**Licenciatura en Sistemas**

**Ingeniería de Software I**

**Herramientas para el proceso de software**

**Ingeniería de software asistida por computadora**

**(CASE)**

**Herramienta de diseño: Visual Paradigm**

**Descripción y propósito**

* **Visual Paradigm** es una plataforma de diseño usada para crear **diagramas UML**, **BPMN**, **ER** e **historias de usuario**.
* Provee ayudas visuales que facilitan la comprensión del diseño y comunican la arquitectura del software y los procesos de negocio de manera efectiva.
* Ofrece soporte a distintas metodologías de desarrollo de software.

**Aplicabilidad en procesos ágiles**

* Incorpora soporte para el modelado ágil a través de herramientas como **mapas de historias de usuario**, **tableros para planificación de sprints**, y gestión del **backlog**.
* Su flexibilidad lo vuelve ideal para el desarrollo iterativo, permitiendo que los diagramas y modelos evolucionen a la par que el proyecto.
* Los equipos ágiles lo pueden usar para visualizar el progreso, ajustar prioridades y refinar los requerimientos continuamente.

**Ejemplo de uso (fitness tracker app)**

* Modelado de **casos de uso** como “Definir metas de ejercicio”, “Registro de nutrición”, etc.
* Creación de **diagramas de clase** (p.ej Usuario, RutinaDeEjercicio, LogDeNutricion).
* Uso de **diagramas de secuencia** para mapear interacciones de usuario como la sincronización de datos de ejercicio con dispositivos (p.ej. smartwatch).

**Herramienta de gestión de proyectos: Jira**

**Descripción y propósito**

* **Jira** es una herramienta desarrollada por **Atlassian**, ampliamente utilizada para el seguimiento de proyectos e incidencias.
* Originalmente diseñada para rastrear bugs, hoy en día es una plataforma poderosa para la gestión de proyectos ágiles, con características como **tableros de Scrum y Kanban**, **planificación de sprints** y **paneles de informes**.

**Aplicabilidad en procesos ágiles**

* Provee plantillas **Scrum** y **Kanban**.
* Permite planificación y seguimiento flexibles de sprints con **gráficos de trabajo pendiente**, **informes de velocidad** y **paneles en tiempo real**.
* Los **backlogs** pueden ser priorizados y refinados por product owners y stakeholders.
* Los equipos pueden fácilmente adaptar **flujos de trabajo**, definir **historias de usuario**, seguir **épicas** y mapear **dependencias**.
* Se integra con **Confluence**, **Bitbucket**, **GitHub**, y herramientas **CI/CD**, creando un ambiente ágil interconectado.

**Ejemplo de uso**

* Permitiría al equipo mantener un **backlog** de características (como contador de pasos, definición de metas, etc).
* Permite seguir el progreso a medida que nuevas características (p.ej. monitor de frecuencia cardíaca) van siendo implementadas.
* Ayuda a coordinar el trabajo a lo largo del desarrollo, diseño y QA, usando tableros y **links entre incidencias.**

**Herramienta de análisis y modelado: Lucidchart**

**Descripción y propósito**

* **Lucidchart** es una herramienta de colaboración visual basada en la nube.
* Se usa para crear **diagramas UML**, **ER**, **de flujo**, **arquitecturas de sistemas**.
* Usada ampliamente en el desarrollo de software para planear y visualizar procesos, sistemas y estructuras organizacionales.

**Aplicabilidad en procesos ágiles**

* Permite la comunicación en tiempo real, lo cual está en línea con el énfasis en la comunicación y el trabajo en equipo.
* Permite crear e iterar rápidamente sobre diagramas como **resúmenes de arquitectura**, **flujos de proceso** o **mapas de experiencia de usuario**, actualizandolos a medida que los requerimientos evolucionan.
* Se integra con **Confluence**, **Jira** y **Slack** permitiendo que los diagramas estén embebidos directamente en la documentación o los tableros de tareas.
* Útil para planear **ceremonias** como **sprint planning/retrospectivas**.

**Ejemplo de uso**

* Mapeo de la arquitectura del sistema (**UI**, **flujos de datos**, integración con **API**s).
* Crear **mapas de recorrido de usuario** (p.ej. registro, loguear una rutina de ejercicio, visualizar progreso).

**Herramienta de generación de código: Mendix**

**Descripción y propósito**

* **Mendix** permite a desarrolladores y usuarios de negocio construir aplicaciones web y móviles rápidamente y con mínima codificación manual.
* Ofrece componentes **drag-and-drop**, **flujos** **lógicos**, **módulos reutilizables** y **modelado visual de datos**, lo que lo hace útil para el prototipado rápido y el desarrollo iterativo.
* Soporta el despliegue, versionado y la colaboración.

**Aplicabilidad en procesos ágiles**

* Habilita el **prototipado rápido** y los ajustes basados en **feedback**.
* **Usuarios de negocio**, **desarrolladores** y **stakeholders** pueden trabajar juntos en un ambiente compartido.
* Gracias al desarrollo visual y el testeo automático los cambios se pueden entregar en ciclos cortos (ideal para **sprints**).
* Incluye **product backlogs**, **historias de usuario** y **tableros de tareas**.
* El **seguimiento de versiones** y el **rollback** permite desplegar apps frecuente e incrementalmente.

**Ejemplo de uso**

* Construir la **UI** rápidamente con elementos **drag-and-drop**.
* Conectarse con **API**s de dispositivos o plataformas web de fitness.
* Implementar mecanismos de feedback de usuario.
* Lanzar versiones tempranas de la app para testeo en pocas semanas.

**Herramienta de prueba: Selenium**

**Descripción y propósito**

* **Selenium** es un paquete de herramientas **open source** diseñado específicamente para automatizar interacciones en navegadores web.
* Permite a desarrolladores e ingenieros de QA crear scripts que simulan el comportamiento del usuario. Especialmente popular para **pruebas funcionales** y **de regresión** en interfaces web.
* Soporta múltiples lenguages de programación como **Java**, **Python** y **C#**.

**Aplicabilidad en procesos ágiles**

* Las pruebas automatizadas son ideales para ciclos de iteración rápidos.
* Se integra fluidamente con **herramientas CI/CD** (**Jenkins**, **Git Actions**).
* Permite **pruebas de regresión automatizadas** lo cual es crucial en entornos ágiles donde se agregan nuevas características frecuentemente.
* Combinado con frameworks como **Cucumber** o **TestNG**, permite **BDD** (Behavior Driven Development) o **TDD** (Test Driven Development).

**Ejemplo de uso**

* Probar automáticamente el login al portal web de la app.
* Verificar que el sistema de definición de metas funciona correctamente.
* Simular varios flujos de usuario (p.ej. actualización de perfil, sincronizacion de datos).

**Herramienta de control de versiones: Git**

**Descripción y propósito**

* **Git** es un sistema de **control de versiones** distribuido que permite a equipos gestionar cambios en el código fuente desarrollar colaborativamente.
* Cada desarrollador tiene una copia completa del **repositorio** y los cambios son rastreados por **commits**, lo que permite colaborar sin sobreescribir el trabajo de los demás.

**Aplicabilidad en procesos ágiles**

* Se integra con **herramientas CI** (integración contínua) que testean y lanzan el código automáticamente.
* La capacidad de hacer **pulls** y **peer reviews** se alinean con los principios ágiles de transparencia y trabajo en equipo.
* Cada **commit** documenta el progreso y facilita revertir las modificaciones lo cual respeta la filosofía ágil de capacidad de adaptación al cambio.

**Ejemplo de uso**

* Organizar características (rastreo por GPS, contador de pasos) en diferentes ramas.
* Permitir que múltiples desarrolladores trabajen en paralelo en módulos de **UI**, **backend** y **almacenamiento de datos**.
* Usar pull requests para revision y mergeo de de los entregables del sprint.
* Volver a versiones previas del código si una característica nueva causa bugs.

**Herramienta de documentación: Doxygen**

**Descripción y características**

* **Doxygen** es un generador de documentación usado principalemnte para **C**, **C++**, **Java**, **Python** y otros lenguajes.
* Analiza el código fuente y produce documentación estructurada y legible en varios formatos (**HTML**, **PDF**, **LaTeX**, etc).
* Al insertar comentarios formateados especialmente directamente el los archivos fuente, **Doxygen** automáticamente genera documentación detallada (p.ej. jerarquías de clases, definiciones de funciones y relaciones entre componentes).

**Aplicabilidad en procesos ágiles**

* Soporta documentación incremental y continua que además se genera automáticamente lo cual se adapta a las iteraciones y actualizaciones frecuentes.
* Evita a los desarrolladores hacer documentación manual lo cual les permite enfocarse en el software.

**Ejemplo de uso**

* Doxygen podría documentar por ejemplo, las clases Usuario, Actividad, RastreadorDeMetas, o **API**s públicas y **endpoints** si la app se conecta con servicios externos (p.ej. **Google Fit** o **Apple Health**).