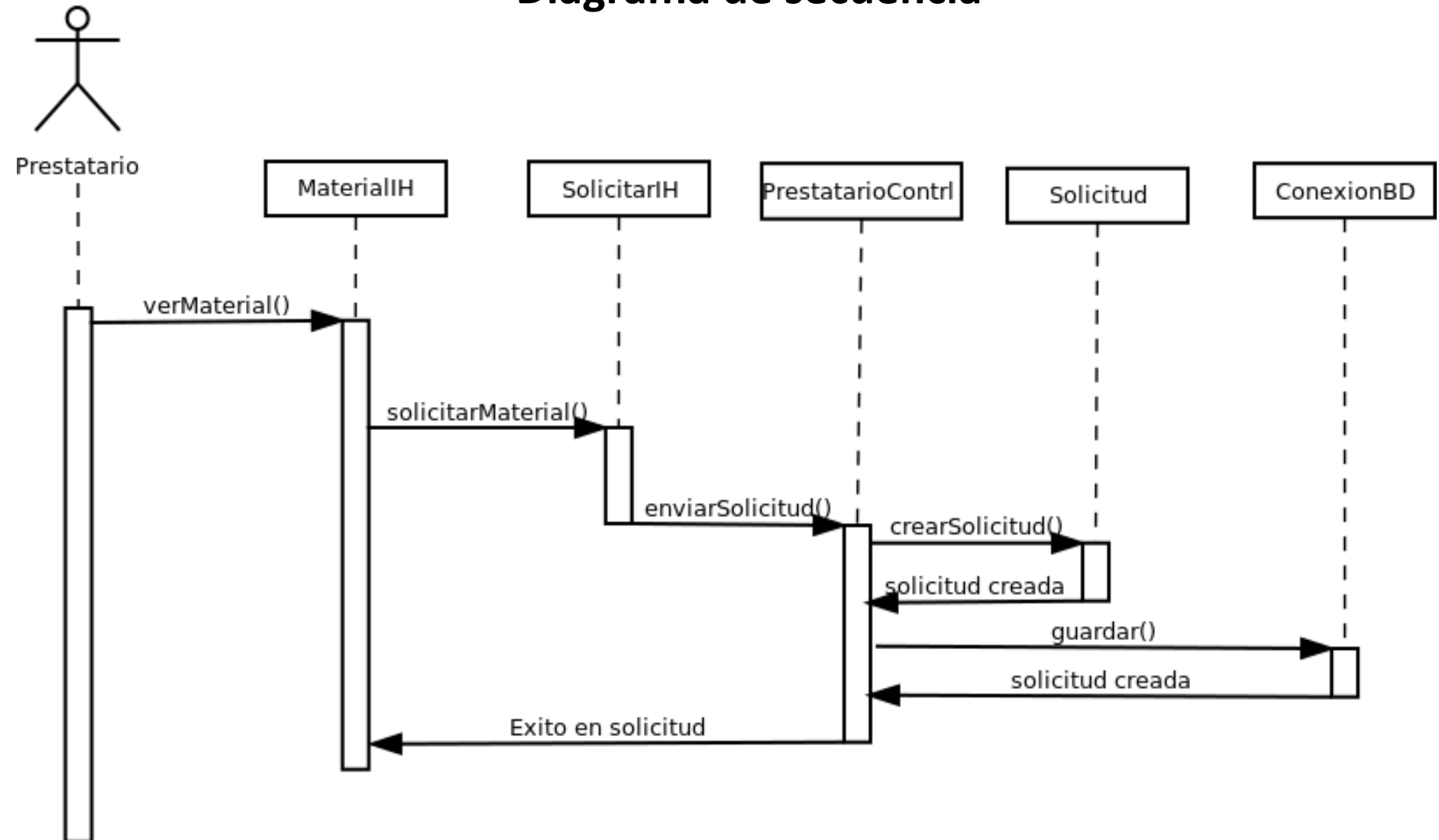
- **Casos de uso** - especifican el resultado de la interacción entre un actor (usuario u otro sistema) y el sistema de software mismo.  
(Diagrama de casos de uso)
- **Secuencia** - muestran la interacción de envío de mensajes entre los objetos de las clases del sistema.  
(Diagrama de secuencia)
- **Estados** - representan las máquinas de estados finitos que ayudan a modelar diferentes aspectos del comportamiento dinámico del sistema basado en cambio de estados a raíz de eventos.  
(Diagrama de estados o navegación)

## Diagrama general de casos de uso

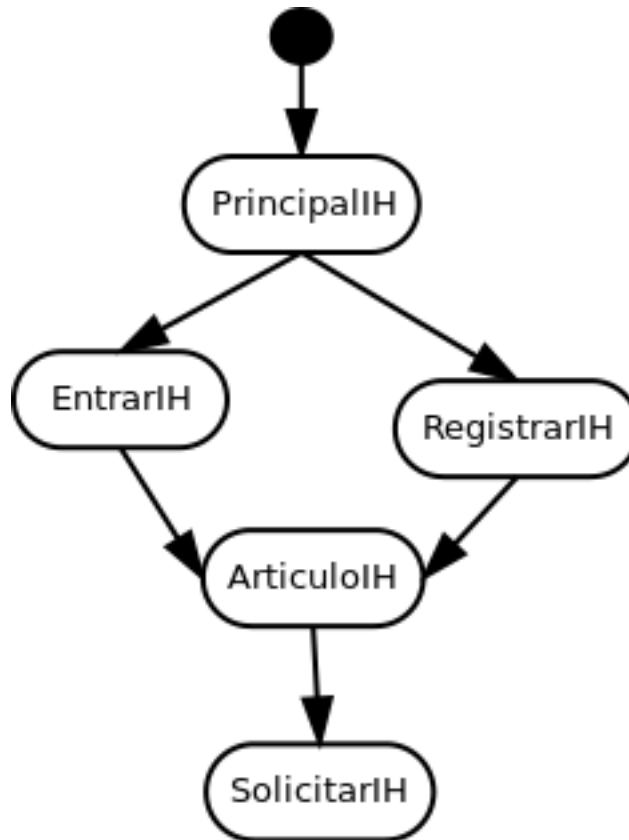




# Diagrama de secuencia



## Diagrama de estados (navegación)



# Para qué lo vamos a usar

- Los diagramas de **UML** vamos a usar para **modelar** diferentes **vistas de la estructura y del comportamiento** del software que se va a construir.

# ¿Qué hemos aprendido?

- ¿Cuáles son los **estándares** que vamos a usar en el curso y **de qué van a servir?**

# Método Inicial de Desarrollo de Software

## MIDS

- El método **MIDS** está definido como un conjunto de prácticas según **KUALI-BEH**.
- Las prácticas están basadas en el estándar **ISO/IEC 29110 Perfil Básico**, enriquecido con las prácticas de **SCRUM** y **DevOps**.
- Se **incluyen las prácticas sociales** para facilitar el trabajo en equipo.
- Los conceptos teóricos provienen de **SWEBOK**.
- Las **técnicas de modelado** para los **requerimientos funcionales y el diseño de software** proponen el uso de **UML**.

# Método Inicial de Desarrollo de Software MIDS

## Propósito

Los **alumnos** conocerán y aplicarán **las prácticas básicas de Ingeniería de Software** mediante desarrollo de un **proyecto de software**, trabajando en **equipos**.

# Método Inicial de Desarrollo de Software

## MIDS

### Entrada

- **Condiciones**

Los alumnos tienen los conocimientos de programación y bases de datos suficientes para desarrollar un producto de software.

- **Productos de trabajo**

- *Planteamiento de Necesidades*
- *Guión del Curso*
- *Plantillas*

# Prácticas del MIDS (1)

- **Prácticas Sociales**

- **PS1. Conformar el equipo y su ambiente de trabajo**
- **PS2. Reunión diaria**
- **PS3. Reunión de toma de decisiones**
- **PS4. Retrospectiva de la iteración**



# Prácticas del MIDS (2)

- **Prácticas Administrativas:** basadas en el proceso de **Administración de Proyecto** de la **ISO/IEC 29110**, enriquecidas con **SCRUM**.
  - PA1. Planificar el proyecto
  - PA2. Planificar la iteración
  - PA3. Ejecutar el plan de iteración
  - PA4. Evaluar y controlar la iteración
  - PA5. Cerrar la iteración
  - PA6. Cerrar el proyecto

# Prácticas del MIDS (3)

- **Prácticas para el Desarrollo:** basadas en el proceso de **Implementación de Software** de la **ISO/IEC 29110** enriquecidas con **DevOps**.
  - PD1. Requerimientos de software
  - PD2. Diseño de software
  - PD3. Construcción de software
  - PD4. Integración y pruebas de software
  - PD5. Despliegue de software
  - PD6. Prueba de entrega de software

# Método Inicial de Desarrollo de Software

## MIDS

### Resultado

- **Condiciones**

Los alumnos han practicado conceptos básicos de Ingeniería de Software desarrollando un proyecto de software en equipos.

- **Productos de trabajo**

Software, que responde al planteamiento de necesidades, puesto en operación, con su documentación correspondiente incluida.

# Apoyos al MIDS

- *Presentaciones*
- *Plantillas*
- *Guión del curso*
- *Planteamiento de Necesidades*

# Roles para miembros del equipo

- Para seguir el **MIDS** se formarán **equipos de 4-5** personas.
- Los miembros del equipo tendrán que **ejecutar actividades** de las **prácticas**, a veces **en equipo** y a veces de manera **individual**.
- Para saber **quién va a ser responsable por ejecutar qué actividad**, vamos a definir **cinco roles** que se asignarán a los miembros del equipo.
- Cada **rol** tiene definido su **objetivo**, las **habilidades** que se requieren para desempeñarlo y sus **responsabilidades**.

# Desarrollador



## Objetivo

- Participar en la ejecución de todas las prácticas del desarrollo de software.

## Habilidades

- Tener conocimientos y experiencia en programación, estructuras de datos y bases de datos.

## Responsabilidades

- Entender los requisitos del software, participar en la especificación de requerimientos, el diseño, construcción, integración, pruebas y despliegue del software.
- Participar en las reuniones de trabajo.
- Revisar y corregir los productos de los que sea responsable.
- Aplicar los estándares solicitados o acordados.

# Responsable del equipo



## Objetivo

- Mantener motivados a todos los miembros del equipo para que participen activamente en el proyecto y trabajen en armonía.

## Habilidades

- Sociable y amistoso.
- Es un líder natural en los grupos.
- No conflictivo, sabe motivar a los demás.
- No es impositivo sino conciliador, pero intolerante a faltas de compromiso.

## Responsabilidades

- Construye y mantiene la cohesión del equipo y su efectividad.
- Motiva a los miembros a trabajar en equipo y cumplir sus compromisos.
- Ayuda a resolver los conflictos que se presenten en el equipo. Si no lo logra, los comunica a los docentes de manera oportuna para que ayuden a resolverlos.
- Convoca y coordina las reuniones del equipo.
- Coordina la retrospectiva de la iteración.

# Responsable técnico



## Objetivo

- Lograr que el software creado por el equipo sea de la mejor calidad técnica.

## Habilidades

- Experiencia en programación.
- Reconocimiento del equipo por sus habilidades técnicas.
- Conocimientos de lenguajes de programación, ambientes de programación y herramientas de apoyo.

## Responsabilidades

- Dirigir al equipo en la toma de decisiones en las actividades técnicas de desarrollo.
- Aprovechar al máximo las habilidades y los conocimientos en programación de los miembros del equipo.
- Ayudar a los miembros del equipo en la solución de problemas técnicos.
- Ayudar en la capacitación de los miembros del equipo en el uso de las herramientas.
- Coordinar la integración pruebas y el despliegue del código del equipo.



# Responsable de la calidad



## Objetivos

- Asegurar que se sigan los estándares establecidos en el MIDS para los productos de trabajo.

## Habilidades

- Persona ordenada e interesada en la calidad del software.
- Saber hacer buenas revisiones y pruebas a los productos.

## Responsabilidades

- Coordinar la integración de la documentación de todas las prácticas del MIDS con el apoyo de todo el equipo.
- Encontrar los defectos y vigilar a que se corrijan.
- No permitir que se hagan cambios no autorizados a productos ya aprobados.
- Coordinar las pruebas y revisiones del producto de software.

# Responsable de la colaboración



## Objetivo

- Apoyar el trabajo colaborativo del equipo mediante uso de herramientas para la comunicación, la coordinación del trabajo y los repositorios compartidos de documentación y del código.

## Habilidades

- Conocedor de redes sociales y repositorios compartidos.
- Entusiasmado en aprender herramientas colaborativas y explicar al equipo su uso.
- Disciplinado en el manejo de los repositorios comunes de documentación y del código para facilitar la comunicación asíncrona del equipo.

## Responsabilidades

- Seleccionar y/o aprender las herramientas necesarias para apoyar al equipo en la comunicación y coordinación.
- Ayudar en la creación y mantener los repositorios comunes de documentos y de código.
- Ayudar en la coordinación de la realización de las actividades de administración del proyecto manteniendo actualizado el contenido de la herramienta de coordinación según las actividades realizadas por el equipo.

# ¿Qué hemos aprendido?

- ¿Cuál es el propósito del **MIDS** y los **tipos de prácticas** que se van a seguir?
- ¿Qué **roles** van a jugar los **miembros de equipos**?

# Bibliografía

- Ambler S.W. (2005). *The Elements of UML 2.0 Style*. Cambridge University Press.
- ISO/IEC 29110. (2011). *29110-5-1-2Software Engineering-lifecycle Profiles for Very Small Entities Management and Engineering Guidde. s.1*. Software Engineering.
- Kim Gene, Jez Humble, Patrick Debois, John Willis (2016). *DevOps Handbook*. IT Revolution Press.
- OMG. (2018). *Essence – Kernel and Language for Software Engineering*. Annex B KUALI-BEH Kernel Extension. Obtenido de <https://www.omg.org/spec/Essence/About-Essence/>
- Rumbaugh J., I. Jacobson, G. Booch. (1998). *The Unified Modeling Language. Reference Manual*. Addison Wesley.
- Schwaber, K. Sutherland (2011). *The Scrum Guide – The Definitive Guide to Scrum: The Rules of the Game*. Obtenido de <https://www.scrum.org/resources/scrum-guide>
- SWEBOK. (2014). *Guide to the Software Engineering Body of Knowledge v3.0*. IEEE Computer Society.