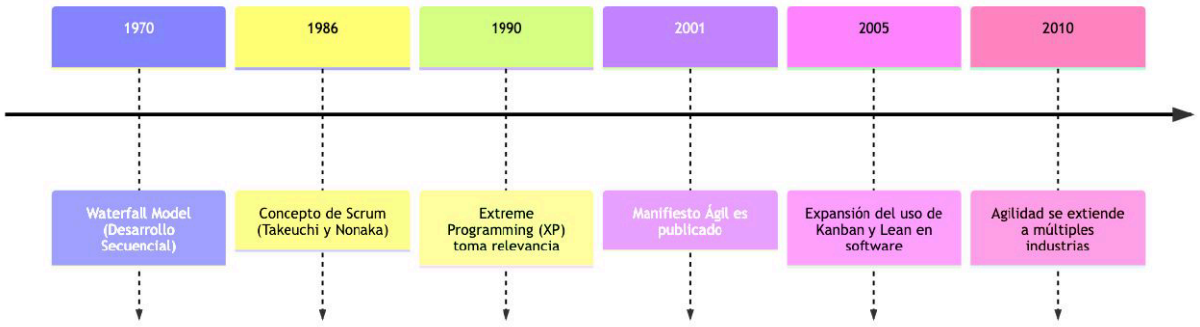
Entrega: Nadja Villarroel N.

1. Descripción de los Fundamentos de las Metodologías Ágiles:

- Explicar los orígenes de las metodologías ágiles y las diferencias clave con los modelos tradicionales de desarrollo como Waterfall y RUP.

Las metodologías ágiles aparecieron hace más de 30 años, a partir de la necesidad de detectar y corregir errores, o realizar cambios de la manera más rápida posible. Para abordar situaciones imprevistas o cambios de contexto, los enfoques tradicionales no lograban dar señas temprana de éxito o fracaso, ya que se debía esperar al final del desarrollo para hacer el balance: tomaban demasiado tiempo en entregar valor, no era adaptable a cambios, se centraba en hacer mucha documentación antes del desarrollo y los equipos tenían poca interacción con el cliente lo que llevaba a muchos errores, estimaciones de tiempo incorrectas o requerimientos incorrectos.



Luego de algunas iniciativas por cambiar estas metodologías rígidas (como el Modelo Waterfall), en 2001 se creó el Manifiesto Ágil, que oficializó esta metodología de desarrollo. El manifiesto Ágil fue creado por un grupo de desarrolladores en Utah, con el objetivo de mejorar el proceso de desarrollo de software, enfocándose en la colaboración, **flexibilidad** y entrega continua de valor.

Las metodologías ágiles se basan en este Manifiesto Ágil y difieren de los enfoques tradicionales en lo siguiente:

- Waterfall: Modelo secuencial donde cada fase debe completarse antes de avanzar. Los cambios son difíciles de manejar una vez que se pasa de fase. Agile propone separar y subdividir tareas que son revisadas periódicamente con el objetivo de reaccionar con tiempo y readaptar el trabajo en caso de existir imprevistos.

- RUP (Rational Unified Process): Aunque es iterativo, RUP sigue un enfoque más estructurado, puede ser costoso en proyectos pequeños, requiere un análisis y documentación extensos al inicio, y los riesgos se gestionan desde el inicio mediante identificación y mitigación. Mientras que Agile es más flexible y

adaptativo a los cambios constantes con planificación progresiva y los riesgos se manejan con entregas frecuentes y retroalimentación.

- PMBOK: Un enfoque que prioriza la planificación detallada y el control estricto del proyecto mediante un conjunto de buenas prácticas, procesos y estándares. Pese a que es útil en proyectos grandes que requieren una planificación y documentación detallada, se presenta bastante rígida y protocolar para proyectos que tienen requisitos o situaciones en constante cambio. Agile, en cambio, promueve una mayor flexibilidad y adaptación continua a los requisitos.

Aspecto	Waterfall	RUP	PMBOK	Agile
Enfoque	Secuencial y lineal. El proyecto sigue fases rígidas en orden.	Basado en procesos y planificación. Se centra en definir y seguir un marco estructurado para el desarrollo.	Basado en procesos y planificación detallada. Define estándares y mejores prácticas para la gestión de proyectos en cualquier industria.	Iterativo y adaptable. Prioriza la entrega rápida de valor al usuario y la flexibilidad ante cambios.
Planificación	Se realiza al inicio, con un plan detallado de todo el proyecto en fases (1.Requerimientos, 2.Diseño, 3.Implementación, 4.Pruebas, 5.Despliegue, 6.Mantenimiento)	Requiere una planificación detallada al inicio, con fases bien definidas.	Se elabora un plan maestro con objetivos, cronograma, costos y riesgos desde el inicio.	Se planifica progresivamente, adaptándose a nuevas necesidades en cada iteración.
Flexibilidad	Poco flexible, los cambios requieren volver a fases anteriores.	Más rígido ante cambios (necesita aprobación formal).	Poco flexible, los cambios deben pasar por procesos formales de gestión del cambio.	Altamente adaptable a cambios en requisitos.
Entrega de Valor	Se entrega el producto solo al final del proyecto.	Se entregan versiones funcionales en cada iteración, pero el producto completo se lanza en la fase final.	Se entrega el producto al final del proyecto o en hitos predefinidos.	Se entregan versiones funcionales en cada iteración.

Documentación	Extensa y detallada, con documentos para cada fase.	Extensa, detallada y obligatoria en cada fase.	Extensa y detallada, con informes formales en cada fase.	Mínima, solo lo esencial para el desarrollo.
Gestión del riesgo	Los riesgos se analizan al inicio, pero pueden ser difíciles de manejar si surgen tarde.	Se identifican y gestionan los riesgos desde el inicio del proyecto.	Se identifican y mitigan riesgos desde el inicio del proyecto.	Los riesgos se abordan de manera iterativa con entregas continuas y retroalimentación.
Tamaño del equipo	Equipos grandes con roles especializados.	Equipos grandes y estructurados con roles específicos.	Equipos grandes, con roles bien definidos.	Equipos pequeños y multidisciplinarios, autogestionados.
Jerarquía	Estructura jerárquica, con decisiones centralizadas.	Jerárquico, con roles bien definidos (analistas, arquitectos, desarrolladores, testers, etc.).	Estructura jerárquica, con niveles de liderazgo y supervisión.	Horizontal, con comunicación constante entre los miembros.
Roles principales	- **Gerente de Proyecto** (dirige el proceso). - **Analistas** (definen requisitos). - **Arquitectos** (diseñan la solución). - **Desarrolladores** (implementan el código). - **Testers** (prueban el software).	- **Gerente de Proyecto** (liderazgo y gestión). - **Analista de Negocios** (requisitos y especificaciones). - **Arquitecto de Software** (diseño de la arquitectura). - **Desarrolladores** (implementación). - **Testers** (pruebas y control de calidad).	- **Gerente de Proyecto (Project Manager)** (supervisa todo el proyecto). - **Patrocinador (Sponsor)** (apoya financieramente el proyecto). - **Equipo del Proyecto** (ejecuta tareas específicas).	- **Scrum Master** (facilitador del equipo). - **Product Owner** (gestiona el backlog y prioridades). - **Equipo de Desarrollo** (multidisciplinario, sin jerarquías fijas).

Interacción con el cliente	Se define al inicio y al final. Poco contacto durante el desarrollo.	Se define al inicio y se revisa en ciertas fases clave.	Se define al inicio y se mantiene a través de informes y reuniones planificadas.	Continua y frecuente, con iteraciones basadas en retroalimentación.
Comunicación	Formal, con reuniones planificadas y documentación detallada.	Formal, con reuniones planificadas y documentación extensa.	Formal, con informes y reuniones de seguimiento.	Informal y constante, con reuniones diarias (Daily Scrum en Scrum).

Según lo descrito en esta tabla, podemos destacar

- Identificar las diferencias entre Agile, Agilidad y Agilismo, describiendo cómo se aplican en el contexto actual del desarrollo de software.

- **Agile**: Es el conjunto de metodologías que implementan los principios del Manifiesto Ágil (ejemplos de metodologías: *Scrum* - entregas rápidas con ciclos cortos, *XP* - donde la calidad del código es clave, *Kanban* - flujo de trabajo continuo, *Lean Software Development* - para equipos grandes, maximizar la eficiencia y reducir desperdicios en el desarrollo, *Crystal* - para equipos pequeños o medianos sin reglas estrictas, *Feature-Driven Development (FDD)* - proyectos grandes con estructura, *Dynamic Systems Development Method (DSDM)* - proyectos grandes con estructura, etc.)

- **Agilidad**: Se refiere a la capacidad de una organización o equipo para adaptarse rápidamente a los cambios, ajustarse a las nuevas demandas y responder de manera efectiva. Aplicar esto, por ejemplo, permitiría a un equipo cambiar su estrategia ante unj imprevisto.

- **Agilismo:** Describe la cultura o mentalidad de una organización que sigue los principios de Agile en todos sus aspectos, no solo en el desarrollo de software (negocios, educación, marketing, etc). Adoptar esta filosofía a cualquier aspecto implica desarrollar planificación, procesos y testear resultados basados en la flexibilidad y pronta respuesta a cambios o imprevistos; aunque muchas veces las empresas confunden el concepto con “entregar/desarrollar rápido”

2. Valores y Principios del Manifiesto Ágil:

- Identificar los 4 valores y los 12 principios del Manifiesto Ágil.

El Manifiesto Ágil fue creado por un grupo de 17 desarrolladores de software en 2001, con el objetivo de mejorar el proceso de desarrollo de software, enfocándose en la colaboración, **flexibilidad** y entrega continua de valor.

Para garantizar su aplicación, el Manifiesto Ágil define 4 valores y 12 principios:

Valores:

1. Individuos e interacciones sobre procesos y herramientas
2. Software funcionando sobre documentación extensiva.
3. Colaboración con el cliente sobre negociación de contratos.
4. Respuesta ante el cambio sobre seguir un plan rígido.

Principios:

1. La prioridad es satisfacer al cliente con entregas rápidas y continuas.
2. Bienvenidos los cambios de requisitos en cualquier etapa del proyecto.
3. Entregas de software frecuentes y funcionales.
4. Colaboración entre equipos de negocio y desarrollo.
5. Confianza y apoyo a los equipos motivados.
6. Comunicación cara a cara como la mejor forma de transmitir información.
7. Software funcionando como medida principal de progreso.
8. Desarrollo sostenible sin sobrecargar al equipo.
9. Excelencia técnica y buen diseño mejoran la agilidad.
10. Simplicidad y eficiencia en el trabajo.
11. Equipos autoorganizados generan mejores resultados.
12. Reflexión y mejora continua en cada iteración.

Al aplicar estos principios para garantizar el cumplimiento de esos valores, los equipos tendrán facultad de evaluar caso a caso qué aspectos del proyecto pueden reacomodar o postergar en función del cumplimiento de otros aspectos prioritarios para el momento. Por ejemplo, responder al principio “Software funcionando como medida principal de progreso” en un proyecto donde han cambiado algunos requisitos en el camino y el cliente necesita cumplir con una versión en una fecha estipulada; se podría gestionar conversar con el cliente, proponer postergar la documentación técnica para el final del proyecto; sólo elaborar documentación de uso, bloquear algunas funcionalidades del proyecto para su posterior desarrollo, asegurar la navegación principal y animaciones de elementos, integrar el diseño paulatinamente, etc.

Todas estas decisiones guiadas por estos principios, permiten el desarrollo sostenible del proyecto, asegurando una mejora continua, adaptabilidad a los cambios y flexibilidad en el desarrollo.

3. Explicación del Marco de Trabajo Scrum y Roles:

- Explicar los principios y valores de Scrum y cómo este marco de trabajo ágil ayuda a organizar el desarrollo de proyectos.

Scrum es un marco de trabajo ágil para gestionar desarrollo de software y productos digitales con requisitos cambiantes. Se basa en iteraciones cortas y entregas incrementales, llamadas *sprints*: periodo de tiempo determinado por el equipo, generalmente de 1 a 4 semanas máximo.

Los valores y principios de Scrum se alinean con los fundamentos del Manifiesto Ágil y se centran en la colaboración, transparencia y mejora continua.

Principios Clave:

1. **Transparencia**: Todos pueden ver el progreso y los problemas.
2. **Inspección:** Se revisa constantemente el avance y la calidad.
3. **Adaptación**: Se ajustan estrategias según sea necesario.

Los cinco valores clave de Scrum son:

1. **Compromiso:** El equipo se compromete a alcanzar los objetivos del sprint.
2. **Coraje:** Se fomenta la toma de decisiones y enfrentar desafíos.
3. **Enfoque:** Se prioriza lo más importante en cada Sprint.
4. **Respeto:** Los miembros del equipo se respetan mutuamente como profesionales.
5. **Apertura:** Se aceptan opiniones, retroalimentación y mejoras.

Este marco de trabajo ágil permite la horizontalidad de un equipo, que pueden proponer cambios y mejoras siempre alineados con lograr la mejor calidad del producto posible. Seguir estos valores ayuda a mantener el enfoque del equipo, a ser críticos y creativos con su trabajo, a enfrentar desafíos y apoyarse en sus compañeros para encontrar soluciones. Al estar constantemente revisando el trabajo hecho y no sólo dejándolo en el olvido como una tarea completada más, se permite lograr ese compromiso para alcanzar los objetivos del sprint.

- Describir los roles clave en Scrum (ScrumMaster, ScrumDeveloper, ProductOwner) y sus responsabilidades dentro de un equipo ágil.

Scrum Master: facilitador del equipo

El ScrumMaster es responsable de facilitar el proceso Scrum y asegurarse de que el equipo siga los principios y prácticas ágiles. Actúa como un coach, ayudando al equipo a eliminar impedimentos, promoviendo la mejora continua y asegurando que los principios de Scrum se mantengan.

Scrum Developer (o Team Members): desarrollan el producto

Los Scrum Developers son los miembros del equipo responsables de realizar el trabajo técnico. Ellos se encargan de hacer estimaciones y desarrollar, probar y entregar el incremento funcional del producto al final de cada sprint. Los developers trabajan de manera colaborativa, autoorganizada y multidisciplinar.

Product Owner: gestiona el backlog y prioridades

El Product Owner es el responsable de maximizar el valor del producto que el equipo está desarrollando. Administra el Product Backlog y se asegura de que las tareas más importantes y que aportan mayor valor al producto se prioricen. También es el punto de contacto principal con los stakeholders.

La interacción entre estos roles se puede entender, por ejemplo, en un equipo que define que en un sprint trabajará en crear cuentas de usuario y gestionar tareas de una aplicación: el Product Owner define las funcionalidades en el product backlog (por ejemplo: crear cuentas de usuario, permitir creación y asignación de tareas, agregar notificaciones “pendiente”, “hecho”, “archivado”); el Scrum Master facilita la reunión de planificación y los Scrum Developers eligen las historias que trabajarán en el sprint y crean el sprint backlog. Durante las dos semanas que dura el sprint, el equipo trabaja colaborativamente: El Product Owner está disponible para responder dudas, el Scrum Master asegura que no haya bloqueos y que se cumplan las prácticas ágiles, y el Development Team diseña la interfaz, implementa el backend y realiza pruebas. Al final del sprint, el Product Owner valida que los requisitos del sprint se cumplieron, el/los clientes prueban la aplicación y se realiza una retrospectiva para recoger ideas de mejoras.

4. Identificación de las Prácticas y Artefactos en Scrum:

Las prácticas principales de Scrum son las siguientes ceremonias:

1. **Sprint Planning:** se define qué trabajo se realizará en el Sprint. El Scrum Master facilita la reunión, el Product Owner presenta las prioridades del Product Backlog y el Development Team selecciona las tareas a completar en el sprint y planifica cómo trabajarlas.
2. **Daily Scrum:** reunión de equipo diaria, de no más de 15 minutos, para revisar avances y discutir posibles impedimentos. Cada integrante del equipo responde tres preguntas claves: ¿qué hice ayer?, ¿qué haré hoy?, ¿tengo algún bloqueo?
3. **Sprint Review:** Presentación del trabajo terminado al cliente, que tiene por objetivo recibir feedback y evaluar si es necesario hacer ajustes.
4. **Sprint Retrospective:** Reflexión del equipo sobre el resultado y el trabajo, y propone mejoras.

- ***"Sprint Refinamiento":** No es una ceremonia oficial, pero se estila hacer una reunión para tener un sprint planificado por adelantado y se usa esta instancia para aplicar mejoras provenientes del Sprint Retrospective.

  Las Ceremonias Scrum aseguran que el equipo:

  - Tenga una visión clara del trabajo.
  - Se mantenga enfocado y sincronizado.
  - Reciba retroalimentación constante.
  - Mejore continuamente el proceso.

Los artefactos principales de Scrum son:

- **Product Backlog:** es una lista priorizada de todas las características, funciones, mejoras y correcciones que se necesitan en el producto.

  Contiene historias de usuario, requerimientos técnicos y mejoras. Es un documento vivo, evoluciona según las necesidades del negocio y el feedback del usuario. Los elementos más prioritarios están en la parte superior y se refinan constantemente.

  El responsable es el Product Owner quien administra y prioriza el backlog; el Scrum Team refina los requisitos según las necesidades.

- **Sprint Backlog:** Conjunto de tareas seleccionadas (del Producto Backlog) para el sprint actual.

  Se define en el Sprint Planning. Sólo contiene elementos que el equipo puede completar en el sprint. Puede actualizarse según el avance del trabajo.

  El responsable es el Development Team que decide qué tareas incluir y cómo ejecutarlas; mientras que el Scrum Master ayuda a eliminar bloqueos.

- **Incremento:** Resultado del sprint: versión funcional y entregable del producto con las nuevas funcionalidades completadas en el sprint.

  Debe estar listo para usarse y cumplir con los criterios de calidad. Se presenta en el Sprint Review. Se acumula con incrementos anteriores para construir el producto final.

  El responsable es el Development Team que se encarga que el incremento cumpla con la Definition of Done (DoD); mientras que el Product Owner revisa si cumple con los requisitos del negocio y los stakeholders o cliente entregan feedback del nuevo incremento.