Curso de Ingeniería de Software

Unidad 2
Cómo y con qué vamos a trabajar

Guadalupe Ibargüengoitia G.

Hanna Oktaba

Estándares a utilizar

- En esta unidad te presentaremos brevemente 5 estándares internacionales de Ingeniería de Software que servirán de guía para el trabajo de este curso.

¿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, …)

¿De qué sirven los estándares?

¿De qué sirven los estándares?

- Lograr ciertos objetivos siguiendo formas de trabajar sugeridas, como son las recetas de cocina (poner ejemplos de recetas)

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

SWEBOK

Para las definiciones

ISO/IEC 29110

Para los procesos

KUALI-BEH

Para expresar el método y las prácticas del curso

SCRUM

Para las prácticas ágiles

UML

Para modelar productos

SWEBOK 3.0

- SWEBOK (Software Engineering Body of Knowledge) (SWEBOK 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.

SWEBOK® V3.0

Guide to the Software Engineering Body of Knowledge

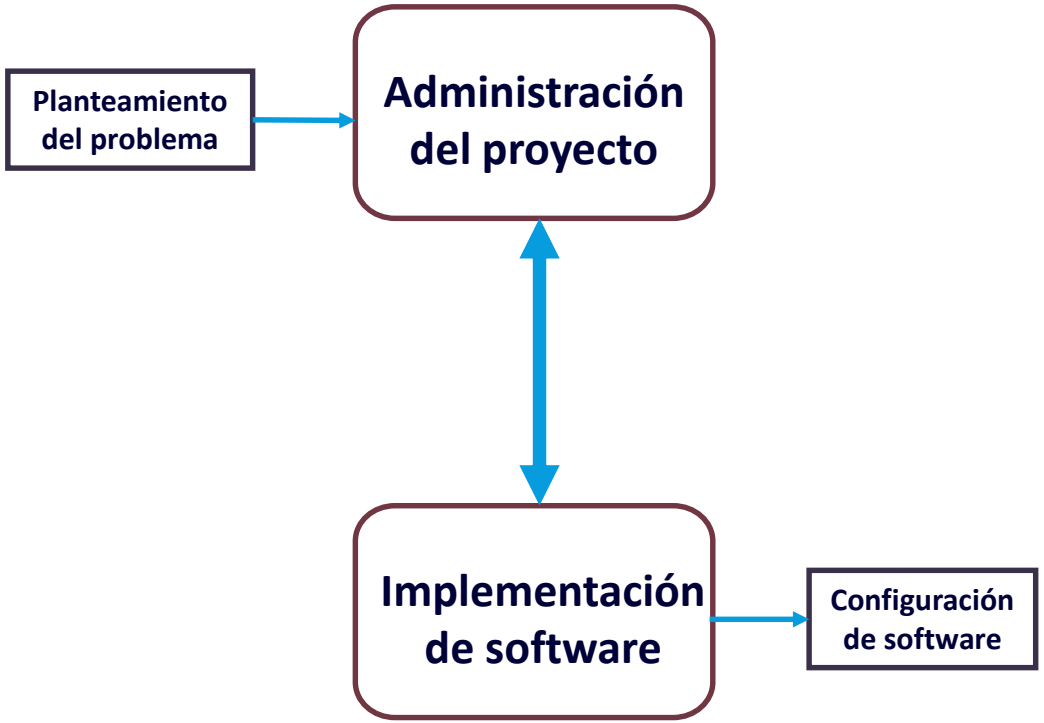
Editors

Pierre Bourque
Richard E. (Dick) Fairley

ISO/IEC 29110 Perfil Básico

- ISO/IEC 29110 Perfil Básico (ISO/IEC29110., 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 l**a **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**.

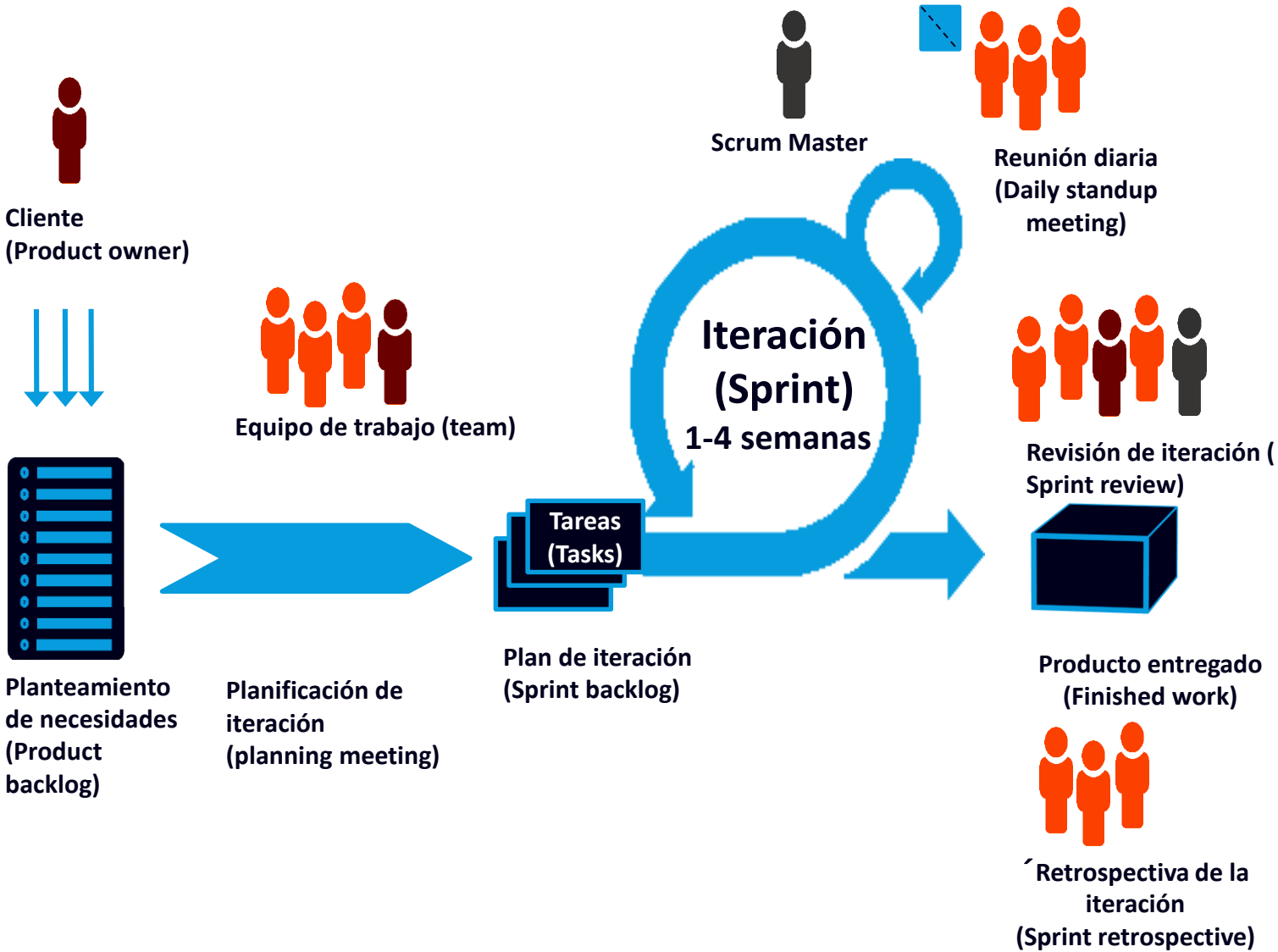
Implementación de Software (2)

- A partir del diseño se **construyen** (programan) los componentes de software y se **prueban** de manera individual.
- La siguiente actividad es la **integración** de los componentes y **probar el sistema integrado**, además incluye 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.

SCRUM prácticas ágiles

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

Conceptos más importantes de SCRUM



KUALI-BEH (1)

- 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 algunas **herramientas** pueden facilitar su realización*" (OMG, 2013).

KUALI-BEH (2)

- 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).

UML (1)

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

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 (Booch G., 1998) (Ambler S.W., 2005).
- 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

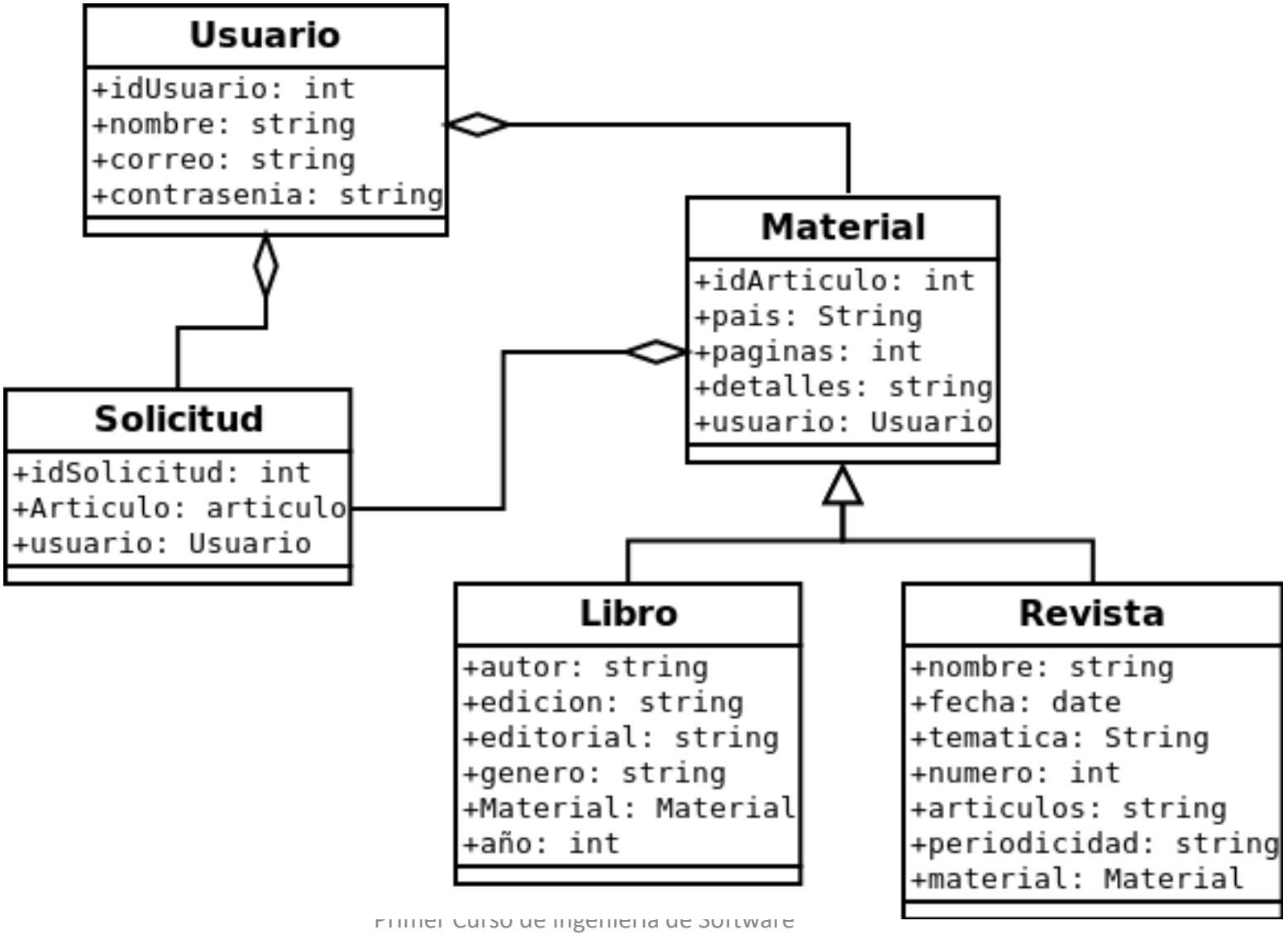


Diagrama de paquetes

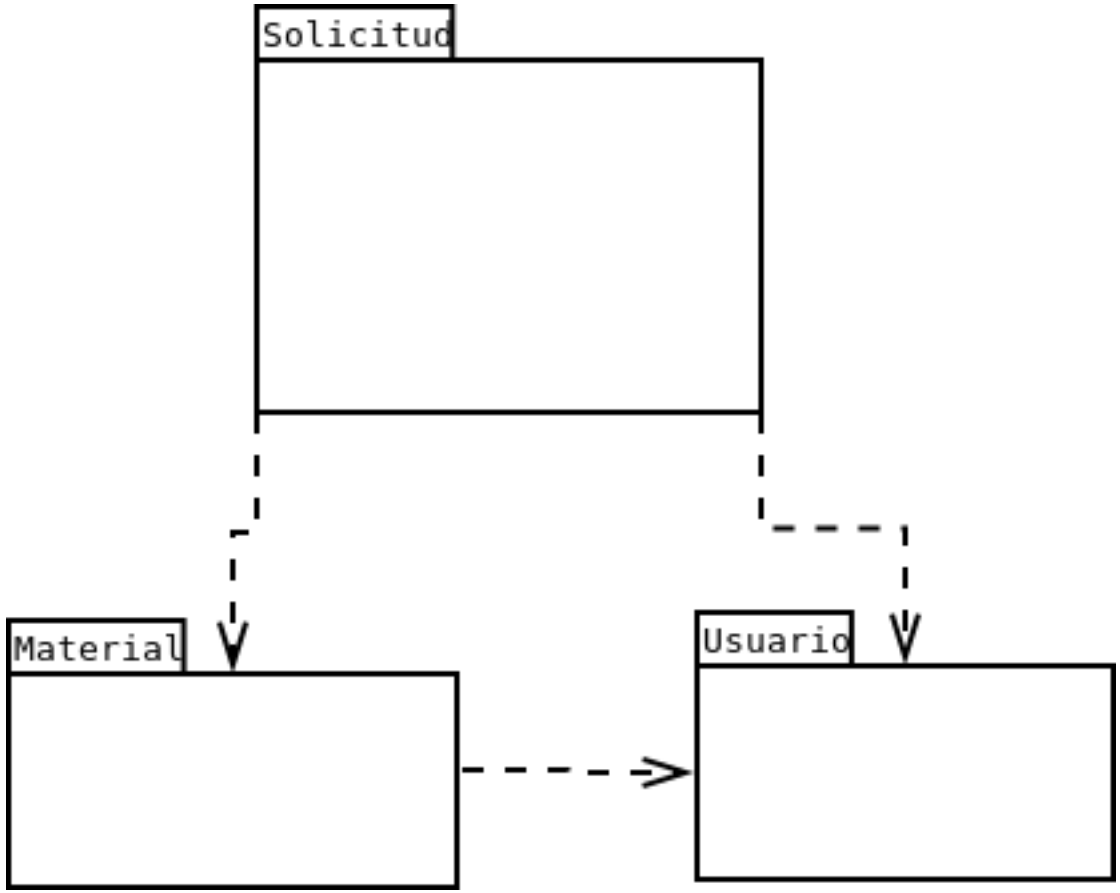
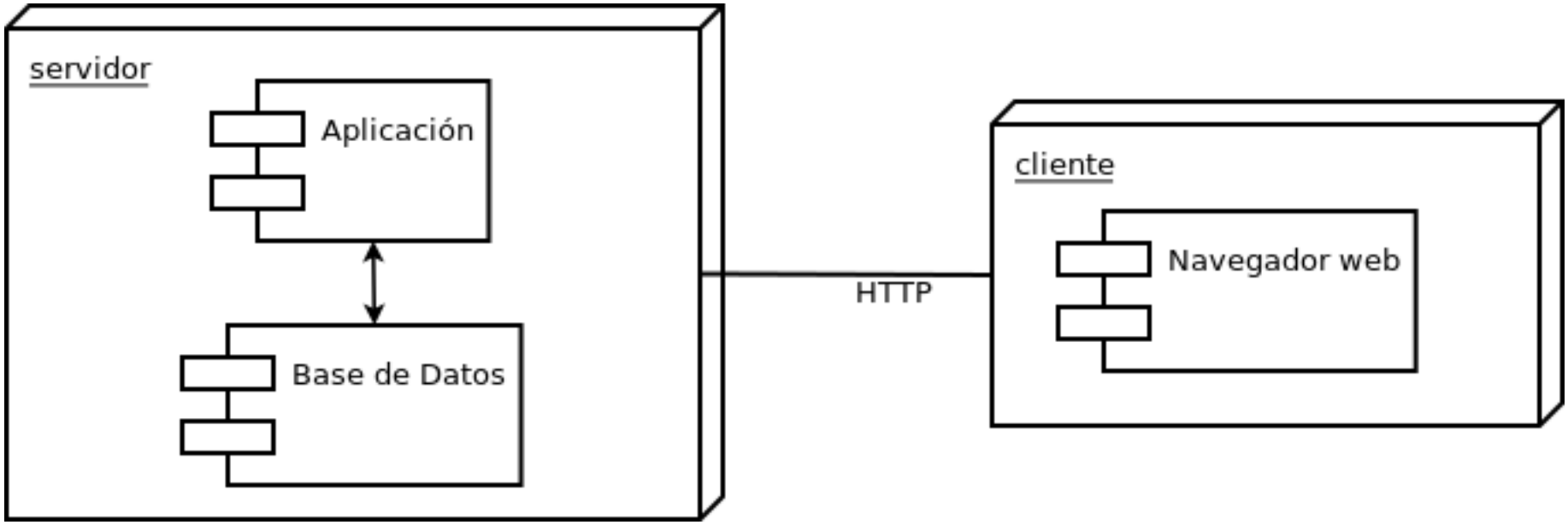


Diagrama de distribución



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 navegación)

Diagrama general de casos de uso

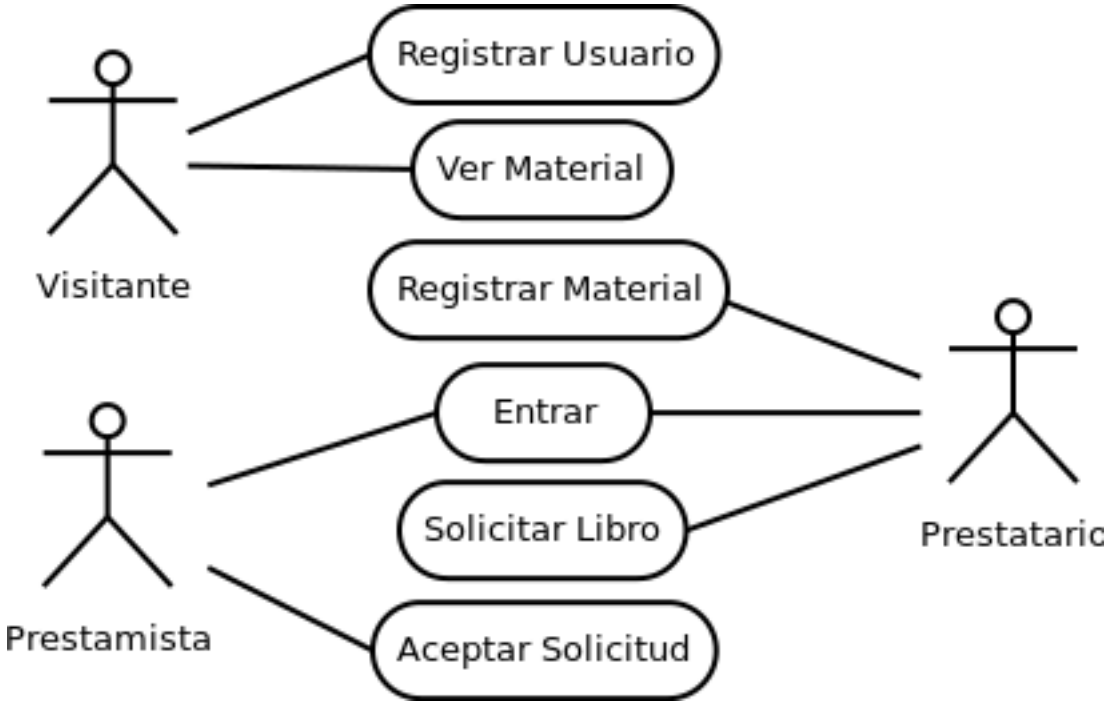


Diagrama de secuencia

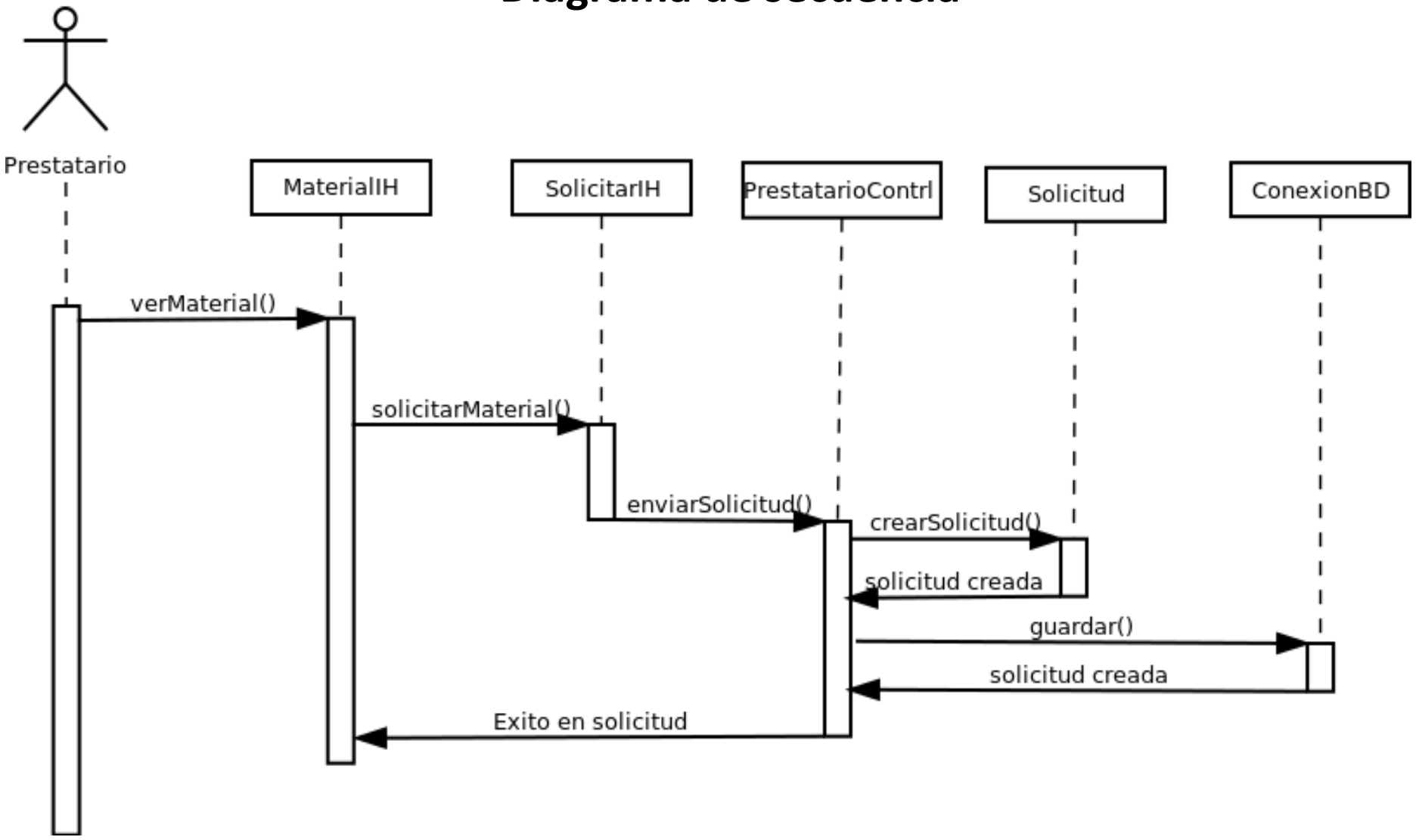
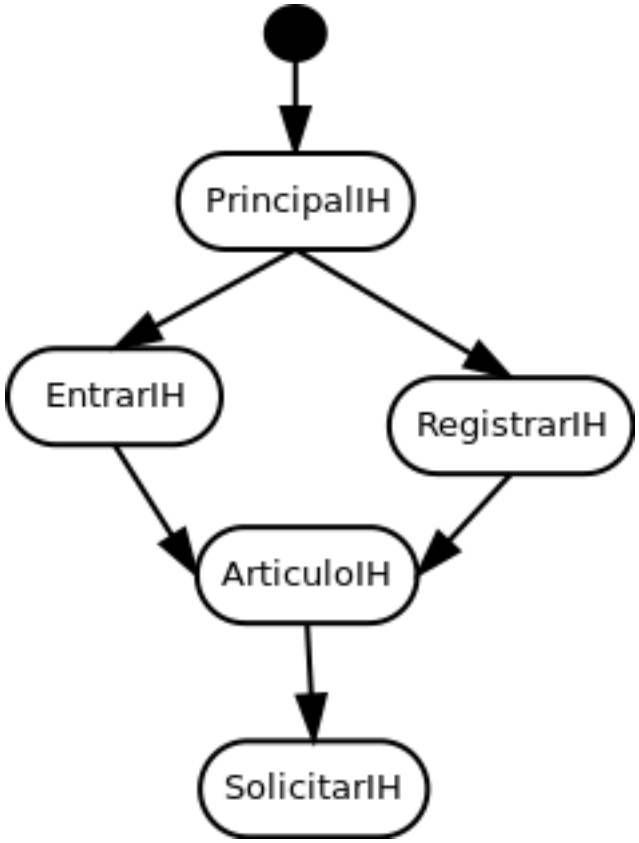


Diagrama de estados (navegación)



Método y prácticas a utilizar

Método Inicial de Desarrollo de Software

- Para este curso, hemos definido el método que se llama ***Método Inicial de Desarrollo de Software*** *(MIDS)*. Este método está basado en el estándar **ISO/IEC 29110 Perfil Básico** por lo que sigue los dos procesos de desarrollo que se definen para este perfil.
- Los nombres de las prácticas que conforman el método corresponden a las actividades de estos dos procesos y se agregaron unas prácticas sociales para facilitar el trabajo en equipo.

Método	
MIDS	**Método Inicial de Desarrollo de Software**

Propósito

Los alumnos conocerán y aplicarán las prácticas básicas de Ingeniería de Software mediante desarrollo de un proyecto de software por equipos.

Entrada	Resultado
Condiciones •Los alumnos tienen los conocimientos de programación suficientes para desarrollar un producto de software **Productos de trabajo** •*Planteamiento de Necesidades* •*Guión del Curso* •*Plantillas*	**Condiciones** •Los alumnos han practicado conceptos básicos de Ingeniería de Software desarrollando un proyecto de software **Productos de trabajo** •Documentación y Producto de Software del proyecto

Prácticas

Prácticas Sociales
Prácticas Administrativas
Prácticas para el Desarrollo

Prácticas del MIDS (1)

- **Prácticas Sociales**
  - **PS1. Conformar el equipo**
  - **PS2. Definir la comunicación en el equipo**
  - **PS3. Crear el repositorio común de documentos**
  - **PS4. Retrospectiva de la iteración**

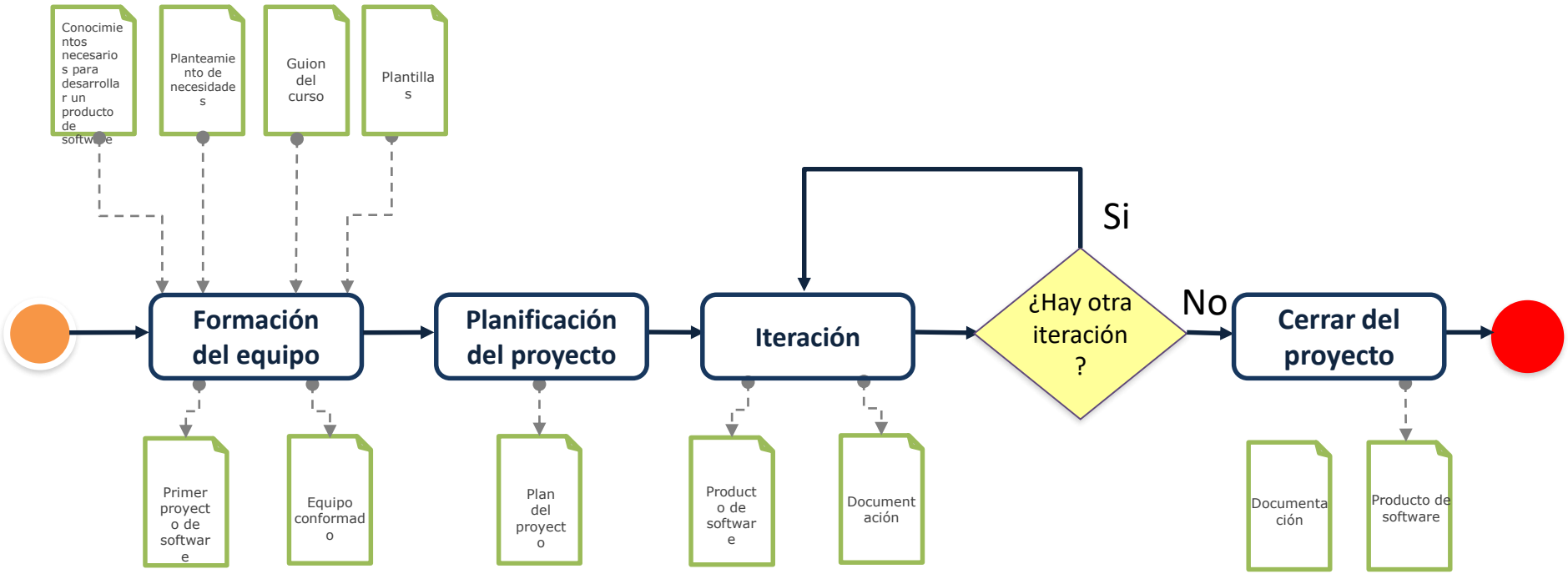
Prácticas del MIDS (2)

- **Prácticas Administrativas:** Estas prácticas están basadas en el proceso de *Administración de Proyecto* de la ISO/IEC 29110 y se aplican algunos de los conceptos ágiles de 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**

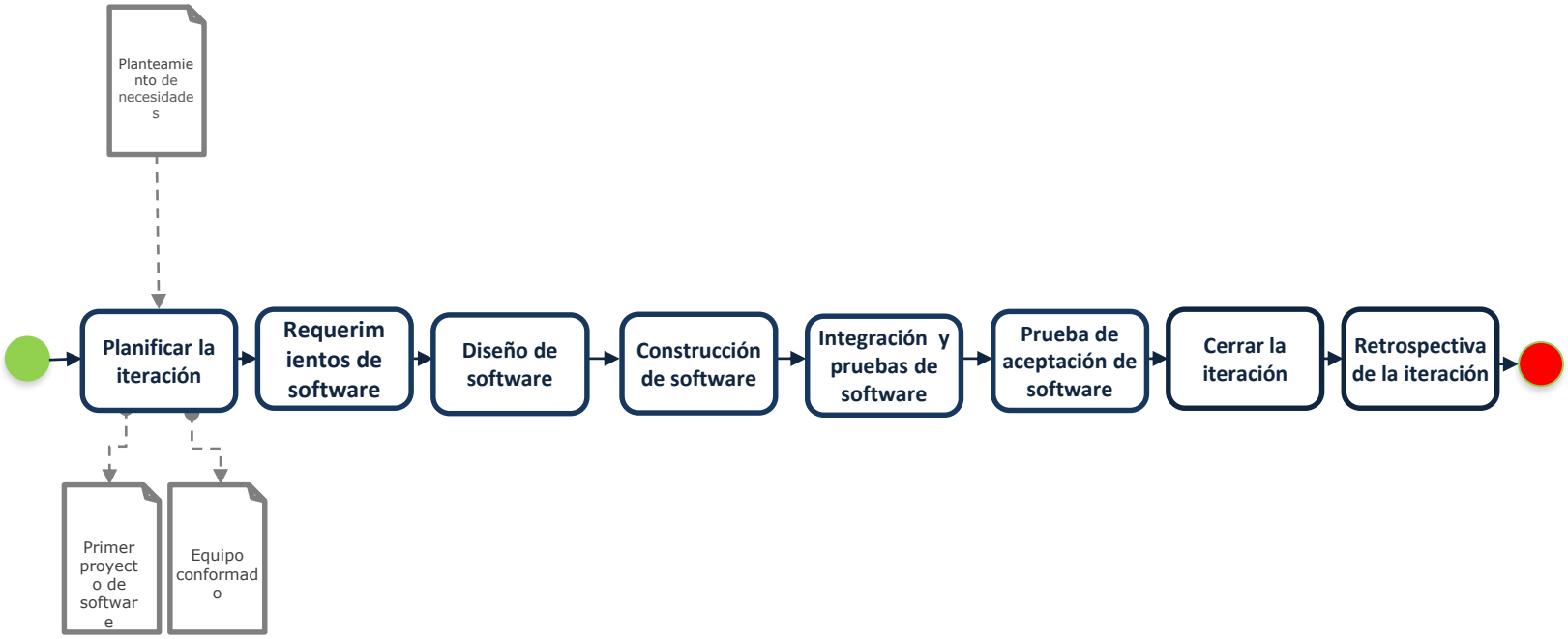
Prácticas del MIDS (3)

- **Prácticas para el Desarrollo** basadas en el proceso de *Implementación de Software* de la ISO/IEC 29110.
  - **PD1. Requerimientos de software**
  - **PD2. Diseño de software**
  - **PD3. Construcción de software**
  - **PD4. Integración y pruebas de software**
  - **PD5. Prueba de Aceptación del Software**

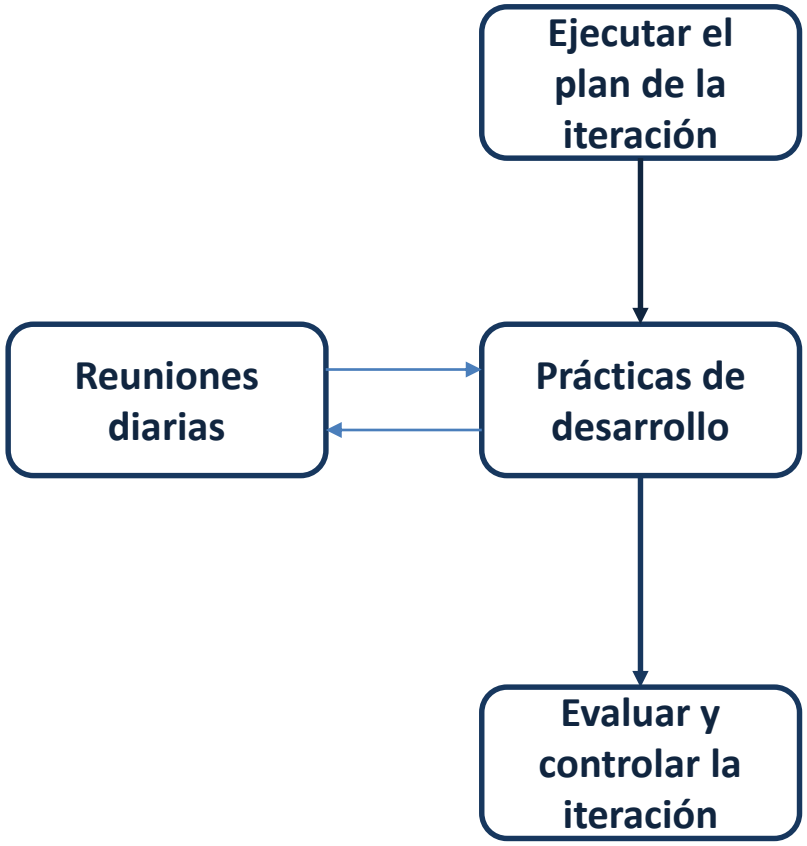
Aplicación del método inicial de desarrollo de software (MIDS)



Prácticas de una iteración



Ejecución, evaluación y control del desarrollo



Apoyos al MIDS

- *Planteamiento de Necesidades*
- *Plantillas*

- *Planteamiento de Necesidades ejemplo*
- *Plantillas ejemplo*

Roles para miembros del equipo

- Las **prácticas** tienen **actividades** que los miembros del equipo tendrán que **ejecutar,** 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 y pruebas del software.

•Participar en las reuniones de trabajo.

•Revisar y corregir los productos de los que sea responsable.

•Aplicar los estándares solicitados.

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 al docente de manera oportuna para que ayude a resolverlos.

•Convoca y dirige las reuniones del equipo.

•Coordina la retrospectiva de la iteración.

Responsable técnico

Objetivo

•Lograr que los resultados del equipo sean 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.

•Seleccionar las herramientas necesarias para el trabajo.

•Entrenar a los miembros del equipo en el uso de las herramientas que trabajarán.

•Coordinar la integración del código de los desarrolladores.

Responsable de la calidad

Objetivos

•Asegurar que se sigan los estándares establecidos para cada producto.

Habilidades

•Persona ordenada e interesada en la calidad del software.

•Saber hacer buenas revisiones y pruebas a los productos.

Responsabilidades

•Coordinar la integración de los documentos de todas las actividades del desarrollo apoyado por los demás roles.

•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 documentos y del código.

Habilidades

•Conocedor de redes sociales y repositorios compartidos

•Entusiasmado en aprender el uso de repositorios y explicar al equipo su uso.

•Disciplinado en el manejo del repositorios comunes de documentos y del código para facilitar la comunicación asíncrona del equipo.

Responsabilidades

•Seleccionar las herramientas necesarias para apoyar al equipo en la comunicación y coordinación.

•Crear 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.

Bibliografía

- Ambler S.W. (2005). *The Elements of UML 2.0 Style.* Cambridge University Press.
- Ambler S.W. (2005). *The Elements of UML 2.0 Style.* Cambridge University Press.
- Booch G., R. J. (1998). *The Unified Modeling Languajes. Users Guide.* Addison Wesley.
- Ikujiro Nonaka & HirotakaTakeuchi. (1986). *The New Product Development Game.* HArvard Business Review.
- ISO/IEC29110. (2011). *29110-5-1-2Software Engineering-lifecycle Profiles for Very Small Entities Management and Engineering Guidde. s.1.* Software Engineering.
- MoProSoft. (2005). *Modelo de Procesos para la Industris de Software.* Estándar Mexicano Nacional MNX-1-059-NYCE-2005 .
- OMG. (06 de 2013). *Essence – Kernel and Language for Software Engineering*. Obtenido de http://www.omg.org/spec/Essence/1.0/Beta1/PDF/
- Rumbaugh J., I. J. (1998). *The Unified Modeling Language. Reference Manual.* Addison Wesley.
- Schwaber, K. S. (2011). *The Scrum Guide – The Definitive Guide to Scrum: The Rules of the Game.* Obtenido de http://www.scrum.org/storage/scrumguides/Scrum%20Guide%20-%202011.pdf
- SEMAT. (27 de 09 de 2012). *http://semat.org/?page_id=2.*
- SWEBOK. (2014). *Guide to the Software Engineering Body of Knowledge v3.0.* IEEE Computer Society.