



**Agile  
Coach**

## **Copyright and Disclaimer**

Copyright © T-CERT

Miami, Florida

2024

Todos los derechos reservados.

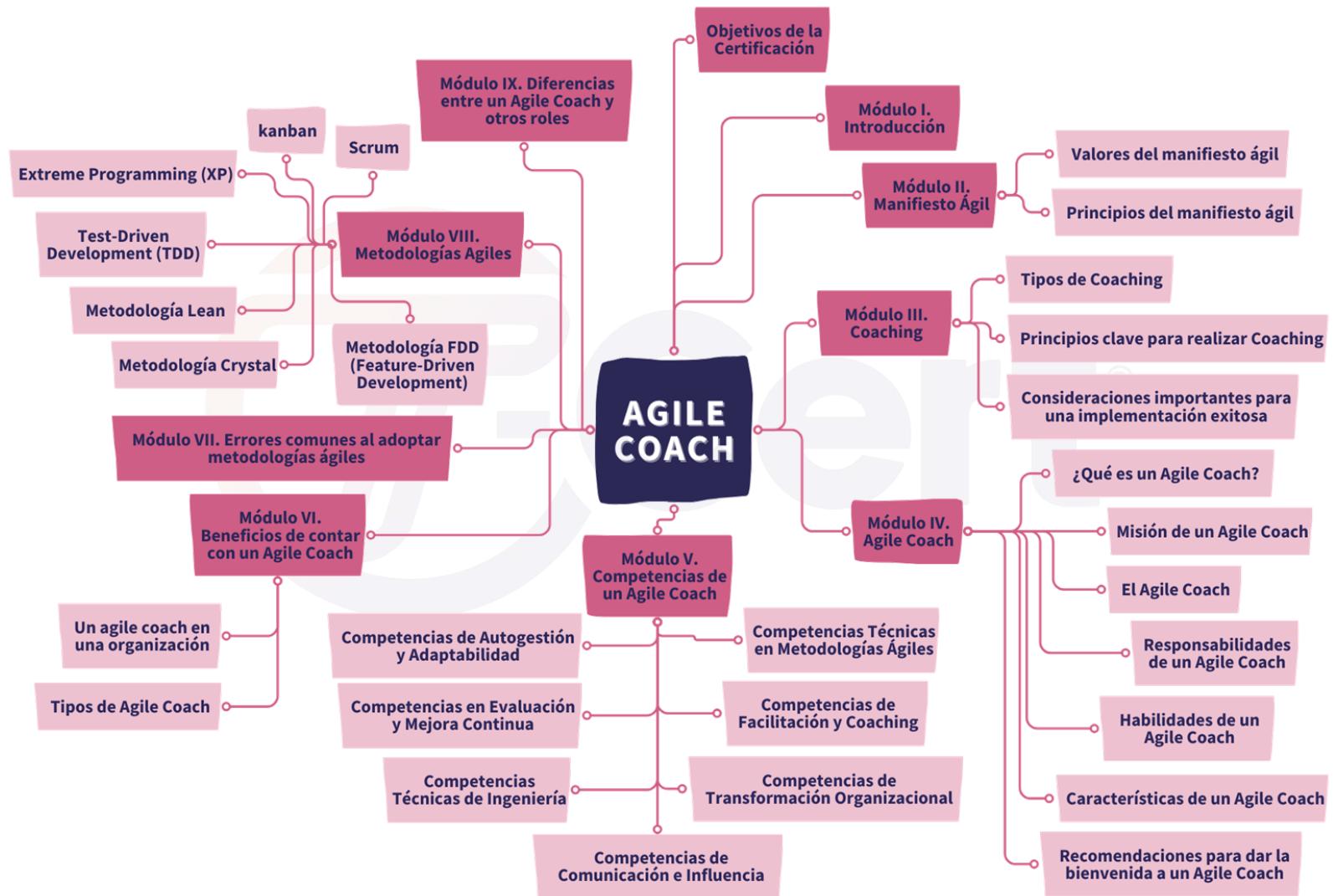
Ninguna parte de esta publicación puede reproducirse, de ninguna forma y por ningún medio, sin el permiso por escrito de T-CERT.

Esta es una publicación comercial confidencial. Todos los derechos reservados. Este documento no puede ser copiado, reproducido en parte, reproducido, traducido, fotocopiado o reducido a cualquier medio sin el consentimiento previo y expreso por escrito del editor. Este curso incluye trabajos sujetos a derechos de autor bajo licencia y está protegido por los derechos de autor

## **Disclaimer**

La información proporcionada sobre el curso, los módulos, los temas y cualquier servicio para los cursos, incluyendo simulaciones o folletos, son sólo una expresión de intenciones y no deben tomarse como una oferta firme o compromiso.

# Agenda



# Contenido

	Pág.
<b>Objetivos de la Certificación .....</b>	<b>8</b>
<b>Módulo I. Introducción.....</b>	<b>10</b>
<b>Módulo II. Manifiesto Ágil.....</b>	<b>13</b>
2.1 Valores del manifiesto ágil .....	13
2.2 Principios del manifiesto ágil .....	18
<b>Módulo III. Coaching.....</b>	<b>24</b>
3.1 Tipos de Coaching.....	24
3.2 Principios clave para realizar Coaching .....	25
3.3 Consideraciones importantes para una implementación exitosa .....	28
<b>Módulo IV. Agile Coach.....</b>	<b>31</b>
4.1 ¿Qué es un Agile Coach?.....	31
4.2 Misión de un Agile Coach .....	32
4.3 El Agile Coach .....	34
4.4 Responsabilidades de un Agile Coach .....	35
4.5 Habilidades de un Agile Coach .....	37
4.6 Características de un Agile Coach.....	38
4.7 Recomendaciones para dar la bienvenida a un Agile Coach .....	40
<b>Módulo V. Competencias de un Agile Coach .....</b>	<b>42</b>
5.1 Competencias Técnicas en Metodologías Ágiles.....	42
5.1.1 Conocimiento profundo de las metodologías ágiles .....	42
5.1.2 Conocimiento de herramientas ágiles .....	43
5.2 Competencias de Facilitación y Coaching .....	44
5.2.1 Habilidades de facilitación .....	44
5.2.2 Habilidad de mentoría .....	44
5.2.3 Coaching de equipos y personas .....	45
5.3 Competencias de Transformación Organizacional.....	45
5.3.1 Habilidad para gestionar el cambio .....	45

5.3.2 Liderazgo y gestión de la cultura organizacional .....	46
5.4 Competencias de Comunicación e Influencia .....	46
5.4.1 Comunicación efectiva .....	46
5.4.2 Influencia y gestión de interesados .....	47
5.5 Competencias Técnicas de Ingeniería.....	47
5.5.1 Conocimiento de prácticas de desarrollo ágil .....	48
5.5.2 Capacidad para facilitar la colaboración entre equipos técnicos y no técnicos.....	48
5.5.3 Conocimiento del negocio .....	48
5.6 Competencias en Evaluación y Mejora Continua.....	48
5.6.1 Medición del rendimiento ágil .....	48
5.6.2 Promoción de la mejora continua .....	49
5.7 Competencias de Autogestión y Adaptabilidad .....	49
<b>Módulo VI. Beneficios de contar con un Agile Coach .....</b>	<b>52</b>
6.1 Un agile coach en una organización .....	52
6.1.1 Mejora en la comunicación y colaboración entre equipos.....	52
6.1.2 Incremento en la eficiencia y productividad .....	53
6.1.3 Fomento de una cultura de mejora continua .....	53
6.2 Tipos de Agile Coach.....	54
6.2.1 Team Coach .....	54
6.2.2 Enterprise Coach .....	55
6.2.3 Technical Coach.....	55
6.2.4 Leadership Coach .....	56
6.2.5 Agile Transformation Coach .....	56
<b>Módulo VII. Errores comunes al adoptar metodologías ágiles .....</b>	<b>58</b>
<b>Módulo VIII. Metodologías Agiles .....</b>	<b>65</b>
8.1 Scrum.....	66
8.1.1 Roles en Scrum.....	67
8.1.2 Eventos en Scrum .....	68
8.1.3 Artefactos en Scrum.....	70
8.1.4 Definición de Terminado o de Hecho .....	71
8.1.5 Beneficios de Scrum .....	72
8.2 Kanban.....	73
8.2.1 Principios de Kanban .....	74
8.2.2 El Tablero Kanban .....	76
8.2.3 Límite de Trabajo en Proceso (WIP).....	77
8.2.4 Flujo y Métricas en Kanban .....	77

8.2.5 Ventajas de Kanban .....	78
8.2.6 ¿Cuándo usar Kanban? .....	79
8.3 Extreme Programming (XP).....	80
8.3.1 Principios de Extreme Programming .....	81
8.3.2 Prácticas de Extreme Programming .....	82
8.3.3 Prácticas de Gestión .....	84
8.3.4 Beneficios de Extreme Programming .....	85
8.3.5 Desventajas o Desafíos de XP .....	86
8.4 Test-Driven Development (TDD).....	87
8.4.1 Ciclo de TDD: Red-Green-Refactor .....	88
8.4.2 Beneficios de TDD .....	89
8.4.3 Desafíos de TDD .....	91
8.4.4 Herramientas de TDD .....	92
8.5 Metodología Lean.....	93
8.5.1 Principios de Lean.....	94
8.5.2 Tipos de Desperdicios de Lean .....	96
8.5.3 Prácticas Lean en el Desarrollo de Software .....	98
8.5.4 Beneficios de Lean en el Desarrollo de Software .....	99
8.5.5 Desafíos de Implementar Lean .....	100
8.6 Metodología Crystal.....	101
8.6.1 Principios de la Metodología Crystal .....	102
8.6.2 Las Variantes de Crystal.....	104
8.6.3 Ventajas de Crystal .....	106
8.6.4 Desventajas y Desafíos de Crystal .....	107
8.6.5 Comparación con Otras Metodologías Ágiles.....	108
8.7 Metodología FDD (Feature-Driven Development).....	109
8.7.1 Principios de FDD .....	110
8.7.2 Etapas del Desarrollo en FDD .....	112
8.7.3 Roles en FDD.....	114
8.7.4 Ventajas de FDD .....	115
8.7.5 Desventajas y Desafíos de FDD .....	116
8.7.6 Con relación a otras Metodologías Ágiles .....	117
<b>Módulo IX. Diferencias entre un Agile Coach y otros roles .....</b>	<b>120</b>
9.1 Agile Coach vs Scrum Master .....	120
9.2 Agile Coach vs Product Owner .....	120
9.3 Agile Coach vs Project Manager.....	120



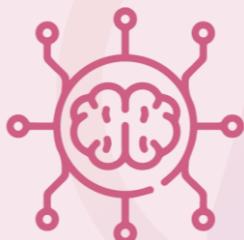
# Objetivos de la Certificación

## Objetivos de la Certificación



Contar con una visión general sobre la agilidad, principales prácticas, conceptualización del Coaching y las competencias fundamentales que un Agile Coach debe tener para ser un catalizador de cambio en las personas y en las organizaciones.

Aprender y comprender los Marcos Ágiles y los principios Lean, a nivel de las prácticas y de los principios y valores ágiles.



Desarrollar la capacidad de enseñanza para ofrecer el conocimiento correcto, en el momento adecuado, de la manera correcta, para que las personas, los equipos y las organizaciones asimilen el conocimiento para su mejor beneficio.

Certificarse internacionalmente como Agile Coach, avalando así sus conocimientos y aplicación de las buenas prácticas en este campo.





# Módulo I. Introducción

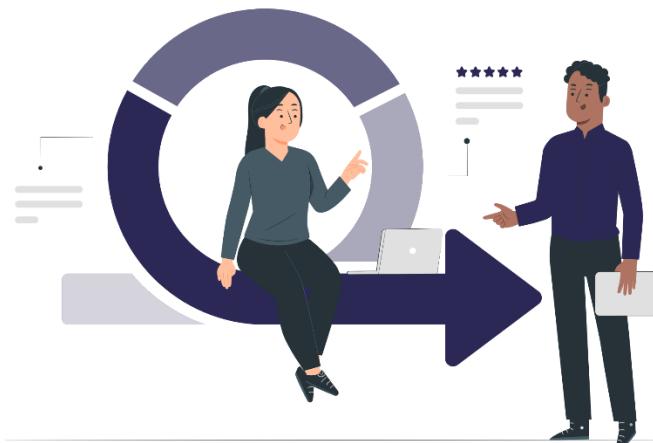
## Módulo I. Introducción

La Agilidad Empresarial está relacionada con la capacidad de una organización para adaptarse rápidamente a los cambios del entorno y evolucionar de manera eficiente con sus productos, servicios, procesos y prácticas en general.

La agilidad se ha constituido como una característica y una cualidad imprescindible para las organizaciones que buscan mantenerse en su entorno o mercado de manera exitosa. Las organizaciones ágiles están capacitadas para gestionar desafíos inesperados, responder más rápido a las necesidades de los consumidores, mejorar las estrategias actuales o generar nuevas en pro del cambio constante y de la oportunidad dentro de los mercados. Se enfocan en la capacidad para adaptarse al entorno a través de procesos simples y flexibles, pensamiento ágil y trabajo altamente competente.

Adicionalmente, las organizaciones deben lograr la agilidad operativa, es decir, la capacidad para reconfigurar sus estrategias, de manera dinámica, en sus operaciones. Dentro de este reto se pueden incluir características de la evolución ágil, como lo son:

- Implementar sistemas escalables y modulares que minimicen los tiempos de inactividad y las interrupciones en las operaciones de la organización.
- Mantener continuidad y eficiencia a través de operaciones más flexibles, enfocadas en la experiencia de los clientes, usuarios y consumidores.
- Asegurar que las soluciones sean resilientes para anticipar fallos, mejorar estrategias proactivamente y recuperarse de manera iterativa e inteligente.



- Ampliar la capacidad de innovación y creatividad enfocados, primero, en las personas.

Parte de la fórmula para lograr la agilidad empresarial implica contar con profesionales expertos en el pensamiento ágil, como el Agile Coach, enfocados en lograr una mentalidad que abrace los cambios, conlleve una comunicación abierta y transparente, impulse la colaboración auténtica y promueva el aprendizaje continuo y masivo.





# Módulo III. Manifiesto Ágil

## Módulo II. Manifiesto Ágil

El manifiesto ágil es un documento que resume en cuatro (4) valores y doce (12) principios las mejores prácticas para el desarrollo de software, basados en la experiencia de diecisiete (17) industriales del software, en procura de desarrollos más rápidos conservando su calidad.

### 2.1 Valores del manifiesto ágil



#### 1. Los individuos e interacciones por encima de los procesos y las herramientas



Para garantizar una mayor productividad, las metodologías ágiles valoran el talento humano como el principal factor de éxito. Reconocen que contar con recursos humanos calificados, con capacidades técnicas adecuadas, facilidades para adaptarse al entorno, trabajar en equipo e interactuar

convenientemente con los negocios y los clientes, da mayor garantía de éxito que contar con herramientas, políticas y procesos rigurosos.

Las metodologías ágiles reconocen que es más importante construir un buen equipo de trabajo que las herramientas y procesos. Primero se debe conformar el equipo de trabajo y que éste defina el entorno más conveniente de acuerdo con las necesidades y las circunstancias de los proyectos y operaciones de las organizaciones.



Los equipos que se apoyan en las metodologías ágiles apoyan la colaboración entre el equipo y trabajar juntos, de manera cohesionada. Lo importante es impulsar la comunicación abierta y efectiva entre las personas del equipo para se promueva la colaboración de unos con otros y, así, obtener los mejores resultados.

Claramente los procesos ayudan al trabajo, son una guía de operación y las herramientas mejoran

la eficiencia. Sin embargo, existen tareas que requieren talento y trabajo coequípero, basado en valores y actitudes adecuados.

## **2. Software y productos funcionando por encima de la documentación exhaustiva**

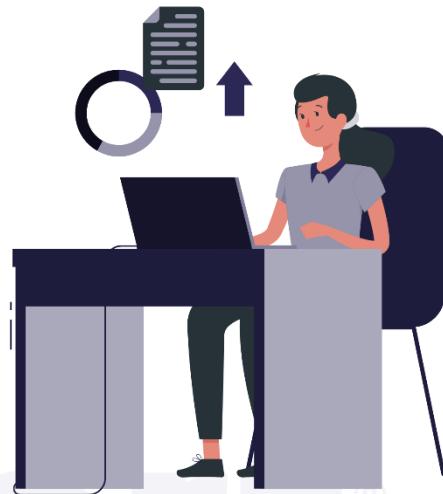
Lograr anticipar cómo funcionará el producto final, observando prototipos previos, o partes del producto ya construidas, ofrece un espacio de retroalimentación enriquecedor para todos los involucrados, generando ideas que permitan mejorar los entregables en la medida que se van incrementando en la ejecución de los proyectos. Esto sería muy difícil de lograr en



la definición de requisitos funcionales y no funcionales al inicio del proyecto.

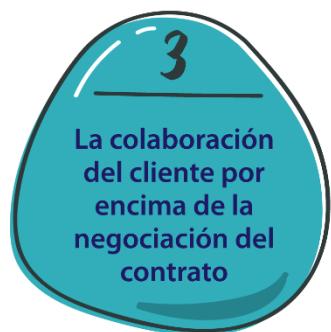
Los profesionales involucrados en el desarrollo de productos, aunque no necesariamente se especializan en producir documentos, si comprenden su importancia y relevancia, al igual que reconocen el tiempo y costo de mantener una gestión de conocimiento efectiva y actualizada.

Las metodologías ágiles respetan la importancia de la documentación y su aporte a la gestión del conocimiento, como parte del proceso y del resultado de los proyectos, por lo que enfatiza en que se deben generar los documentos estrictamente necesarios, concentrándose en una documentación precisa, limitándose a lo fundamental (pensamiento LEAN) y dando prioridad al contenido.



La documentación, en las metodologías ágiles procura mecanismos más dinámicos y menos costosos como son la comunicación personal, el trabajo en equipo, la auto documentación y los estándares.

Es relevante garantizar una documentación adecuada y de buena calidad, con la cual se apoya la gestión del conocimiento de toda organización, se soportan los hechos, se habilita la transferencia del conocimiento, se registra información histórica y se mantiene disponible para cualquier otro requerimiento o uso.

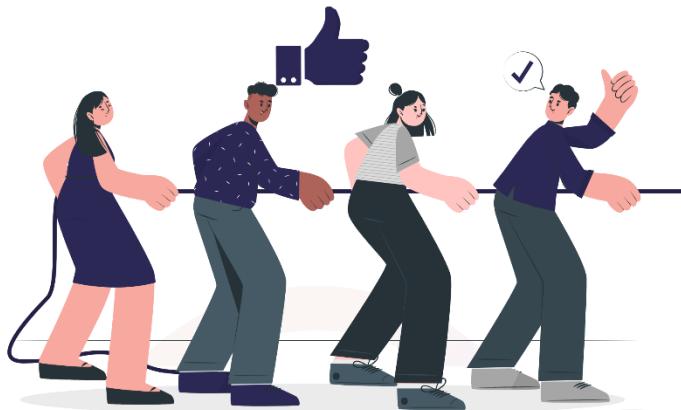


### **3. La colaboración del cliente por encima de la negociación del contrato**

El cliente o los negocios son quienes solicitan e indican qué debe hacer el software o el producto, y esperan los

resultados de acuerdo con sus necesidades y expectativas en los tiempos pactados y con los niveles de calidad establecidos.

Las metodologías ágiles incluyen de manera directa y comprometida al cliente o a los negocios en el equipo de trabajo, para contribuir activamente en la toma de decisiones y en las tareas donde se requiere su participación.



Cuando todos los involucrados comprenden que el trabajo en equipo, colaborativo, empático y activo, junto a los equipos de proyectos ágiles, se establecen bases hacia el éxito de los productos y servicios resultantes.

La colaboración implica el aporte de todos a un beneficio u objetivo común, por tanto, la responsabilidad colectiva para lograr esos resultados incluye a los interesados.

La participación del cliente o los negocios debe ser constante, desde el comienzo hasta la culminación del proyecto, y su interacción con el equipo, de excelente calidad.

Las prácticas ágiles están indicadas para productos cuyo detalle resulta difícil prever al principio del proyecto; y si se detallara al comenzar, el resultado final tendría menos valor que si se mejoran y precisan con retroalimentación continua.

En los proyectos, los clientes indican lo que necesitan o desean, y aunque son los más indicados para corregir o hacer recomendaciones, en el ámbito ágil se impulsa a que los miembros de los equipos de proyectos, quienes son especialistas en lo que hacen, también lo realicen en cualquier momento del proyecto.

El objetivo de un proyecto ágil no es controlar la ejecución conforme a procesos y cumplimiento de planes, sino proporcionar el mayor valor posible al producto. Entonces

resulta más adecuada una relación colaborativa y continua con el cliente, más que una contractual que delimita responsabilidades.

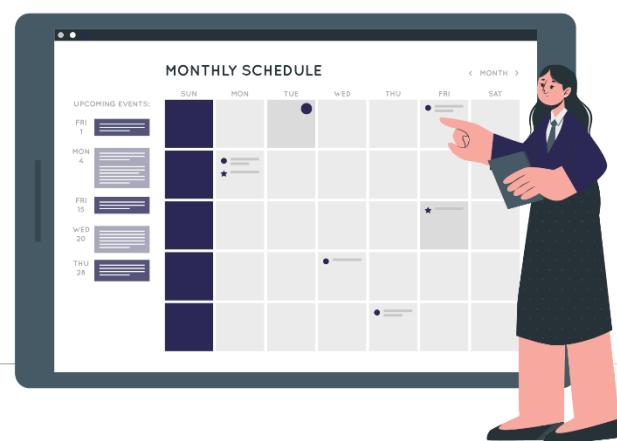
#### 4. La respuesta al cambio por encima del seguimiento de un plan

Dada la naturaleza cambiante de la tecnología y la dinámica de la sociedad moderna, un proyecto se enfrenta con frecuencia a cambios durante su ejecución. Van desde ajustes sencillos en la personalización del producto hasta cambios en las leyes, pasando por la aparición de nuevos productos en el mercado, comportamiento de la competencia, nuevas tendencias tecnológicas, entre otros.



En este sentido, las metodologías pesadas con frecuencia caen en la idea de tener todo completo y correctamente definido desde el comienzo. No se cuenta entre sus fortalezas la habilidad para responder a los cambios.

En las metodologías ágiles la planificación no debe ser estricta, puesto que hay muchas variables en juego, debe ser flexible para poder adaptarse a los cambios que puedan surgir.



Una buena estrategia es hacer planificaciones detalladas para unas pocas semanas y planificaciones mucho más abiertas para los siguientes meses. Para desarrollar productos de requisitos inestables, que tienen como factor inherente el cambio y la evolución rápida y continua, resulta mucho más valiosa la capacidad de respuesta que la de

seguimiento y aseguramiento de planes. Los principales valores de la gestión ágil son la anticipación y la adaptación.

## 2.2 Principios del manifiesto ágil



1. Nuestra mayor prioridad es **satisfacer al cliente** mediante entregas tempranas y continuas de software con valor. Para que una metodología puede ser calificada como ágil debe empezar a entregar software funcionando y útil en pocas semanas. Esto acaba con la incertidumbre, desconfianza, insatisfacción y desmotivación producidas en el cliente debido a las largas esperas para ver resultados concretos. Por lo tanto, la participación del cliente se hace más productiva en la medida en que el software está siendo probado, revisado y aprobado constantemente por quien lo requirió y lo va a usar
2. **Bienvenidos los cambios a los requerimientos, incluso los tardíos.** Los procesos ágiles aprovechan los cambios para la ventaja competitiva del cliente. Es ambicioso esperar que el cliente defina de manera definitiva todos sus requerimientos desde el comienzo y peor aún depender de ello para adelantar el

proyecto. Los cambios en los requerimientos deben asumirse como parte del proceso de maduración del software, debe entenderse que cuando el cliente describe una necesidad lo hace desde su perspectiva de usuario y que sus conocimientos técnicos lo pueden limitar para hacerse entender completamente.

Por lo tanto, las novedades en los requerimientos pueden ser ajenas a la voluntad del cliente. Esta forma de ver los cambios en los requerimientos induce al equipo de desarrollo a preferir los diseños flexibles, lo cual aumenta la satisfacción del cliente y redunda finalmente en beneficio del equipo de desarrollo dada la comodidad en el diagnóstico y ajustes que se requieren en la etapa de mantenimiento.

**3. Liberar frecuentemente software funcionando, desde un par de semanas a un par de meses, con preferencia por los períodos más cortos.**

El cliente siempre espera ver funcionando el programa, y es eso lo que debe entregársele. Pocas veces resulta conveniente, después de varios meses de trabajo, entregar sólo informes, modelos abstractos y planes. Se deben entregar resultados que incluyan software que el usuario pueda ver trabajando. Si hay una circunstancia que motiva al cliente es poder usar el software que solicitó.

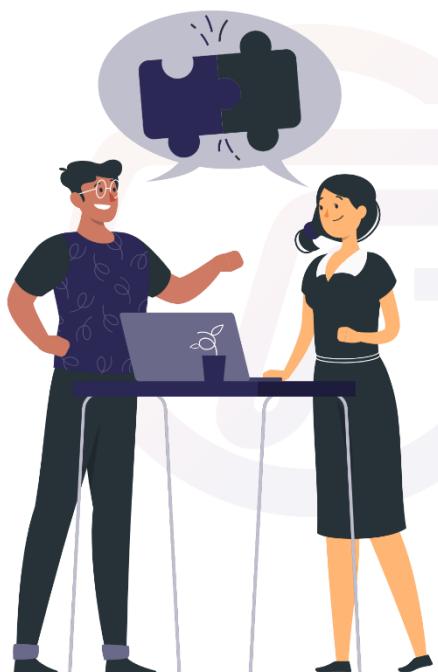


**4. Las personas del negocio y los desarrolladores deben trabajar juntos diariamente a lo largo del proyecto.**

Si bien el usuario desconoce lo referente al lenguaje, el diseño de bases de datos, protocolos y demás aspectos técnicos, es él, quien nos puede señalar qué está bien desde el punto de vista de la funcionalidad y resultados entregados por el software. La intervención oportuna del usuario puede resultar decisiva en el éxito de un proyecto y puede reducir el costo o el tiempo. Esta

intervención puede ser en cualquier momento, por lo cual el usuario debe estar involucrado todo el tiempo que dure el proyecto.

**5. Construir proyectos en torno a individuos motivados.** Darles el entorno y apoyo que necesiten, y confiar en ellos para que consigan hacer su trabajo. El ánimo, el sentido de pertenencia y la disposición del equipo de trabajo son fundamentales en un proyecto de software. Parte de la motivación está en la confianza que se muestre en el equipo de trabajo, el respeto por sus aportes y la comodidad que se les conceda en el momento de realizar su trabajo. Todo lo que se pueda hacer por dar ánimo y motivación a las personas participantes en el proyecto debe hacerse.



**6. El método más efectivo y eficiente de compartir información dentro de un equipo de desarrollo, es la conversación cara a cara.** El trabajo en equipo debe apoyarse con un buen sistema de comunicación tanto entre los miembros del equipo de desarrollo como entre éstos y el usuario. La mejor forma de hacerlo es hablando personalmente; en la medida en que se evitan los intermediarios en el proceso de comunicación, como son el papel, el teléfono, el sistema de correo, y demás medios de comunicación, se incrementa la posibilidad de que el resultado sea el que se solicitó.

**7. El software funcionando es la medida de progreso.** Cuando se trata de establecer el estado de un proyecto, si bien existen diversas formas de medirlo, es la cantidad de requerimientos implementados y funcionando la que más claridad y confiabilidad ofrecen para establecer una medida del avance del proyecto. Cualquiera otra que se presente será superada por una que involucre el software qué ya ha sido probado y aprobado por el usuario.

**8. Los procesos ágiles promueven el desarrollo sostenible.** Los patrocinadores, desarrolladores y usuarios deberían ser capaces de mantener relaciones cordiales. Se debe trabajar de forma que lo urgente no se imponga sobre lo importante. Desde el inicio del proyecto se debe asignar responsabilidades y tareas de manera que siempre se puedan cumplir.

**9. La atención continua a la excelencia técnica y al buen diseño incrementan la agilidad.** Además de satisfacer los requerimientos del usuario, los aspectos técnicos deben ser excelentes, independientemente de su cantidad y complejidad. La calidad debe ser vista desde dos perspectivas, la del usuario y la del equipo desarrollador. Para el personal técnico resulta evidente que cuanta más calidad tenga el software en cuanto a diseño y estándares de implementación, más rendimiento obtiene en las tareas de pruebas, mantenimiento, y mayor reusabilidad.

**10. La simplicidad como el arte de maximizar la cantidad de trabajo no hecho, es esencial.** Se estima que el cliente nunca usará el 90% de la funcionalidad que se implementa sin que esta haya sido solicitada. Se deben centrar los esfuerzos en lo que realmente importa, de manera simple, sin excederse en refinamientos y optimizaciones innecesarias. Si funciona así, déjelo así, si se va a perfeccionar u optimizar una rutina o programa se debe evaluar minuciosamente el costo beneficio.

**11. Las mejores arquitecturas, requerimientos y diseños emergen de los equipos auto organizados.** Los principios que ríjan en equipo de trabajo deben surgir de su interior, los ajustes, estructuras administrativas deben formularse con la participación de todo el equipo teniendo siempre presente el bien colectivo, la responsabilidad es de todos.



**12. En intervalos regulares, el equipo reflexiona sobre cómo volverse más efectivo, entonces afina y ajusta su comportamiento como corresponde.** El equipo de trabajo está todo el tiempo dispuesto a cambiar lo que sea necesario para mejorar. En cada tarea siempre existe la posibilidad de hacerlo mejor la próxima vez.

El Manifiesto Agile determinó un punto de inflexión para las organizaciones y las personas, impulsando principios y pilares como la colaboración, la adaptabilidad, la flexibilidad y el foco en el valor para los clientes o negocios. Estas bases reformularon la manera en que se aborda la gestión de proyectos y la ejecución de estrategias en las empresas.

Hoy día la agilidad es una necesidad, ya no es opcional si las organizaciones quieren mantenerse a flote y ser diferenciadoras, en medio de las complejidades e incertidumbres del mercado.





# Módulo III. Coaching

## Módulo III. Coaching

El coaching es un proceso de acompañamiento o entrenamiento personalizado que tiene como objetivo ayudar a una persona, a quien se denomina Coachee, a alcanzar sus metas, mejorar su rendimiento o resolver problemas específicos.

El Coach guía al Coachee a descubrir sus propias respuestas, a tomar decisiones y a accionar de manera efectiva, para lograr sus objetivos, a través de conversaciones estructuradas y técnicas específicas.



El coaching se enfoca principalmente en el desarrollo personal y profesional, y se diferencia de otras formas de apoyo, en que el coaching no se centra en tratar problemas emocionales profundos ni en ofrecer soluciones directas, sino en empoderar a la persona para que sea capaz de encontrar sus propias soluciones y avanzar en su proceso de autodescubrimiento.

### 3.1 Tipos de Coaching



© T-CERT®



Coaching personal o de vida  
(Life Coaching)

**Coaching personal o de vida (Life Coaching):** Enfocado en el bienestar personal, el crecimiento emocional y la realización de objetivos en la vida cotidiana.



Coaching ejecutivo



Coaching de carrera

**Coaching de carrera:** Ayuda a las personas a gestionar su trayectoria profesional, decidir sobre cambios de carrera o alcanzar objetivos laborales específicos.

**Coaching de equipos:** Aplica principios de coaching en el contexto de un equipo de trabajo, buscando mejorar la colaboración, comunicación y el desempeño colectivo.



Coaching de equipos

### 3.2 Principios clave para realizar Coaching



Establecer una relación de confianza



Definir objetivos claros



Explorar la situación actual



Identificar acciones y estrategias



Promover la responsabilidad



Evaluación y seguimiento



Uso de herramientas y técnicas

© T-CERT®



### Establecer una relación de confianza

**Establecer una relación de confianza:** La base del coaching es una relación de confianza entre el coach y el coachee. El coach crea un ambiente seguro y de respeto, donde el coachee se siente libre de expresar sus pensamientos y emociones sin ser juzgado.

**Definir objetivos claros:** Es fundamental que el coachee tenga claro qué quiere lograr. Los objetivos deben ser específicos, medibles, alcanzables, relevantes y en un tiempo definido (siguiendo el modelo SMART). El coach ayuda a clarificar y refinar estos objetivos de ser necesario.



### Definir objetivos claros



### Explorar la situación actual

**Explorar la situación actual:** Antes de trabajar hacia la meta, el coach y el coachee deben analizar la situación actual, cuál es la línea base, identificando obstáculos, creencias, limitantes y recursos disponibles. Tienen gran importancia los ejercicios de reflexión y las preguntas poderosas.



**Identificar acciones y estrategias:** El coach ayudará al coachee a diseñar un plan de acción que lo lleve hacia sus objetivos, lo cual implica toma de decisiones y planeación estratégica. Esto incluye

### Identificar acciones y estrategias

establecer pasos concretos, plazos y recursos necesarios para lograr los resultados esperados.



#### Promover la responsabilidad

**Promover la responsabilidad:** El coaching se basa en que el coachee es el responsable de su propio avance y cambio. El coach actúa como guía y facilitador, pero es el coachee quien debe comprometerse a las acciones necesarias. El coach ayuda a promover la responsabilidad fomentando la reflexión y la toma de decisiones autónoma.

**Evaluación y seguimiento:** Durante el proceso, se deben realizar revisiones periódicas para evaluar el progreso del coachee hacia sus metas y ajustar las estrategias de ser necesario. El seguimiento se puede realizar con sesiones periódicas de inspección y retroalimentación.



#### Evaluación y seguimiento



#### Uso de herramientas y técnicas

**Uso de herramientas y técnicas:** El coach puede utilizar diversas herramientas y técnicas para facilitar el proceso, como:

- **Preguntas poderosas** para promover la reflexión
- **Escucha activa** para comprender completamente al coachee
- **Visualización y técnicas de mindfulness** (Atención Plena o Conciencia Plena) para fomentar la claridad mental
- **Ejercicios de metas y planificación** para proyectar y validar los objetivos definidos y los resultados esperados.
- **Retroalimentación constructiva** para motivar y mejorar el rendimiento.

### 3.3 Consideraciones importantes para una implementación exitosa



Formación y habilidades del coach



Ética y límites



Medir el éxito



Adaptabilidad

© T-CERT®



Formación y habilidades del coach

**Formación y habilidades del coach:** Para ser efectivo, un coach debe tener formación adecuada y habilidades en áreas como escucha activa, comunicación efectiva, empatía y liderazgo. Los coaches suelen tener certificaciones y formación especializada, como por ejemplo las certificaciones emitidas por la ICF International Coaching Federation.

**Ética y límites:** El coach debe mantener la confidencialidad y actuar siempre con integridad y ética, asegurándose de que los intereses del coachee estén por encima de cualquier asunto. También debe ser consciente de cuándo el coaching no es adecuado y referir al coachee a otro profesional (por ejemplo, un terapeuta) si se detectan problemas emocionales profundos o cualquier otro indicio.



Ética y límites



Medir el éxito

**Medir el éxito:** El progreso del coaching debe ser medible, aunque el éxito no siempre es cuantificable de manera directa. Los indicadores pueden incluir el logro de objetivos, el aumento de la confianza, el desarrollo de habilidades o la mejora del rendimiento.

**Adaptabilidad:** El proceso con el Coachee debe ser flexible y ajustarse a las necesidades cambiantes. Cada persona es única, por lo que el coach debe estar en capacidad de adaptar sus métodos y enfoques según el contexto y los resultados que se vayan obteniendo, de tal forma que se puedan mejorar las técnicas y partes del proceso en la medida que se avanza.



En pocas palabras, el coaching es una herramienta poderosa para el desarrollo personal y profesional, que permite a las personas alcanzar sus objetivos a través de la reflexión, la toma de decisiones y la acción.



# Módulo IV. Agile Coach

## Módulo IV. Agile Coach

### 4.1 ¿Qué es un Agile Coach?

Un Agile Coach es un facilitador y guía que ayuda a las organizaciones, equipos y personas a adoptar, aplicar y mejorar los procesos ágiles en su trabajo. Su papel se enfoca en avanzar la cultura organizacional, los procesos y la mentalidad de los equipos para alinearse con los principios de Agile y de esta manera generar grandes beneficios a la organización y a todos aquellos que buscan la agilidad.

Este rol es clave en las transformaciones ágiles de las organizaciones. Gracias a su profundo conocimiento en agilidad y sus cualidades como líder servicial, apoya la evolución de la organización a través de la facilitación, la formación, la mentoría y el coaching.



El Agile Coach se enfoca en las personas, en el desarrollo de los equipos y en últimas de la organización. Trabaja muy cerca de las personas, pero también del gobierno organizacional para lograr la implementación de la cultura ágil.

Un Agile Coach está identificando, de forma permanente, a las personas interesadas en el proceso de ser ágiles, dentro y fuera de la organización. Une a los equipos y a las organizaciones a través de un pensamiento, una filosofía ágil.

En definitiva, un Agile Coach es un profesional especializado en facilitar y promover la adopción de metodologías ágiles en una organización, guiando a los equipos, y a la empresa, hacia un pensamiento y prácticas más ágiles. Con ello se puede mejorar la eficiencia, la calidad y la satisfacción de los empleados y de los clientes.

## 4.2 Misión de un Agile Coach

Un Agile Coach está orientado a ayudar, guiar, proponer y hacer mentoring a las personas para adoptar la agilidad en dos vías:



El alcance del rol dependerá de lo que cada organización requiera o defina. Puede ir, desde trabajar a nivel estratégico para implementar la agilidad a nivel empresarial, hasta trabajar a nivel de equipos y, por lo tanto, tener un alcance más acotado.

Cualquiera sea el caso, la naturaleza del trabajo no cambia, porque de lo que se trata en última instancia es de ayudar a las personas y organizaciones a adaptarse a los cambios rápida y continuamente. Por eso y, en definitiva, un Agile Coach es un Agente de Cambio.

Los agile coaches son especialmente útiles durante las transformaciones ágiles, cuando una organización está adoptando agile por primera vez. También pueden ser útiles para ayudar a los equipos existentes a mejorar sus prácticas. Ayudan a maximizar el rendimiento y a evitar los errores comunes que suelen cometerse en las implementaciones ágiles. Además, puede entrenar la habilidad de una organización para adaptarse a los cambios constantes, convirtiendo estos cambios en una ventaja competitiva.

Un agile coach puede ayudar a un equipo a mejorar su rendimiento, a resolver problemas y conflictos, y a adoptar prácticas ágiles más efectivas. A nivel de proyecto, un agile coach puede ayudar a asegurar que el proyecto se ejecuta de manera ágil, lo que puede llevar a una entrega más rápida y a una mayor satisfacción del cliente.

Un agile coach entiende que las organizaciones son sistemas complejos adaptativos, y las implicaciones que esto conlleva. Los sistemas complejos se componen de varios agentes que interactúan entre sí y con su entorno, y cambian y se adaptan según las circunstancias.

En una organización los agentes son las personas, los equipos, los departamentos, y en su entorno se incluyen otros interesados como los clientes, competidores, regulaciones, entre otros.

Lo que hace que estos sistemas sean complejos es que las interacciones entre todos estos elementos pueden dar lugar a comportamientos emergentes, es decir, comportamientos que no se pueden prever simplemente conociendo las propiedades de los componentes del sistema, lo que genera un entorno de incertidumbre, que requiere una gestión de equipos cohesionados, interesados involucrados y prácticas ágiles bien aplicadas.



### 4.3 El Agile Coach

Guía a los equipos en la implementación de prácticas flexibles y en la mejora continua de los procesos para facilitar la adaptación ágil. Estas prácticas ágiles permiten a las organizaciones adaptarse rápidamente a los cambios del mercado, adquirir conocimientos sobre las tecnologías emergentes y las necesidades de los clientes.

Trabaja en estrecha colaboración con los equipos para identificar y eliminar obstáculos que puedan afectar su rendimiento. A través de sesiones de coaching individual y en grupo, ayuda a los miembros a desarrollar habilidades de colaboración, comunicación y resolución de problemas, lo que contribuye a un aumento de la productividad y la calidad del trabajo.



El Agile Coach promueve los valores y principios ágiles, como la transparencia, la colaboración, la adaptabilidad y el enfoque en el cliente. Trabaja con los líderes de la organización para crear un entorno que fomente la experimentación, el aprendizaje continuo y la autonomía de los equipos. Además, ayuda a comprender que la transformación ágil va más allá de la implementación de procesos y herramientas, implica un cambio cultural en toda la organización.

Orienta a los equipos hacia una mentalidad centrada en el cliente, ayudándoles a priorizar y entregar el valor de manera incremental y constante, a través de mecanismos como ofrecer productos y servicios de mayor calidad en menos tiempo, satisfacer las necesidades y expectativas de los clientes de manera más efectiva y adaptadas a sus necesidades reales.

Representa un catalizador para el cambio, un líder que guía a las organizaciones a través de la complejidad y la incertidumbre de la era digital, a través de su enfoque en los valores y principios ágiles y su habilidad para trabajar con sistemas complejos.

#### 4.4 Responsabilidades de un Agile Coach



##### Entrenamiento y mentoría



- Enseñar los conceptos básicos de Agile a equipos y líderes.
- Guiar en la implementación de marcos ágiles como Scrum, Kanban o XP.
- Acompañar a los equipos durante las primeras etapas de la transformación ágil.

##### Identificar áreas de mejora en los procesos actuales.

- Fomentar la auto-organización, autonomía y colaboración dentro de los equipos.
- Introducir y facilitar retrospectivas para evaluar y ajustar el enfoque ágil regularmente.

##### Mejora continua



### Facilitador de cambio organizacional



- Ayudar a los líderes y gerentes a cambiar su estilo de liderazgo para apoyar la agilidad.
- Romper barreras o silos que impidan la adopción de Agile.
- Fomentar una cultura ágil a nivel organizacional, más allá de los equipos de desarrollo.

### Resolución de impedimentos



- Identificar obstáculos que limitan la agilidad de los equipos.
- Colaborar con stakeholders para resolver bloqueos y facilitar la entrega de valor.

### Promover la mentalidad ágil



- Enseñar a las personas a ser flexibles ante el cambio, colaborar de manera efectiva y enfocarse en la entrega de valor continuo.
- Ayudar a cambiar la mentalidad de los equipos de una estructura jerárquica tradicional a una mentalidad de auto-gestión.

Debe tener una sólida comprensión de las metodologías ágiles, así como experiencia práctica en su uso. También necesita habilidades de coaching, facilitación y liderazgo. En general, se combinan 5 funciones:

1. Consultor
2. Coach
3. Mentor
4. Facilitador
5. Entrenador.

#### 4.5 Habilidades de un Agile Coach



**Habilidades de comunicación efectiva:** Se debe comunicar claramente con personas de diferentes niveles dentro de la organización, desde miembros del equipo hasta altos directivos, para transmitir conceptos ágiles de manera comprensible, persuasiva y empática.

**Empatía y capacidad de escucha:** Para comprender las necesidades y preocupaciones de los equipos y de la organización en su conjunto, debe ser empático y tener habilidades sólidas de escucha activa. Realmente reconoce que la mejor manera de comprender a las personas y a los equipos es interactuando positiva y constructivamente, identificándose con ellos y compartiendo su sentir.

**Empatía y capacidad de escucha**



## Facilitación y resolución de conflictos



**Facilitación y resolución de conflictos:** actúa como facilitadora en reuniones y sesiones de trabajo, promoviendo la colaboración, la comprensión de las temáticas y ayudando a que los equipos y las personas logren resolver conflictos.

## Conocimiento técnico y experiencia práctica:

aunque no es necesario que sea un experto en todas las áreas técnicas, debe tener un buen entendimiento de los principios y prácticas ágiles, así como experiencia práctica en entornos empresariales.

## Conocimiento técnico y experiencia práctica



## Liderazgo y motivación



**Liderazgo y motivación:** debe actuar como un líder, y tener claro que el liderazgo en el ámbito ágil es servicial y constructivo, así como motivar a los miembros del equipo a través de su ejemplo, para inspirar y guiar a los equipos hacia la agilidad.

## 4.6 Características de un Agile Coach

- Cuenta con experiencia en el uso de metodologías y procesos ágiles.
- Comprende profundamente los valores y principios del Manifiesto Ágil.
- Se comunica efectiva y empáticamente.
- Cuenta con un pensamiento lateral, basado en la creatividad, para la resolución de problemas y desafíos.
- Es paciente y resiliente para enfrentar los obstáculos y dificultades durante todo el proceso.

- Guía y motiva a los equipos hacia la adopción de prácticas ágiles.
- Tolera el fracaso, entendiendo que el aprendizaje se obtiene a través de la experimentación y la mejora continua.
- Orienta a los equipos hacia el aprendizaje y la auto-organización.
- Identifica cuándo guiar, cuándo enseñar y cuándo permitir que los equipos tomen sus propias decisiones.
- Promueve la comunicación abierta y la colaboración.
- Leer las situaciones a través de la observación cuidadosa y objetiva.
- Se preocupa por la gente más que por el producto.
- No asume nada, pregunta poderosamente, es curioso y objetivo.
- Considera que las personas son bien intencionadas.
- No admite el desperdicio, tiene claro el pensamiento Lean.
- Sabe que es necesario algo de caos y desequilibrio para conseguir los resultados esperados.
- No tiene miedo de equivocarse, y lo admite ante los equipos.
- Un Agile Coach es capaz de transformar a varios equipos y sabe cómo llevarlos a un nivel de alto rendimiento en diferentes contextos y formas.
- Persigue la excelencia del equipo.
- Está enfocado en la evolución y la mejora continua.
- Se adapta a la cultura organizacional para trabajar efectivamente con los equipos en las diferentes áreas y niveles.



## 4.7 Recomendaciones para dar la bienvenida a un Agile Coach



© T-CERT®



**Formación y capacitación:** proporcionar al Agile Coach la formación y la capacitación necesarias sobre la empresa, su cultura, sus procesos y sus objetivos estratégicos.

**Asignación de un mentor:** asignar a un mentor dentro de la organización que pueda orientar y apoyar al Agile Coach en sus primeros meses. El mentor puede ayudarlo a comprender mejor la dinámica organizacional y a superar cualquier desafío que pueda surgir durante su integración.



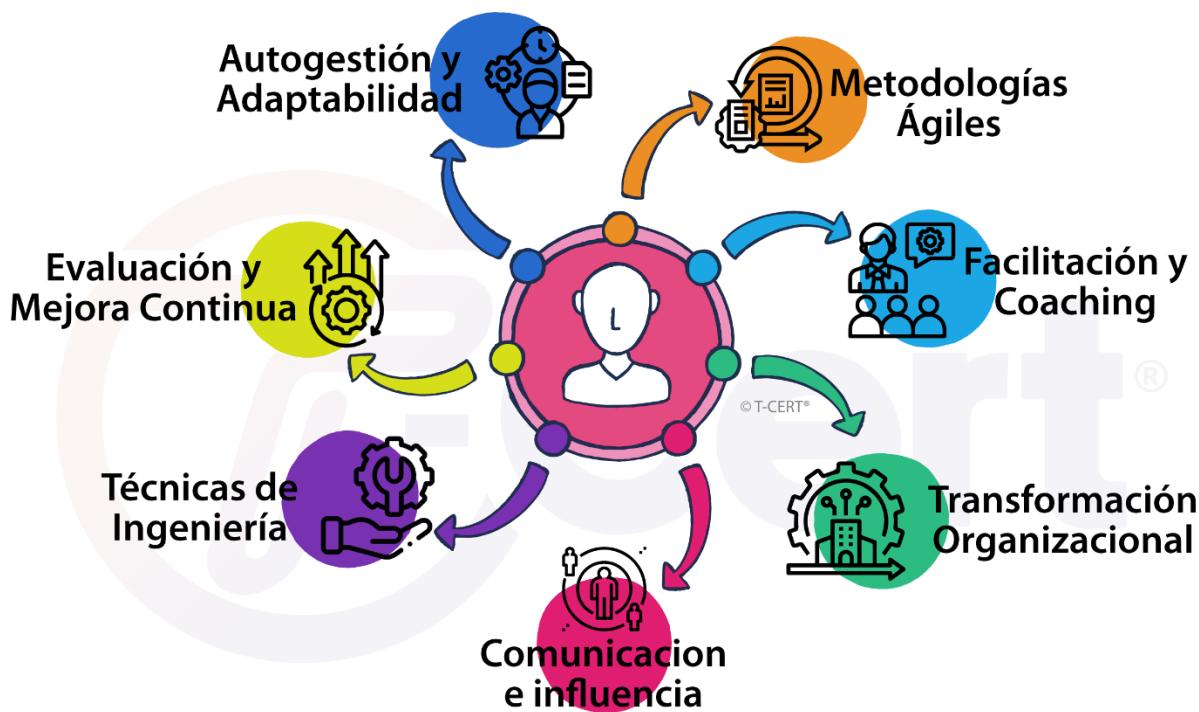
**Expectativas claras:** también es importante definir claramente las expectativas y los objetivos del Agile Coach desde el principio, así como los indicadores clave de rendimiento que se utilizarán para evaluar su éxito.



# Módulo V. Competencias de un Agile Coach

## Módulo V. Competencias de un Agile Coach

Las competencias de un Agile Coach van mucho más allá de entender las metodologías ágiles pues requiere una combinación de habilidades técnicas, interpersonales, comunicativas y de liderazgo para guiar a los equipos y a las organizaciones en su transición hacia la agilidad.



### 5.1 Competencias Técnicas en Metodologías Ágiles

#### 5.1.1 Conocimiento profundo de las metodologías ágiles



Un Agile Coach debe tener un conocimiento profundo de las metodologías ágiles más populares, así como la capacidad de adaptar esas prácticas a las necesidades específicas de la organización.

Además, debe ser capaz de comprender cómo se interrelacionan y cómo combinar elementos de estas metodologías cuando sea necesario, promover los valores de la agilidad como la transparencia, la colaboración, la apertura al aprendizaje a los cambios, la inspección o monitoreo, la adaptación, la gestión de impedimentos, entre otros.

- **Scrum:** claridad acerca de los eventos, roles y artefactos de Scrum.
- **Kanban:** conocimiento acerca de la gestión de flujo, visualización del trabajo y mejora continua.
- **Lean:** conocimiento de la gestión de desperdicios, mejora continua (Kaizen), y valor para el cliente.
- **XP (Extreme Programming):** entender prácticas de ingeniería como TDD (Test-Driven Development), Pair Programming, Refactorización e Integración continua.



### 5.1.2 Conocimiento de herramientas ágiles

El coach debe estar familiarizado con herramientas de gestión ágil para facilitar la implementación de prácticas como planificación de sprints, gestión de tareas, seguimiento de rendimiento y más.

Ejemplos de herramientas:



- **Gestión de tareas y proyectos ágiles:** Jira, Trello, Asana, Azure DevOps.
- **Mapas visuales de trabajo:** Miro.
- **Documentación colaborativa:** Confluence.

## 5.2 Competencias de Facilitación y Coaching

Como Facilitador se busca guiar, de manera neutral, a un grupo de personas en la interacción y colaboración para que encuentren soluciones y tomen decisiones; y como Coach se busca acompañar a las personas en un proceso reflexivo y creativo de aprendizaje donde desarrollan su máximo potencial personal y profesional.



### 5.2.1 Habilidades de facilitación

Un Agile Coach necesita ser un excelente facilitador de reuniones y actividades de equipo. Su función es guiar a los equipos en la aplicación de prácticas ágiles y asegurar que los eventos (como las retrospectivas, planificación de sprint y otros) sean productivos y beneficiosos.

- **Habilidad para crear un ambiente seguro:** Fomentar la colaboración abierta y la participación de todos los miembros del equipo.
- **Capacidad para guiar discusiones:** Mantener las conversaciones enfocadas y efectivas.
- **Técnicas de resolución de conflictos:** Ayudar a resolver desacuerdos o malentendidos de forma constructiva.

### 5.2.2 Habilidad de mentoría



Consiste en compartir conocimientos, acompañar, tutorizar y aconsejar desde la propia experiencia y conocimiento, y fomentar así el crecimiento personal y profesional de las personas.

### 5.2.3 Coaching de equipos y personas

Un Agile Coach también debe tener habilidades para coaching individual y de equipo. Las personas dentro de los equipos necesitan apoyo para adoptar nuevas formas de trabajar y cambiar su mentalidad.

- **Coaching en liderazgo ágil:** Guiar a los miembros del equipo (como Scrum Masters o Product Owners) para que asuman sus roles con efectividad.
- **Coaching de habilidades blandas:** Ayudar a los miembros del equipo a desarrollar habilidades interpersonales como la comunicación, colaboración, gestión de conflictos e inteligencia emocional.



## 5.3 Competencias de Transformación Organizacional



La capacidad en transformación requiere de experiencia práctica en acompañar y catalizar procesos de cambios, aportando una mirada sistémica. Un Agile Coach es un Agente de Cambio.

### 5.3.1 Habilidad para gestionar el cambio

Un Agile Coach no solo debe trabajar con equipos individuales, sino también con toda la organización. Deberá gestionar la transformación ágil a nivel organizacional, lo que incluye cambiar mentalidades, estructuras y procesos que no están alineados con los principios ágiles.

- **Evaluación de la madurez ágil:** Ayudar a la organización a evaluar su nivel de madurez ágil e identificar las áreas que necesitan desarrollo.

- **Gestión de la resistencia al cambio:** Los equipos y las personas pueden ser resistentes a adoptar nuevas formas de trabajo. El coach debe ser capaz de gestionar esta adversidad al cambio con empatía y estrategias efectivas.
- **Implementación de la cultura ágil:** Ayudar a integrar la mentalidad ágil en la cultura de la organización, asegurando que todos los niveles estén comprometidos con el cambio.

### 5.3.2 Liderazgo y gestión de la cultura organizacional



Un Agile Coach es un líder de cambio, capaz de alinear a los equipos y a los interesados con la visión y los valores ágiles. Esto incluye influir en la cultura organizacional y fomentar un entorno que valore la transparencia, la colaboración y la mejora continua.

- **Promoción de la transparencia:** Fomentar la apertura de las personas y la visibilidad de la información en todos los niveles.
- **Fomento de la autonomía y responsabilidad:** Ayudar a los equipos a adoptar autonomía en la toma de decisiones y a ser responsables de sus resultados, así como de las acciones a tomar en consecuencia.

## 5.4 Competencias de Comunicación e Influencia

### 5.4.1 Comunicación efectiva

La comunicación es clave en un entorno ágil. Un Agile Coach debe comunicarse de manera clara y efectiva con diversos grupos, incluidos equipos, líderes e interesados.



- **Escucha activa:** Escuchar con atención las preocupaciones, ideas y sugerencias de todos los miembros del equipo.
- **Claridad en los mensajes:** Explicar los conceptos ágiles de manera sencilla y comprensible para diferentes audiencias
- **Resolución de conflictos:** Abordar conflictos entre miembros del equipo o entre equipos de manera constructiva.

#### 5.4.2 Influencia y gestión de interesados

El Agile Coach también debe influenciar y trabajar con los interesados clave para asegurar que los objetivos ágiles estén alineados con las necesidades del negocio.

- **Alineación con el negocio:** Asegurarse que las iniciativas ágiles estén alineadas con los objetivos estratégicos de la organización.
- **Gestión de expectativas:** Ayudar a los interesados a comprender los beneficios y desafíos de la agilidad, y gestionar sus expectativas a lo largo del proceso.

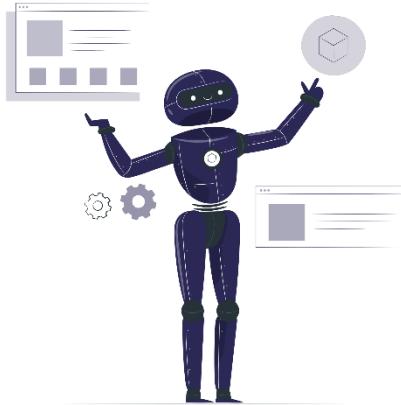


#### 5.5 Competencias Técnicas de Ingeniería



La maestría Técnica hace referencia a la experiencia y conocimiento en aspectos técnicos que específicamente se requieran en el área que se trabaje. Si bien no es un requisito obligatorio para todos los Agile Coaches, tener una base técnica sólida puede ser útil. Esto ayuda al coach a entender mejor los desafíos técnicos y a proporcionar soluciones ágiles más integradas.

### 5.5.1 Conocimiento de prácticas de desarrollo ágil



Como Desarrollo Dirigido por Pruebas (TDD), Integración continua (CI), y Automatización en los ciclos de desarrollo de soluciones digitales.

### 5.5.2 Capacidad para facilitar la colaboración entre equipos técnicos y no técnicos

Ayudar a los equipos de desarrollo y operaciones a trabajar de manera más integrada.

### 5.5.3 Conocimiento del negocio

Conocer el negocio que la organización forma parte, con foco en lo que valor e innovación significan para la industria en la que se trabaja.

## 5.6 Competencias en Evaluación y Mejora Continua

### 5.6.1 Medición del rendimiento ágil

El Agile Coach debe medir el progreso y la efectividad de los equipos utilizando métricas ágiles. No solo se trata de la velocidad o el tiempo de ciclo, sino de cómo el equipo está mejorando en la entrega de valor.



- **Métricas de rendimiento:** Como la velocidad, tiempo de ciclo, tasa de entrega de valor.
- **Análisis de cuellos de botella:** Identificar barreras en el flujo de trabajo y ayudar al equipo a solucionarlas.

### 5.6.2 Promoción de la mejora continua

El coach debe incentivar una mentalidad de mejora continua, lo que implica la retroalimentación constante, la experimentación, el aprendizaje y la toma de decisiones para ajustar lo que se requiera.

- **Facilitación de retrospectivas y revisiones de productos:** Ayudar a los equipos a reflexionar sobre su trabajo y buscar formas de mejorar su rendimiento y mejorar los productos y/o servicios que construyen.
- **Prácticas de Kaizen:** Promover la mejora continua en todos los aspectos del trabajo y en las partes de los procesos y prácticas donde se identifiquen oportunidades de mejora.



### 5.7 Competencias de Autogestión y Adaptabilidad



Un Agile Coach debe gestionar su propio trabajo de manera autónoma y eficiente, ya que frecuentemente trabajará en múltiples equipos o con varios interesados a la vez.

- **Gestión del tiempo:** Organizarse bien para poder atender múltiples necesidades y equipos.
- **Adaptabilidad:** Ajustarse a las circunstancias cambiantes y a las necesidades emergentes.

En definitiva, las competencias de un Agile Coach abarcan una combinación de habilidades técnicas, interpersonales, de facilitación y de liderazgo. No se trata solo de aplicar metodologías ágiles de manera correcta, sino también de ayudar a los equipos a

desarrollar una mentalidad ágil y una cultura organizacional que fomente la mejora continua, la colaboración, la toma de decisiones y la adaptabilidad.

Un Agile Coach exitoso es alguien que guía a los equipos no solo en el "cómo" de las prácticas ágiles, sino también en el "por qué", ayudando a crear un entorno de trabajo más ágil y dinámico.





# Módulo VI. Beneficios de contar con un Agile Coach

# Módulo VI. Beneficios de contar con un Agile Coach

## 6.1 Un agile coach en una organización

La presencia de un Agile Coach en una organización aporta beneficios significativos que contribuyen a la implementación de metodologías ágiles y en la adaptación a un entorno en constante cambio.



### 6.1.1 Mejora en la comunicación y colaboración entre equipos



Un Agile Coach trabaja activamente para fomentar una comunicación clara y abierta dentro de la organización, facilitando la colaboración efectiva entre diferentes áreas, equipos y otros interesados. El Agile Coach ayuda a superar silos organizacionales y a alinear los esfuerzos de todos los equipos hacia objetivos comunes, al promover la transparencia y el intercambio constante de información, entre otros aspectos.

### **6.1.2 Incremento en la eficiencia y productividad**

Incremento en la eficiencia y productividad de los equipos y la empresa en su conjunto, a través de la implementación de prácticas ágiles, como la planificación incremental, la entrega continua y la retroalimentación regular. Se trabaja de manera más eficiente y se genera valor de manera más rápida y consistente.



Incremento en la eficiencia y productividad

El Agile Coach contribuye a maximizar el uso de recursos y a reducir los tiempos de entrega al identificar y eliminar cualquier actividad o paso en un proceso que no agrega valor al producto final o servicio.

### **6.1.3 Fomento de una cultura de mejora continua**



Fomento de una cultura de mejora continua

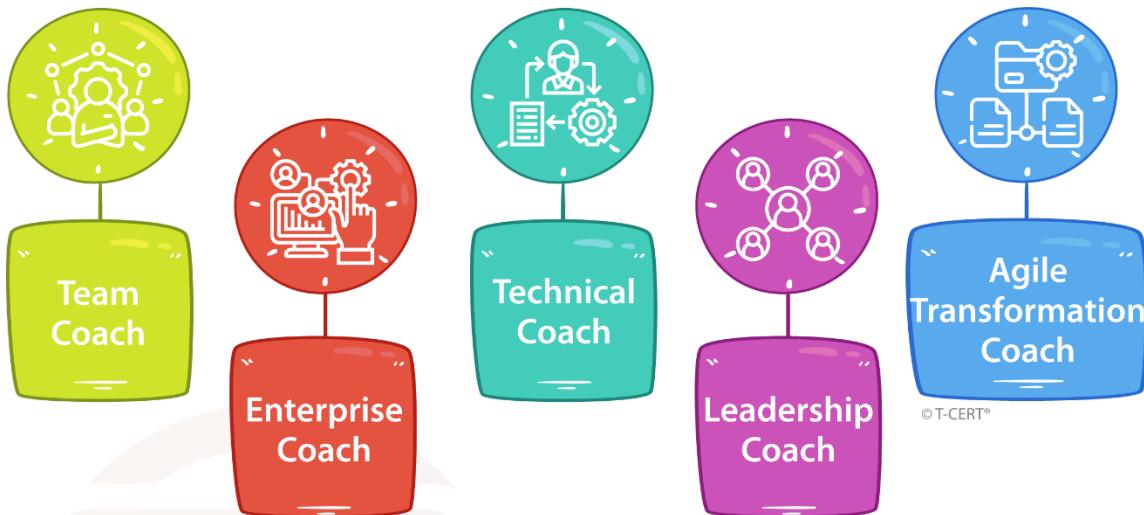
Un Agile Coach promueve un enfoque de aprendizaje constante y experimentación dentro de la organización, animando a los equipos a buscar formas de mejorar continuamente sus procesos, interacciones, productos, servicios y prácticas en general.

Al facilitar sesiones de retroalimentación y revisión periódicas, el Agile Coach ayuda a identificar áreas de oportunidad y a implementar cambios orientados a la innovación y la excelencia operativa. Por ende, esta cultura de mejora continua no solo impulsa el crecimiento y la adaptabilidad de la empresa, sino que también aumenta la satisfacción y el compromiso de los colaboradores e interesados.



## 6.2 Tipos de Agile Coach

Tipos de Agile Coach relevantes para las organizaciones:



### 6.2.1 Team Coach



Se enfoca en trabajar con equipos, ayudándolos a aplicar las prácticas ágiles, mejorar su rendimiento y mantenerlo en el tiempo. Se centra en ayudar a los equipos a mejorar su productividad y aplicar de manera efectiva las prácticas ágiles, como las retrospectivas, el refinamiento de backlog y la planificación de sprints.

- **Áreas de impacto:** Equipo de Proyecto, Product Owners, Scrum Masters e interesados.
- **Habilidades:** coaching de equipo, facilitación de sesiones y eventos, mejora continua, promoción de principios, pilares y valores ágiles.

## 6.2.2 Enterprise Coach

Trabaja a nivel organizacional, ayudando en la transformación ágil de grandes áreas de la organización, incluyendo equipos no relacionados con tecnología. Trabaja a nivel de la organización, ayudando a romper silos, mejorar la comunicación interdepartamental y facilitar la adopción de Agile en las áreas de la organización que se requieran.



- **Áreas de impacto:** Cultura, estructuras organizacionales y alineación de procesos.
- **Habilidades:** definición y alineación estratégica, transformación a nivel cultural y de procesos, alineación con políticas, prácticas y procedimientos.



## 6.2.3 Technical Coach

Centrado en las prácticas técnicas asegurando que la agilidad también esté presente en el lado técnico del desarrollo. Apoya a los equipos con las mejores prácticas técnicas, ayudando a mejorar la calidad del código y la entrega continua. Se involucra en aspectos técnicos como la automatización de pruebas y la integración continua, desde su conocimiento técnico y desde su dominio en agilismo.

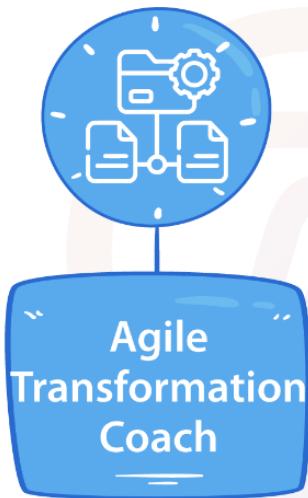
- **Áreas de impacto:** equipos de desarrollo de software, arquitectos, ingenieros de calidad.
- **Habilidades:** Prácticas técnicas ágiles, mentoría técnica, desarrollo de software.

#### 6.2.4 Leadership Coach

Guía a los líderes para que adopten un estilo de liderazgo basado en la facilitación y la creación de equipos autónomos y empoderados, apoyando una cultura de aprendizaje, adaptabilidad y agilismo.



- **Áreas de impacto:** directivos, gerentes, líderes funcionales y líderes en general.
- **Habilidades:** Empoderamiento de equipos, liderazgo, transformación cultural, facilitación, autogestión.



#### 6.2.5 Agile Transformation Coach

Coordina esfuerzos de transformación a gran escala, definiendo estrategias ágiles y alineando a toda la organización para adoptar marcos y prácticas ágiles. Este tipo de coach procura un apoyo global en la transición a Agile.

- **Áreas de impacto:** dirección general, equipos multifuncionales e interesados en general.
- **Habilidades:** Transformación cultural, estrategias para implementar agilismo y gestión y transición al cambio.



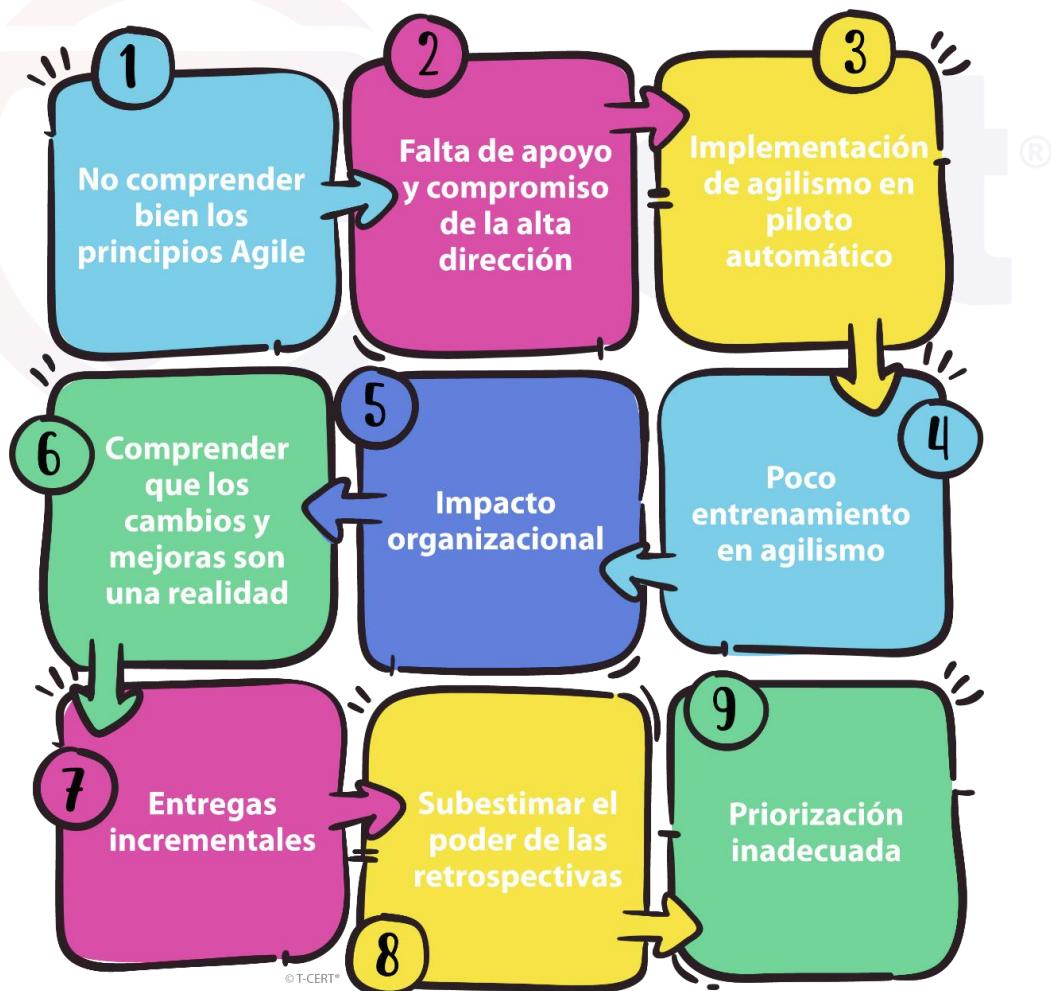


# Módulo VII. Errores comunes al adoptar metodologías ágiles

## Módulo VII. Errores comunes al adoptar metodologías ágiles

La adopción de metodologías ágiles supone un gran reto para las organizaciones, teniendo en cuenta que las empresas son únicas en su estructura, la forma como gestionan, la cultura, el clima, entre otros muchos factores.

Una vez se toma la decisión de trabajar con el agilismo, es posible cometer algunos errores, que pueden estar relacionados con la falta de compromiso del gobierno corporativo, la falta de entrenamiento, dificultades culturales y problemas de comunicación que impiden lograr una implementación ágil exitosa.



## **1. No comprender bien los principios Agile**

Uno de los errores más comunes es no entender completamente los principios ágiles antes de implementarlos. En ocasiones, las organizaciones se enfocan solo en las herramientas, las políticas, los procesos y los procedimientos, y se pierde el enfoque en la relevancia que implica cambiar la mentalidad y la cultura organizacional.



Es esencial instruir a todas las personas de la organización y su entorno, sobre el Manifiesto Ágil, sus valores y sus principios. Cada uno de estos elementos representan los pilares fundamentales del agilismo, inspiran las metodologías emergentes y las nuevas perspectivas empíricas sobre las cuales los líderes, los equipos y las organizaciones podrán adoptar las prácticas, basándose en un cambio cultural hacia la colaboración, la transparencia, la flexibilidad y la adaptación constante.

## **2. Falta de apoyo y compromiso de la alta dirección**

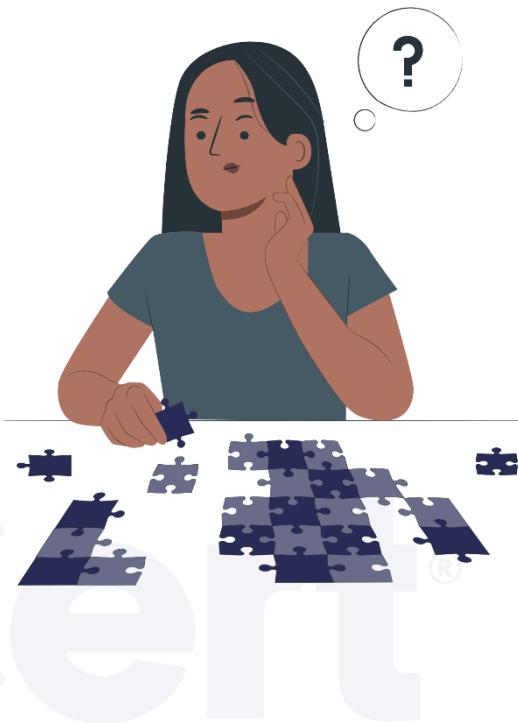
Eventualmente la alta dirección no está completamente comprometida con el cambio hacia el agilismo. La transición hacia el agilismo requiere el apoyo activo del gobierno, los directivos y los líderes de la organización para obtener resultados exitosos. Cuando no se cuenta con este respaldo, los equipos y en general las áreas con iniciativas ágiles enfrentarán grandes obstáculos para avanzar, lo cual conlleva a pérdida de credibilidad en la metodología.

Para emprender la agilidad empresarial, se requiere el compromiso total de la alta dirección. Los líderes se convierten en embajadores ágiles, asegurando los recursos necesarios para dichas prácticas y fomentando una cultura de agilidad en toda la organización, llevando a los equipos en una transición sistemática y paso a paso.

### **3. Implementación de agilismo en piloto automático**

Antes de implementar alguna o varias metodologías ágiles es importante considerar el contexto actual de la organización y entorno. Sus políticas, reglas, procesos, procedimientos, estructura, tipo de liderazgo, productos, servicios, cultura organizacional, entre otros.

Las metodologías ágiles deben ser adaptadas a las necesidades específicas de cada equipo y proyecto, así como de la organización. Las prácticas de la metodología deben ser adaptadas a la realidad de la organización y sus características.



### **4. Poco entrenamiento en agilismo**

Los entrenamientos en metodologías ágiles deben estar enfocadas en comprender muy bien la teoría y aplicar en la práctica mientras se aprende y se ajusta. Saltarse este pre-requisito llevará a los equipos a resultados negativos, baja en la moral de las personas, pérdida de credibilidad en los beneficios de las metodologías y abandono en el intento.

Parte de los legados de las prácticas ágiles es mantenerse en aprendizaje constante. Las personas, los equipos, la organización y su entorno. Se debe garantizar que las personas comprenden y viven el proceso paso a paso, experimentando, ajustando y observando resultados para identificar oportunidades de mejora.

## 5. Impacto organizacional

La transición a la agilidad no solo afecta a los equipos, sino a toda la organización. La construcción de productos y servicios de las organizaciones implica flujos de trabajo en todos los puntos de la organización, por ende, cuando se hacen adopciones ágiles, el impacto es proporcional.

Toda la organización y sus aliados deben involucrarse en la Gestión del Cambio Organizacional, comprender el enfoque ágil, la estrategia, los objetivos en diferentes plazos, y sobre todo, como cada uno contribuirá en el éxito de la adopción ágil.

## 6. Comprender que los cambios y mejoras son una realidad



En la implementación de proyectos, las organizaciones han trabajado en las últimas décadas con una baja aceptación de cambios y mejoras en sus productos y servicios. Con la llegada de las metodologías ágiles, se rompe este paradigma y se abraza el cambio. Es importante que todos comprendan que el alcance de los productos y servicios es flexible en relación con la mejora y el aporte de mayor valor para el cliente.

Cuando los equipos y los interesados son conscientes de dicha condición, logran implementar entregables que generan el máximo valor para los clientes o negocios, evolucionando a las organizaciones y comprometiendo aún más a los involucrados.

## 7. Entregas incrementales

En valor se debe entregar de manera iterativa o cíclica e incremental. Las metodologías ágiles rompieron el esquema de entregar resultados solo al finalizar los proyectos. Ahora, en cambio, se busca mostrar resultados exitosos a través de victorias tempranas, que revierte en altos niveles de satisfacción para los clientes y negocios. La organización y sus aliados percibirán beneficios como, productos funcionales (no completos, pero funcionales) desde etapas tempranas de los proyectos, captura de beneficios anticipada, detección temprana de errores y de manera primordial, alta calidad y adecuación de los entregables en la medida que se construyen.

## 8. Subestimar el poder de las retrospectivas

Con mucha frecuencia, las personas y las organizaciones consideran que las sesiones de retrospectiva son pérdida de tiempo. En realidad, las retrospectivas son una de las prácticas ágiles más importantes para asegurar la mejora continua en el proceso y la forma como se trabaja.

Las retrospectivas deben comprenderse como los espacios donde los equipos:

1. Se aíslan temporalmente del trabajo que vienen realizando y se disponen a realizar una inspección constructiva sobre el proceso.
2. Analizan las prácticas utilizadas en el último período y los resultados obtenidos.
3. Identifican qué resultó muy bien y qué tiene oportunidad de mejora.



4. Lo más importante, a manera de consenso, definen las acciones a implementar de inmediato para garantizar la mejora continua.

Durante las retrospectivas es importante lograr que los equipos reflexionen sinceramente sobre cómo están viviendo el proceso, así como identificar áreas de mejora y acciones que aseguren la evolución en cada uno de los ciclos de trabajo.

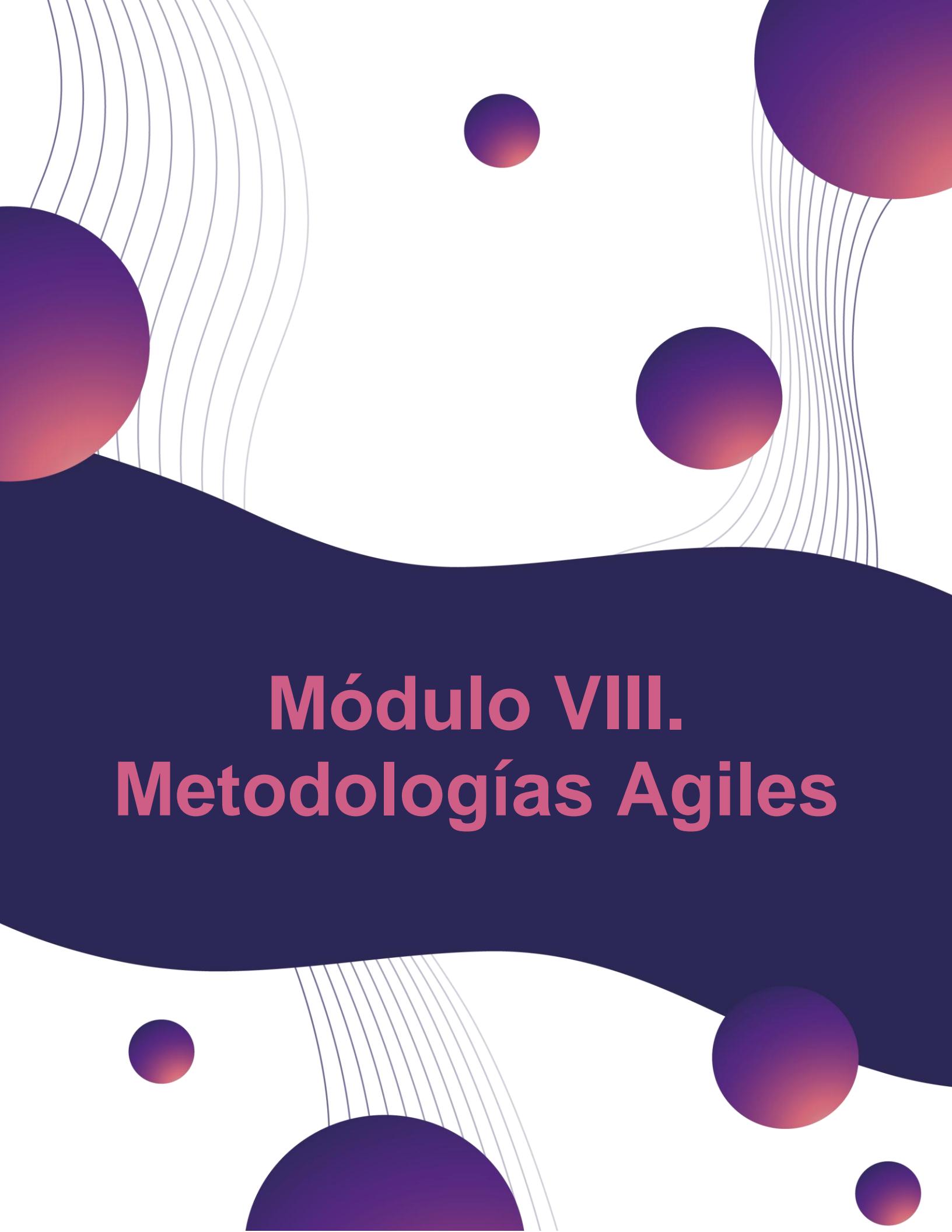
## 9. Priorización inadecuada

La gestión de prioridades es relevante para asegurar la entrega continua de valor al cliente. Dicha priorización debe centrarse en un único dueño, que pueda escuchar a sus partes interesadas y defina la prioridad de los requisitos.

No priorizar, además basándose en el valor, conlleva a bajos niveles de satisfacción del cliente, poca o nula captura de valor y productos o entregables no eficientes.

Para apoyar la priorización existen técnicas como MoSCoW, Marco RICE, Modelo Kano y Refinamiento del Product Backlog, en las cuales es importante la participación activa de las partes interesadas.





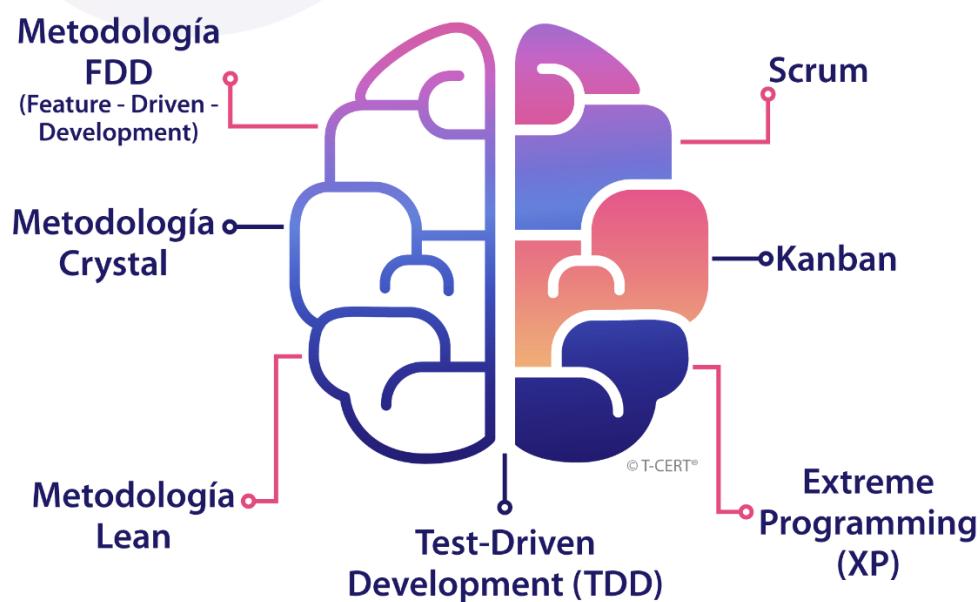
# Módulo VIII. Metodologías Agiles

## Módulo VIII. Metodologías Agiles

Las organizaciones están usando métodos ágiles porque realmente funcionan, y permiten:

- **Adaptación constante:** las metodologías ágiles permiten ajustar la estrategia para la transición hacia el agilismo y en la medida que se generan productos y servicios a través de los proyectos.
- **Entrega continua:** los resultados se pueden obtener desde etapas tempranas, aportando al valor para el cliente y su captura de beneficios, así como ir realizando mejoras o ajustes con base en la retroalimentación.
- **Mejora la colaboración:** los equipos cohesionados obtienen grandes resultados. En las metodologías ágiles se apoya la definición y compresión de roles, así como el trabajo co-equipo basado en la colaboración, es decir, buscando beneficios comunes.

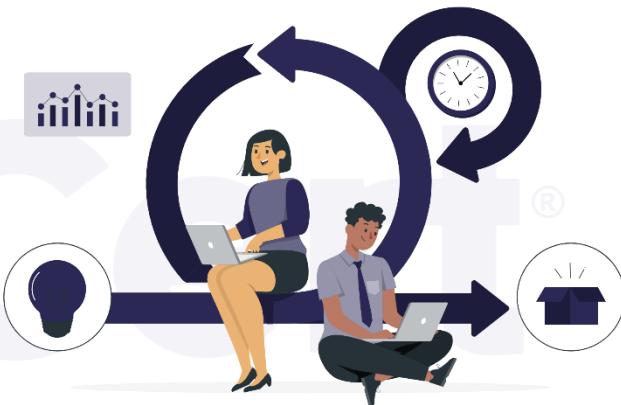
Dentro de las metodologías ágiles a destacar, se tienen las siguientes:



## 8.1 Scrum

Es la metodología ágil más popular. Scrum se basa en ciclos de trabajo iterativos y colaborativos llamados sprints, generalmente de 1 a 4 semanas, donde el equipo desarrolla un incremento de producto en cada ciclo. Scrum tiene roles específicos como el Scrum Master, facilitador del proceso, el Product Owner, responsable de gestionar el backlog del producto y el Development Team, equipo de desarrollo.

Se caracteriza por ser la metodología del caos que se basa en una estructura de desarrollo incremental, esto es, cualquier ciclo de desarrollo del producto y/o servicio se desglosa en pequeños proyectos divididos en distintas etapas: análisis, desarrollo y testing. En la etapa de desarrollo encontramos lo que se conoce como interacciones del proceso o Sprint, es decir, entregas regulares y parciales del producto final.



Esta metodología permite abordar proyectos complejos que exigen una flexibilidad y una rapidez esencial a la hora de ejecutar los resultados. La estrategia irá orientada a gestionar y normalizar los errores que se puedan producir en desarrollos demasiado largos, a través de, reuniones frecuentes para asegurar el cumplimiento de los objetivos establecidos.

Las reuniones son el pilar fundamental de la metodología, donde diferenciamos entre: reuniones de planificación, diaria, de revisión y de retrospectiva, la más importante de todas ellas, ya que, se realiza después de terminar un sprint para reflexionar y proponer mejoras en los avances del proyecto. Los aspectos clave por los que se mueve el Scrum son: innovación, flexibilidad, competitividad y productividad.

Scrum es una metodología ágil eficaz que favorece la colaboración, la transparencia, la entrega continua de valor y la mejora continua. Aunque se originó en el desarrollo de software, sus principios y prácticas también se pueden aplicar en otros campos, como la gestión de proyectos o la innovación de productos de muchas industrias. Al seguir sus principios y roles de manera rigurosa, los equipos pueden adaptarse rápidamente a los cambios y entregar productos de alta calidad de manera regular.

### 8.1.1 Roles en Scrum

Scrum define tres roles principales que son fundamentales para el éxito del equipo:



**Product Owner (Propietario del Producto):** Es el responsable de gestionar el Backlog del Producto (Product Backlog), que es una lista priorizada de todas las funcionalidades, características, mejoras, correcciones, etc., que deben ser desarrolladas en el producto. El Product Owner debe entender las necesidades del cliente y del negocio, tomando decisiones sobre qué debe priorizarse y cuándo debe desarrollarse cada característica. Actúa como el enlace entre el equipo Scrum y los stakeholders (interesados) del proyecto.

**Scrum Master:** Es el facilitador del proceso Scrum. El Scrum Master asegura que el equipo siga las prácticas y principios ágiles de Scrum, ayudando a remover obstáculos o impedimentos que dificulten el progreso. También es responsable de la educación del equipo y la



organización sobre cómo usar Scrum correctamente. Su objetivo es crear un ambiente donde el equipo pueda ser altamente productivo y eficiente.



**Desarrolladores:** Este grupo es el encargado de desarrollar el producto. Está compuesto por profesionales que trabajan en tareas técnicas para construir los entregables o incrementos de producto. El equipo de desarrollo es autónomo y multifuncional, lo que significa que tiene todas las habilidades necesarias para entregar el incremento del producto al final de cada sprint. El tamaño ideal del equipo es de 10 personas o menos.

### 8.1.2 Eventos en Scrum

Scrum tiene un conjunto de eventos o ceremonias que estructuran el trabajo en iteraciones cortas y regulares, llamadas sprints.





**Sprint:** Es el ciclo de trabajo en Scrum. Cada sprint tiene una duración fija de entre 1 y 4 semanas. Durante un sprint, el equipo trabaja para entregar un incremento funcional del producto, es decir, un conjunto de funcionalidades completas, probadas y listas para ser entregadas. Al final de cada sprint, el equipo tiene un incremento de producto listo para la revisión por parte de los stakeholders.



**Sprint Planning (Planificación del Sprint):** Es la reunión donde el equipo define qué se va a hacer en el sprint y cómo se va a hacer. En esta reunión se elige un conjunto de elementos del Product Backlog que se completarán durante el sprint. Luego, el equipo de desarrollo desglosa estos elementos en tareas más pequeñas. El Product Owner explica las prioridades del Product Backlog, y el equipo de desarrollo se compromete con lo que puede entregar durante el sprint.



**Daily Scrum (Reunión Diaria):** Es una reunión breve (15 minutos) que se realiza todos los días del sprint. En esta reunión, cada desarrollador indica el avance de las actividades en curso y las que va a abordar antes del siguiente daily. El objetivo es sincronizar al equipo, identificar obstáculos rápidamente, asegurar que todos estén alineados con la meta del sprint y hacer los ajustes pertinentes.

**Sprint Review (Revisión del Sprint):** Al final de cada sprint, el equipo realiza una revisión para mostrar el trabajo realizado (el incremento del producto) a los stakeholders. Durante esta reunión, se analiza si el producto cumple con los requisitos definidos al inicio del sprint y se obtiene retroalimentación de los interesados. A partir de esta retroalimentación, el Product Backlog puede ser actualizado, priorizando nuevas funcionalidades o cambios necesarios.



**Sprint Retrospective (Retrospectiva del Sprint):** Es una reunión donde el equipo analiza cómo ha trabajado durante el sprint e identifica áreas de mejora en el proceso. El equipo Scrum analiza lo que salió bien, para mantenerlo y fortalecerlo y lo que no salió tan bien para establecer acciones de mejora y efectividad. El objetivo de la retrospectiva es lograr la mejora continua, optimizando el proceso Scrum para hacer al equipo más eficiente en los próximos sprints.

### 8.1.3 Artefactos en Scrum

Scrum define tres artefactos clave que ayudan a garantizar que el trabajo esté organizado y alineado con las metas del proyecto:





**Product Backlog (Backlog del Producto):** Es una lista priorizada de todas las características, funcionalidades, mejoras y arreglos que se desean implementar en el producto. El Product Owner es responsable de gestionarlo y priorizarlo continuamente para reflejar las necesidades del negocio y las expectativas de los stakeholders. El Product Backlog es un documento vivo, que cambia a medida que el proyecto avanza.

**Sprint Backlog (Backlog del Sprint):** Es la lista de tareas que el equipo de desarrollo se compromete a completar durante un sprint específico. El Sprint Backlog está compuesto por los elementos del Product Backlog seleccionados para el sprint, desglosados en tareas específicas que el equipo llevará a cabo. A medida que el sprint avanza, el Sprint Backlog puede evolucionar si el equipo descubre que necesitan ajustar el enfoque.



**Incremento:** El Incremento es la suma de todos los elementos del Product Backlog completados durante un sprint y de los incrementos de sprints anteriores. Al final de cada sprint, el Incremento debe ser un producto funcional, probado y potencialmente entregable, es decir, debe cumplir con la Definición de Hecho (DoD). El Incremento proporciona valor tangible a los stakeholders y al cliente.

#### 8.1.4 Definición de Terminado o de Hecho

La Definición de Hecho (DoD) es un conjunto de criterios que deben cumplirse para que un elemento del Product Backlog o el Incremento se considere "completo" o terminado. La DoD asegura que el trabajo entregado tiene calidad, ha sido probado y cumple con las expectativas de los stakeholders.



Estos criterios pueden incluir, por ejemplo: código probado, documentación actualizada, funcionalidad verificada por QA, Revisión por parte del Product Owner y muchos otros asociados a la calidad del producto, incluyendo seguridad, continuidad, escalabilidad entre otros.

#### 8.1.5 Beneficios de Scrum



- Visibilidad y Transparencia:** Scrum promueve la visibilidad constante del progreso del equipo, lo que permite una toma de decisiones informada.
- Flexibilidad:** Gracias a los sprints cortos, Scrum permite adaptarse a cambios de requisitos y prioridades de manera continua.



**Mejora Continua:** A través de las retrospectivas, Scrum fomenta la mejora continua en los procesos y la manera de trabajar del equipo.



**Entrega Incremental de Valor:** El enfoque en la entrega frecuente de incrementos funcionales asegura que el cliente reciba valor en cada sprint.

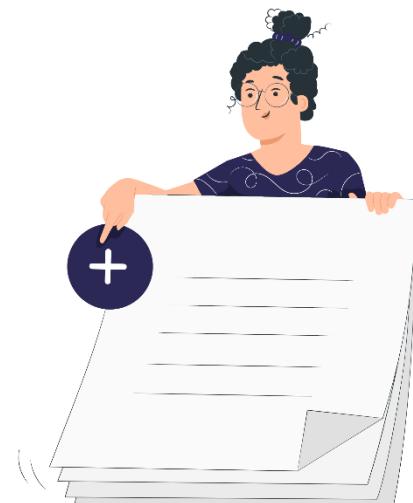


**Colaboración y Comunicación:** Las reuniones diarias y las interacciones continuas entre el equipo y el Product Owner aseguran una comunicación fluida y colaborativa.

## 8.2 Kanban

Kanban es una metodología ágil que se basa en la visualización, la limitación del trabajo en proceso y la mejora continua para optimizar el flujo de trabajo. Su enfoque flexible lo convierte en una excelente opción para equipos que buscan simplificar la gestión de tareas y mejorar la eficiencia sin adoptar la rigidez de un marco de trabajo más estructurado como Scrum.

Kanban proviene de una palabra japonesa que significa “tarjeta” o “señal visual”, y se utiliza para representar el trabajo que se está realizando en un sistema. Es una metodología no prescriptiva y ligera, por lo que no exige roles específicos, ni sprints. Se enfoca en la mejora continua mediante la optimización del flujo de trabajo y la



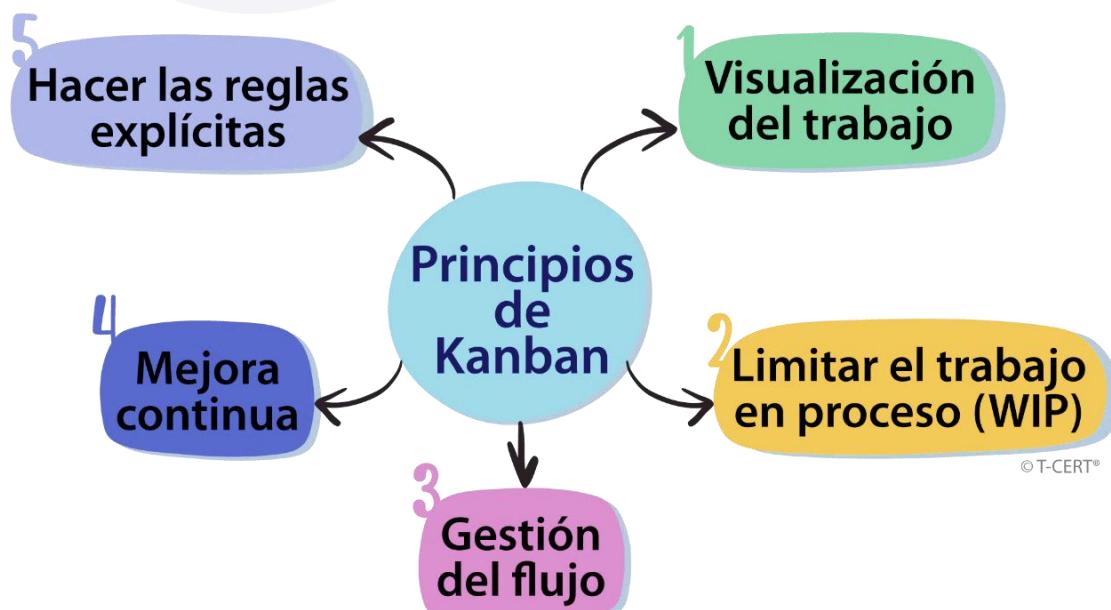
eliminación de cuellos de botella, además de la visualización del flujo de trabajo y la gestión del mismo.

Kanban utiliza tableros (físicos o digitales) donde las tareas son representadas como tarjetas moviéndose entre columnas, que representan los diferentes estados del proceso (por ejemplo: Sin iniciar, En Curso, Finalizado). Este tablero debe estar al alcance de todos los miembros del equipo, evitando así la repetición de tareas o la posibilidad de que se olvide alguna de ellas. Por tanto, ayuda a mejorar la productividad y eficiencia del equipo de trabajo.

Las ventajas que proporciona esta metodología son:



#### 8.2.1 Principios de Kanban

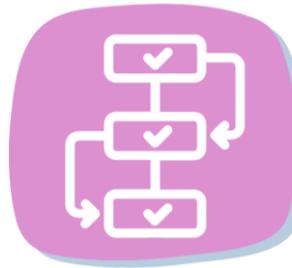


**1. Visualización del trabajo:** En Kanban, se utiliza un tablero Kanban (físico o digital) para visualizar el flujo de trabajo. Las tareas se representan como tarjetas (también llamadas " pósteres") que se mueven a través de las columnas que representan las distintas fases del proceso. Visualizar el trabajo permite identificar fácilmente los cuellos de botella, los bloqueos y las áreas que necesitan atención.



**2. Limitar el trabajo en proceso (WIP):** Una de las características más importantes de Kanban es la limitación del trabajo en proceso. Establecer límites en la cantidad de tareas que pueden estar en una columna en un momento dado ayuda a evitar el exceso de trabajo y la sobrecarga de los miembros del equipo. Esto permite una mayor concentración en las tareas actuales y asegura que el equipo se enfoque en finalizar antes de empezar nuevas tareas.

**3. Gestión del flujo:** Kanban se enfoca en optimizar el flujo de trabajo, es decir, asegurar que las tareas se muevan a través del sistema de manera fluida y continua. Esto implica la identificación de cuellos de botella, la mejora del proceso y la mejora de la eficiencia en todas las fases del trabajo.



**4. Mejora continua:** Kanban fomenta la mejora continua mediante la revisión regular de cómo están funcionando los procesos. El equipo debe reflexionar sobre los problemas que surgen y buscar maneras de mejorar la forma en que se gestionan las tareas. A través de las métricas y el análisis del flujo de trabajo, los equipos pueden tomar decisiones



informadas sobre cómo mejorar la eficiencia y la calidad del trabajo.

**5. Hacer las reglas explícitas:** Las reglas del proceso de trabajo (como los límites de WIP, la definición de lo que significa "terminado", entre otros) deben ser claras y transparentes para todos los miembros del equipo. De esta forma, todos comprenden cómo deben trabajar y qué se espera de ellos, lo que fomenta la colaboración y la alineación en el equipo.



### 8.2.2 El Tablero Kanban

El tablero Kanban es el corazón visual de la metodología. Se organiza en columnas que representan las diferentes etapas del flujo de trabajo. Las tareas o trabajos se colocan en tarjetas que se mueven de una columna a otra según su estado actual. Un tablero Kanban puede ser físico (en una pizarra o pared) o digital (utilizando herramientas como Jira, Trello, Kanbanize u otras.). Las columnas del tablero Kanban típicamente incluyen:

Sin iniciar	En Curso	En Revisión/ Pruebas	Finalizado
<p>Las tareas que aún no se han comenzado, pero están listas para ser trabajadas en el futuro.</p> 	<p>Las tareas en las que el equipo está trabajando activamente. Aquí es donde las tarjetas pasan de una columna a otra según avanza el trabajo.</p> 	<p>Si es necesario, una columna intermedia para las tareas que están siendo revisadas o probadas antes de ser consideradas completas.</p> 	<p>Las tareas que han sido completadas y no requieren más trabajo.</p> 

### 8.2.3 Límite de Trabajo en Proceso (WIP)

Una de las claves de Kanban es limitar el trabajo en proceso. Esto significa que el equipo define un número máximo de tareas que pueden estar en cada columna del tablero en cualquier momento. Por ejemplo: En la columna “En Curso”, solo pueden estar tres tareas a la vez. En la columna “En Revisión”, solo pueden estar dos tareas a la vez.

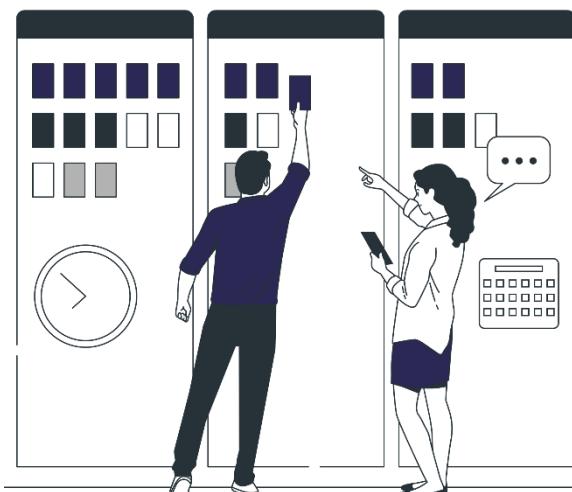
#### ¿Por qué limitar el WIP?

- Ayuda a evitar que el equipo se sobrecargue.
- Fuerza a terminar las tareas antes de comenzar nuevas.
- Reduce el tiempo de ciclo (es decir, el tiempo que tarda una tarea en pasar de Pendiente a Terminado).
- Permite detectar más fácilmente los cuellos de botella o las áreas del proceso que necesitan mejoras.

### 8.2.4 Flujo y Métricas en Kanban

Kanban se enfoca en optimizar el flujo de trabajo y garantizar que las tareas se muevan de manera eficiente a través del sistema. Para hacer esto, se utilizan algunas métricas clave:

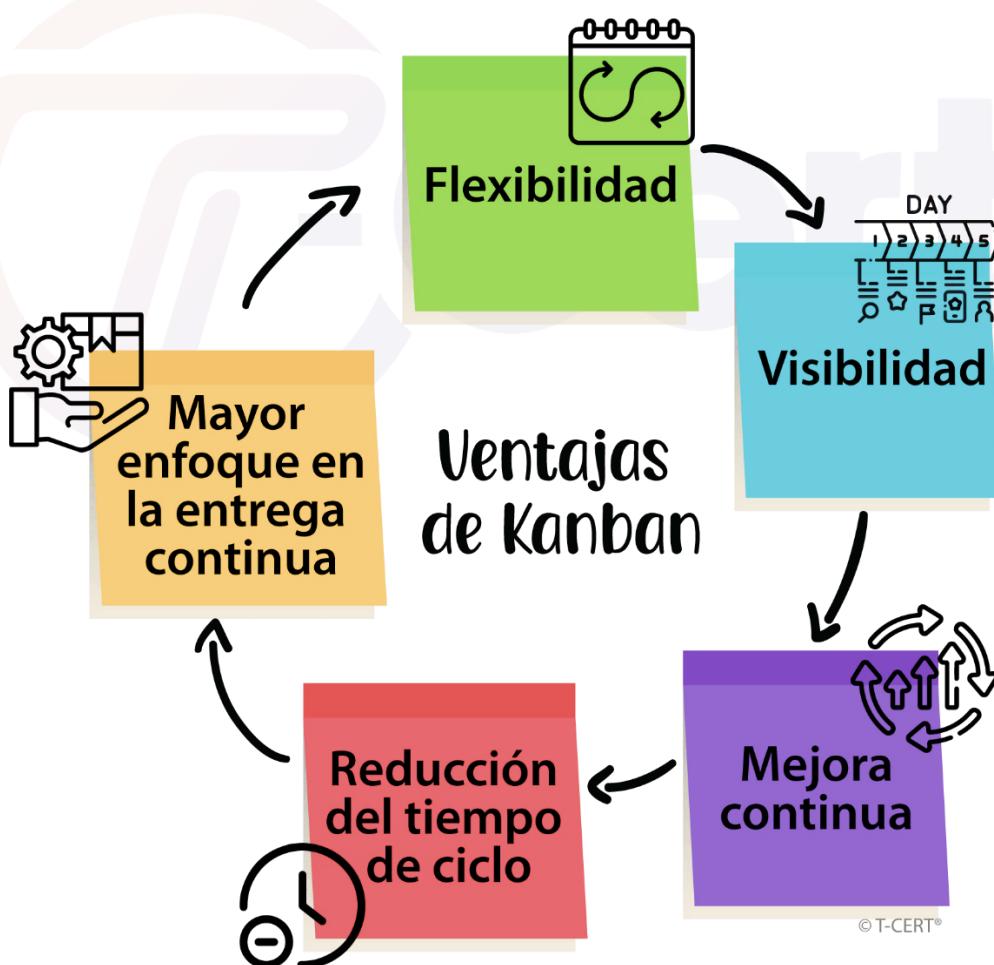
- **Lead time:** El tiempo total que tarda una tarea desde que se coloca en el Backlog hasta que se completa y pasa a Terminado. Es una métrica útil para medir la eficiencia del sistema en general.



- **Cycle time:** El tiempo que tarda una tarea desde que se inicia hasta que se completa. Es una métrica más específica para analizar el tiempo de ejecución de tareas individuales.
- **Throughput:** La cantidad de tareas completadas en un período de tiempo específico (por ejemplo, cuántas tareas el equipo termina cada semana).

Estas métricas se utilizan para mejorar el proceso a medida que se observa cómo fluye el trabajo y qué áreas necesitan ajustes.

#### 8.2.5 Ventajas de Kanban





## Flexibilidad

**Flexibilidad:** A diferencia de Scrum, Kanban no impone ciclos de trabajo fijos ni sprints, lo que lo hace muy flexible para equipos que manejan trabajo continuo o flujos de trabajo variables.

**Visibilidad:** La visualización del flujo de trabajo permite identificar rápidamente cuellos de botella, tareas bloqueadas o áreas que necesitan atención.



## Visibilidad



**Mejora continua:** Kanban fomenta la revisión constante y la mejora del proceso, permitiendo que el equipo evolucione y optimice su flujo de trabajo con el tiempo.

**Reducción del tiempo de ciclo:** Al limitar el WIP y concentrarse en completar tareas antes de comenzar nuevas, Kanban ayuda a reducir el tiempo que lleva completar una tarea.



## Reducción del tiempo de ciclo



**Mayor enfoque en la entrega continua:** Kanban permite entregar valor de manera más fluida, sin la necesidad de esperar a un sprint para lanzar una nueva versión o entrega.

### 8.2.6 ¿Cuándo usar Kanban?

Kanban es adecuado para equipos que:

- Tienen flujos de trabajo continuos y necesitan flexibilidad.

- No requieren el ciclo de iteración estructurado de Scrum (los sprints).
- Trabajan en tareas de mantenimiento o soporte donde el trabajo llega de manera impredecible.
- Buscan una optimización continua del flujo sin grandes interrupciones o cambios en el proceso.

### 8.3 Extreme Programming (XP)

Extreme Programming (XP) es una metodología ágil que pone el énfasis en la calidad del código, la colaboración constante, la retroalimentación rápida y la adaptación continua a los cambios a través de prácticas técnicas sólidas. Sus prácticas, como la programación en pareja, TDD, integración continua y refactorización, garantizan que el software sea de alta calidad, flexible y fácil de mantener.

Está diseñada para mejorar la calidad del software y la capacidad de respuesta ante los cambios en los requisitos del cliente. XP pone un fuerte énfasis en las prácticas técnicas y en la colaboración constante entre desarrolladores y clientes, lo que permite entregar software de alta calidad de manera rápida y eficiente.

Extreme Programming (XP) fue propuesto por Kent Beck en la década de 1990 y tiene como base una serie de prácticas de ingeniería de software que se enfocan en mejorar tanto la calidad del código como la productividad del equipo.



### 8.3.1 Principios de Extreme Programming



- **Comunicación continua:** La comunicación constante y clara entre los miembros del equipo y los stakeholders (clientes, usuarios) es esencial. Los desarrolladores deben interactuar frecuentemente para garantizar que todos estén alineados en los objetivos y los cambios.
- **Simplicidad:** En XP, se busca escribir el código más simple posible que funcione. La idea es evitar la sobreingeniería y las soluciones complejas que no son necesarias en el momento.
- **Retroalimentación rápida:** El ciclo de retroalimentación constante es vital. Se reciben retroalimentaciones tanto del código (a través de pruebas automáticas) como del cliente (para validar los requisitos y expectativas).
- **Coraje:** Los equipos deben tener el coraje de realizar cambios y tomar decisiones difíciles, como refactorizar el código o admitir errores. Este principio también implica que el equipo debe ser valiente a la hora de afrontar los desafíos técnicos.

- **Respeto mutuo:** Los miembros del equipo deben respetarse y confiar los unos en los otros. La colaboración efectiva depende de un ambiente de trabajo donde se valoren las contribuciones de todos.

### 8.3.2 Prácticas de Extreme Programming

XP está compuesto por una serie de prácticas técnicas que buscan mejorar la calidad del software, asegurar la satisfacción del cliente y fomentar la colaboración del equipo. Estas son algunas de las prácticas más destacadas:



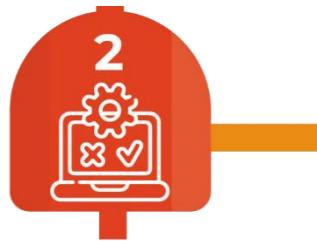


## Programación en pareja (Pair Programming)

**1. Programación en pareja (Pair Programming):** En lugar de que un solo desarrollador trabaje en una tarea, dos desarrolladores trabajan juntos en una sola estación de trabajo. Uno de los desarrolladores escribe el código (el "driver") mientras que el otro revisa y ofrece sugerencias (el "navigator"). Esta práctica fomenta la calidad del código, la revisión continua y el compartir conocimiento dentro del equipo.

## 2. Desarrollo basado en pruebas (Test-Driven Development, TDD):

En XP, el código siempre se desarrolla escribiendo primero las pruebas unitarias antes de escribir el código funcional. TDD asegura que el código se cubra con pruebas desde el principio, lo que mejora la calidad y facilita la detección de errores temprano. El ciclo de TDD consta de tres pasos:



## Desarrollo basado en pruebas (TDD)

- Escribir una prueba que falle.
- Escribir solo el código necesario para que la prueba pase.
- Refactorizar el código para mejorar la calidad y mantenerlo limpio.



## Integración continua (CI)

**3. Integración continua (Continuous Integration, CI):** En XP, el código se integra en un repositorio común varias veces al día. Esto permite que los desarrolladores obtengan retroalimentación inmediata sobre la calidad del código y cualquier posible conflicto de integración. Las pruebas automatizadas se ejecutan en cada integración para detectar errores de manera temprana y garantizar que los cambios no rompan el software existente.

**4. Refactorización:** Refactorizar significa modificar el código para mejorar su estructura sin cambiar su comportamiento. La refactorización es clave en XP, ya que mantiene el código limpio, fácil de entender y de mantener. Se realiza de forma continua para evitar que el código se "ensucie" con el tiempo.



## Refactorización



## Trabajo en pequeñas iteraciones

**5. Trabajo en pequeñas iteraciones:** XP recomienda trabajar en pequeñas iteraciones de una o dos semanas. Cada iteración debe ser completamente funcional, entregando un producto que pueda ser probado y evaluado por el cliente. Las iteraciones pequeñas permiten una retroalimentación rápida, lo que facilita la adaptación a los cambios y mejora la capacidad de respuesta ante los requisitos cambiantes.

**6. Propiedad colectiva del código (Collective Code Ownership):** En XP, todos los desarrolladores son responsables de todo el código. No hay un único propietario del código, lo que significa que cualquier miembro del equipo puede y debe mejorar o corregir cualquier parte del código cuando sea necesario. Esto fomenta la colaboración entre los desarrolladores y asegura que todos comprendan bien el código y el diseño del sistema.



© T-CERT®

## Propiedad colectiva del código (Collective Code Ownership)

### 8.3.3 Prácticas de Gestión

- Planificación de la iteración (Iteration Planning):** Antes de comenzar cada iteración, el equipo se reúne con el cliente para definir qué funcionalidades se desarrollarán durante la iteración. El equipo establece una meta clara para la iteración, y las tareas se desglosan en historias de usuario con estimaciones de tiempo.

- **Historias de usuario:** Las historias de usuario son una descripción simple de una funcionalidad que el sistema debe tener, desde la perspectiva del usuario final. Estas historias de usuario ayudan a definir el trabajo necesario para alcanzar las metas de cada iteración y son fundamentales para la planificación y el seguimiento del progreso.
- **Retroalimentación constante del cliente:** El cliente está involucrado en cada fase del proceso de desarrollo y proporciona retroalimentación constante. Al final de cada iteración, el cliente revisa el software y proporciona sugerencias para ajustes o nuevas características.

#### 8.3.4 Beneficios de Extreme Programming

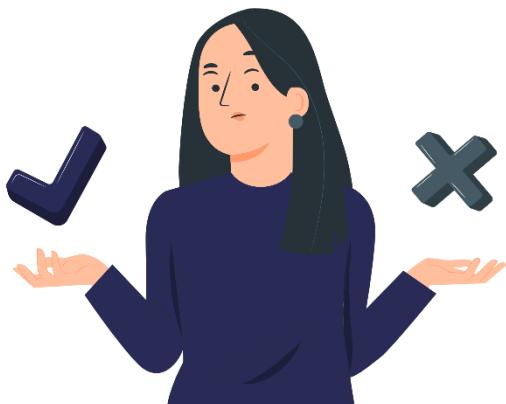


1. **Alta calidad de software:** Gracias a la combinación de prácticas como TDD, refactorización, programación en pareja e integración continua, XP asegura que el software sea más robusto, probado y fácil de mantener.

2. **Flexibilidad ante cambios:** XP está diseñado para adaptarse a cambios frecuentes en los requisitos del cliente. Las iteraciones pequeñas, la retroalimentación constante y el trabajo cercano con el cliente permiten que el equipo se ajuste rápidamente a nuevas prioridades.
3. **Mejora de la colaboración:** La programación en pareja, la propiedad colectiva del código y la planificación en equipo fomentan la colaboración y la comunicación dentro del equipo de desarrollo.
4. **Entrega temprana y frecuente:** Al trabajar en pequeñas iteraciones y entregar versiones funcionales del producto con regularidad, XP permite que el cliente vea y pruebe el software frecuentemente, reduciendo el riesgo de desviarse de lo que el cliente realmente necesita.

#### 8.3.5 Desventajas o Desafíos de XP

- **Requiere un alto nivel de compromiso:** Las prácticas de XP, como la programación en pareja y la integración continua, requieren un alto nivel de compromiso y disciplina por parte del equipo.
- **Puede ser difícil de implementar en equipos grandes:** XP fue diseñado principalmente para equipos pequeños, por lo que su implementación en equipos grandes o distribuidos puede ser más complicada.
- **Requiere una estrecha colaboración con el cliente:** XP asume que el cliente está disponible y dispuesto a colaborar constantemente, lo que puede no ser posible en todos los casos.

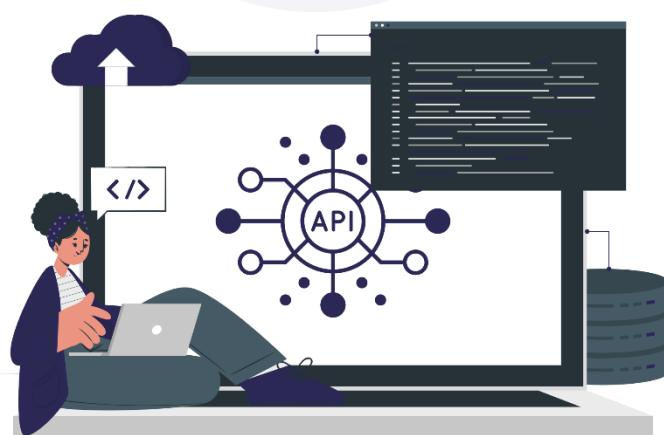


#### 8.4 Test-Driven Development (TDD)

Test-Driven Development (TDD) es una práctica de desarrollo de software donde las pruebas son escritas antes de escribir el código de producción. La idea central de TDD es que el código debe ser probado desde el principio para asegurar que cumpla con los requisitos y se comporte como se espera.

Es una práctica fundamental dentro de las metodologías ágiles. Se centra en la creación de pruebas antes de escribir el código que hace funcionar la funcionalidad deseada. Es una técnica clave para mejorar la calidad del código y promover el desarrollo iterativo y evolutivo. Este enfoque ayuda a detectar errores de forma temprana, mejora la mantenibilidad del código y asegura que el software cumpla con los requisitos establecidos.

El ciclo de TDD implica escribir primero una prueba que falle, luego escribir el código necesario para hacer que la prueba pase, y finalmente refactorizar el código para mantenerlo limpio y bien estructurado. Este ciclo de desarrollo rápido mejora la calidad del software y ayuda a crear un código que sea fácil de modificar y extender en el futuro.



Si bien TDD tiene una curva de aprendizaje y puede llevar más tiempo al principio, sus beneficios a largo plazo, como un código más limpio, menos errores y una mayor confianza en el sistema, lo hacen una práctica valiosa para equipos ágiles, especialmente cuando se combinan con otras prácticas como programación en pareja o integración continua.

#### 8.4.1 Ciclo de TDD: Red-Green-Refactor

TDD sigue un ciclo muy claro de tres pasos que se repite durante todo el proceso de desarrollo. Este ciclo se conoce como el ciclo Red-Green-Refactor:

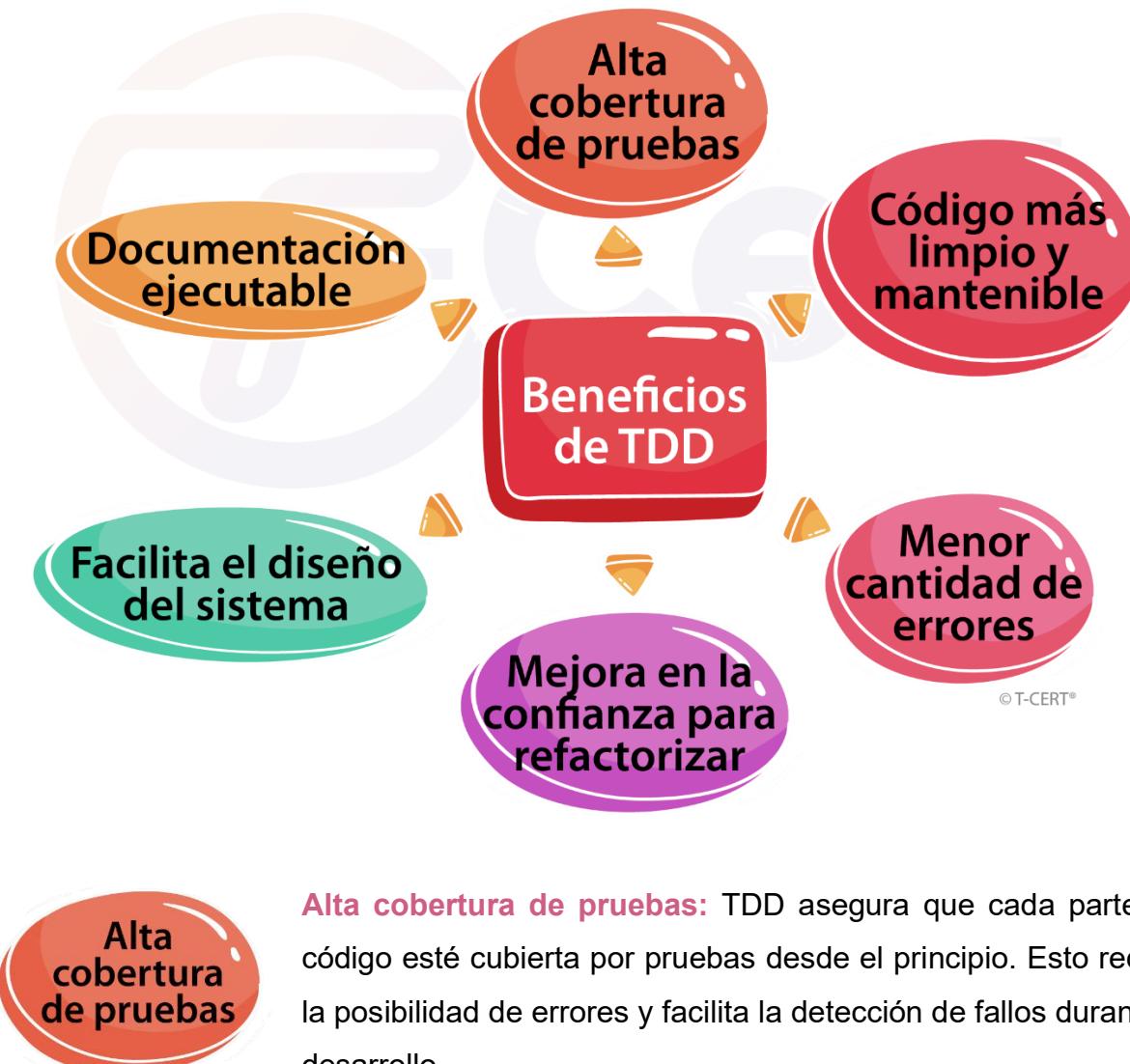


- **Rojo (Red):** Escribe una prueba que falle. El primer paso en TDD es escribir una prueba automatizada que falle. Dado que el código que implementa la funcionalidad aún no está escrito, la prueba debería fallar inicialmente. El objetivo es definir claramente lo que debe hacer el código que vas a escribir. Esta prueba puede ser una prueba unitaria, de integración o cualquier otro tipo de prueba, dependiendo de lo que estés intentando verificar.
- **Verde (Green):** Escribe el mínimo código necesario para hacer que la prueba pase. Una vez que tienes una prueba que falla, el siguiente paso es escribir solo el código mínimo necesario para hacer que esa prueba pase. No te preocupes por la calidad del código en este punto, solo asegúrate de que la prueba pase. El objetivo es centrarse en lo más simple y directo posible, de modo que el código implementado se ajuste a la prueba que escribiste.
- **Refactorizar (Refactor):** Mejora el código sin cambiar su funcionalidad. Una vez que la prueba ha pasado, el siguiente paso es refactorizar el código. Esto significa

mejorar el diseño y la estructura del código, eliminando cualquier redundancia y haciendo el código más limpio y legible. La refactorización se debe hacer sin modificar el comportamiento del código. Después de refactorizar, las pruebas deben seguir pasando. La refactorización es clave porque ayuda a mantener el código limpio y manejable a medida que se agregan nuevas características.

#### 8.4.2 Beneficios de TDD

TDD ofrece una serie de beneficios que lo convierten en una práctica muy popular en el desarrollo ágil de software:



**Alta cobertura de pruebas:** TDD asegura que cada parte del código esté cubierta por pruebas desde el principio. Esto reduce la posibilidad de errores y facilita la detección de fallos durante el desarrollo.

**Código más limpio y mantenable:** Al refactorizar continuamente el código después de cada iteración (después de que cada prueba pase), TDD fomenta la escritura de código modular y bien estructurado. Las pruebas ayudan a los desarrolladores a mantener un código limpio y a eliminar código innecesario o redundante.

Código más limpio y mantenable

Menor cantidad de errores

**Menor cantidad de errores:** Dado que las pruebas se escriben primero y se ejecutan continuamente, es más probable que los errores sean detectados temprano en el ciclo de desarrollo, lo que reduce la probabilidad de problemas en las fases finales del desarrollo.

**Mejora en la confianza para refactorizar:** Con TDD, los desarrolladores tienen una red de seguridad en forma de pruebas automatizadas que aseguran que el código no se rompa al realizar refactorizaciones o cambios.

Mejora en la confianza para refactorizar

Facilita el diseño del sistema

**Facilita el diseño del sistema:** TDD obliga a pensar en los requisitos antes de implementar una funcionalidad, lo que a menudo lleva a un diseño más limpio y enfocado desde el principio. Ayuda a evitar la sobreingeniería, ya que solo se escribe el código necesario para que las pruebas pasen.

**Documentación ejecutable:** Las pruebas que se escriben en TDD actúan como una forma de documentación ejecutable del sistema. Puedes ver cómo se espera que se comporte el sistema a través de las pruebas, lo que ayuda tanto a los desarrolladores como a los nuevos miembros del equipo a comprender rápidamente el comportamiento del código.

Documentación ejecutable

### 8.4.3 Desafíos de TDD

Aunque TDD tiene muchos beneficios, también presenta ciertos desafíos y no es adecuado para todos los proyectos:



Requiere tiempo inicial



Dificultad para escribir pruebas en algunos contextos



Possible sobrecarga de pruebas



Requiere disciplina

© T-CERT®



Requiere tiempo inicial

**Requiere tiempo inicial:** TDD puede parecer más lento al principio porque requiere escribir pruebas antes de implementar el código real. Sin embargo, a largo plazo, este enfoque ahorra tiempo al reducir los errores y la necesidad de corrección de bugs.

**Dificultad para escribir pruebas en algunos contextos:** Hay tipos de código o funcionalidades (como interfaces gráficas de usuario o interacciones complejas con sistemas externos) que pueden ser más difíciles de probar usando TDD. Las pruebas de integración y las pruebas de sistema suelen ser más complejas que las pruebas unitarias y pueden requerir enfoques adicionales.



Dificultad para escribir pruebas en algunos contextos



Possible sobrecarga de pruebas

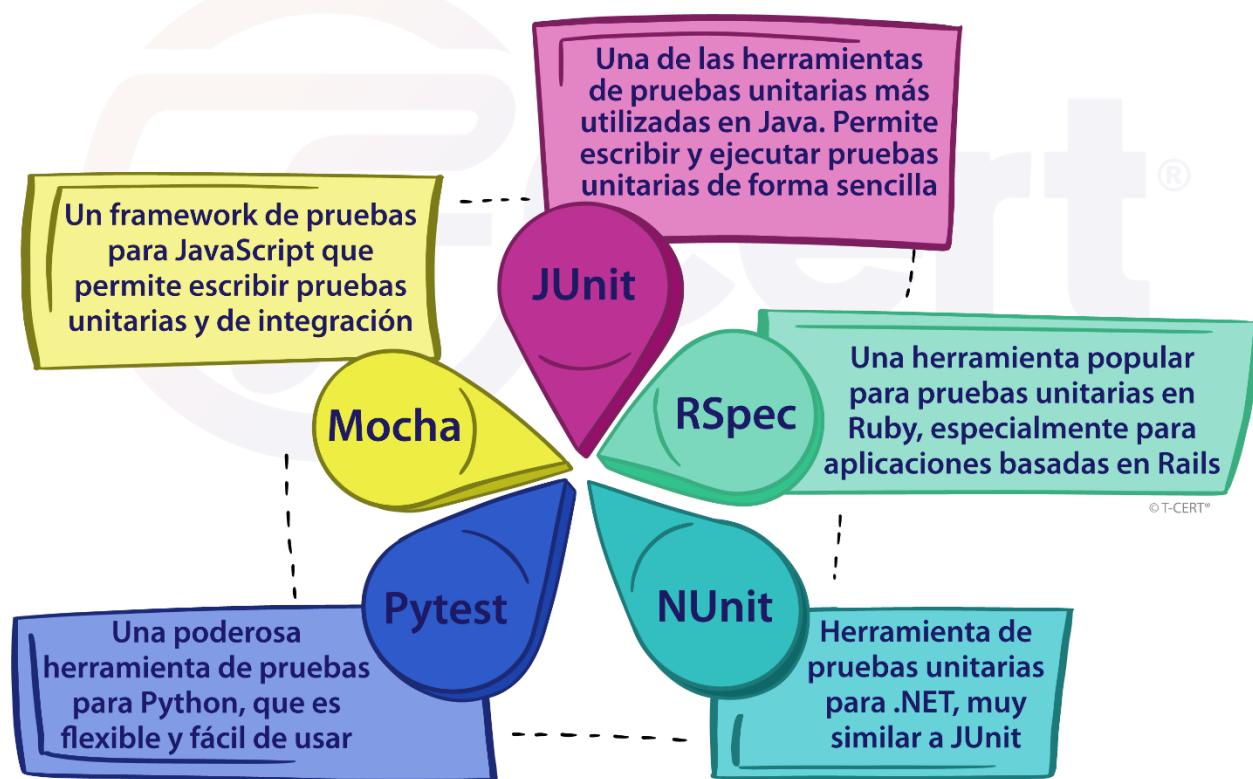
**Possible sobrecarga de pruebas:** Si no se gestionan correctamente, las pruebas pueden volverse demasiado numerosas o demasiado detalladas, lo que puede generar una sobrecarga en el equipo de desarrollo. Se necesita una estrategia clara de pruebas para evitar crear pruebas innecesarias o demasiado complejas.

**Requiere disciplina:** Para aprovechar al máximo TDD, los desarrolladores deben ser disciplinados en seguir el ciclo de Red-Green-Refactor y no saltarse ninguna de las fases. En proyectos con plazos ajustados o equipos sin experiencia en TDD, esto puede ser difícil de mantener.



#### 8.4.4 Herramientas de TDD

Existen diversas herramientas que pueden ayudar a implementar TDD de manera eficiente:



Estas herramientas proporcionan funcionalidades como la ejecución automática de pruebas, la aserción de condiciones (comprobación de que el código funciona como se espera) y la gestión de las pruebas de manera sencilla.

TDD es una práctica técnica que complementa y refuerza otras metodologías ágiles como Scrum, Kanban y Extreme Programming (XP). Estas son algunas de las maneras en que TDD se integra con otras prácticas ágiles:

- **En Scrum:** TDD se puede usar para asegurar que el código desarrollado en cada sprint esté completamente probado y sea de alta calidad.
- **En XP:** TDD es una de las prácticas fundamentales de Extreme Programming. De hecho, XP recomienda que el desarrollo de software se realice con pruebas automáticas escritas antes de escribir el código de producción.
- **En Kanban:** TDD puede ser útil para asegurar que el flujo de trabajo en un sistema Kanban esté bien controlado y que los desarrolladores no introduzcan errores mientras mueven tareas entre las diferentes fases.

## 8.5 Metodología Lean

Es una filosofía de gestión que se originó en la industria manufacturera, especialmente en el sistema de producción de Toyota (TPS Toyota Production System), y ha sido adaptada para el desarrollo de software y otros sectores. Lean se enfoca en la eliminación de desperdicios, la optimización de procesos y la entrega continua de valor al cliente.

La metodología Lean es un enfoque que busca maximizar el valor al cliente mientras se minimizan los recursos, el tiempo y el esfuerzo. Lean se centra en mejorar la eficiencia de los procesos, reduciendo el desperdicio (cualquier cosa que no agregue valor al cliente) y mejorando la calidad de los productos y servicios.

La metodología Lean es un enfoque poderoso para mejorar la eficiencia y la calidad en el desarrollo de software, basado



en los principios de eliminar desperdicios y maximizar el valor al cliente. Adoptar prácticas Lean ayuda a los equipos a trabajar de manera más eficiente, reducir el tiempo de entrega, y crear productos de mayor calidad, alineados con las necesidades del cliente.

En este contexto del desarrollo de software, Lean tiene principios y prácticas similares a los enfoques ágiles, pero con un énfasis adicional en eliminar todo lo que no es esencial para la creación de valor.

#### 8.5.1 Principios de Lean



1

## Eliminar desperdicios (Muda)

**Eliminar desperdicios (Muda):** El concepto clave en Lean es la eliminación de desperdicios, que se definen como cualquier actividad que no agregue valor al cliente. Esto incluye actividades como tareas innecesarias, exceso de trabajo en progreso, defectos, tiempo de espera, y más. En el contexto del desarrollo de software, esto puede implicar evitar la sobreproducción (desarrollar funcionalidades que no se necesitan), el exceso de documentación, el trabajo no alineado con las necesidades del cliente, o la espera entre tareas.

**Ampliar el conocimiento (Build Quality In):** Lean hace énfasis en construir calidad desde el principio, en lugar de verificarla al final del proceso. Esto se refiere a prácticas como la automatización de pruebas, la revisión de código, y la creación de software modular y fácil de mantener. La idea es que cada miembro del equipo es responsable de la calidad y se esfuerza por encontrar soluciones que mejoren el producto y el proceso a lo largo del desarrollo.

2

## Ampliar el conocimiento (Build Quality In)

3

## Reducir los tiempos de ciclo

**Reducir los tiempos de ciclo:** Lean se enfoca en reducir el tiempo necesario para completar una tarea. Esto se logra mediante la mejora del flujo de trabajo, la optimización de los procesos y la eliminación de cuellos de botella. En el desarrollo de software, esto se traduce en la capacidad de entregar valor al cliente de manera rápida y frecuente, lo que permite una retroalimentación más rápida y una adaptación continua a los cambios.

**Construir la calidad en el proceso:** En lugar de agregar calidad al final del proceso mediante pruebas exhaustivas, Lean promueve la integración de la calidad a lo largo de todo el ciclo de desarrollo, desde la concepción del producto hasta la

4

## Construir la calidad en el proceso

implementación. Esto implica la prevención de defectos y la refactorización continua del código, entre otras prácticas.

5

## Optimizar el todo, no las partes

**Optimizar el todo, no las partes:** Lean promueve la optimización de los procesos en su conjunto, en lugar de optimizar individualmente cada parte. Mejorar una sola fase de un proceso sin considerar todo el flujo de trabajo puede crear cuellos de botella en otras partes del sistema. En el contexto de desarrollo ágil, esto implica que todos los equipos (desarrollo, operaciones, pruebas, etc.) trabajen de manera conjunta y fluida para optimizar el rendimiento de todo el sistema.

**Empoderar a los colaboradores:** Lean también promueve el empoderamiento de los equipos. Los equipos deben ser autónomos y tener la responsabilidad de tomar decisiones que mejoren los procesos y la calidad del producto. En lugar de tener una estructura jerárquica rígida, Lean fomenta la colaboración y la toma de decisiones en el nivel más bajo posible, lo que ayuda a los equipos a responder más rápidamente a los problemas y cambios.

### 8.5.2 Tipos de Desperdicios de Lean





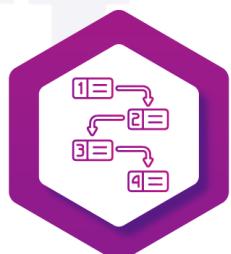
**Sobreproducción:** Crear más de lo necesario o producir funcionalidades que no son esenciales para el cliente en ese momento. Esto puede incluir desarrollar características adicionales que no aportan valor inmediato.

**Esperas:** El tiempo que los equipos o miembros del equipo pasan esperando por algo (por ejemplo, por una revisión, por una aprobación, o por acceso a recursos). Las esperas prolongadas afectan el flujo y generan ineficiencias.



**Transporte:** El movimiento innecesario de información, personas o materiales. En el contexto de software, esto puede incluir el movimiento de código entre diferentes sistemas, herramientas o equipos, o la necesidad de pasar código de un desarrollador a otro.

**Exceso de procesamiento:** Incluir pasos o procesos innecesarios que no agregan valor. Por ejemplo, escribir documentación extensa que no se utiliza o implementar procesos complicados cuando una solución más simple sería suficiente.



**Inventarios:** Mantener más trabajo en proceso (WIP) de lo necesario. En desarrollo de software, esto puede incluir tareas que están en progreso, pero no se completan rápidamente, o funcionalidades que no están siendo utilizadas o que están esperando para ser implementadas.

**Movimientos innecesarios:** Cualquier movimiento físico innecesario que un trabajador realice para cumplir con su tarea, como buscar información o cambiar de contexto. En el software, esto puede incluir



procesos manuales repetitivos o el tener que realizar cambios en diferentes sistemas de forma innecesaria.



**Defectos:** Los errores en el software que requieren correcciones o retrabajo. Los defectos son uno de los mayores desperdicios, ya que generan retrabajo, retrasos y desconfianza en el producto.

### 8.5.3 Prácticas Lean en el Desarrollo de Software

Lean en el desarrollo de software ha sido adaptado para incluir prácticas específicas que ayudan a mejorar la eficiencia y la calidad. Algunas de las prácticas Lean más comunes en este ámbito son:

- **Lean Software Development (LSD):** El desarrollo ágil de software basado en Lean es una versión adaptada de los principios Lean. Aquí se aplican los principios de eliminar desperdicios, construir calidad en el proceso y reducir los tiempos de ciclo en el contexto de la creación de software. Las principales prácticas de LSD incluyen:
  - Integración continua: Asegurar que el código se integre y pruebe continuamente, para detectar errores y mejorar la calidad.
  - Entregas frecuentes: Liberar versiones pequeñas y frecuentes de software que brinden valor al cliente de manera continua.
  - Colaboración estrecha con el cliente: Mantener una comunicación constante con los



clientes para asegurar que el producto está alineado con sus necesidades.

- **Kanban:** Kanban es una práctica derivada de Lean que se utiliza para gestionar el flujo de trabajo. A través de tableros visuales y la limitación del trabajo en proceso (WIP), Kanban ayuda a optimizar el flujo y reducir los cuellos de botella.
- **Control de flujo:** Es importante optimizar el flujo de trabajo y minimizar el tiempo de espera. Esto se logra mediante la implementación de sistemas de integración continua, despliegues automáticos y una rápida entrega de valor.
- **Refactorización:** Lean enfatiza la importancia de la refactorización continua del código para asegurar que se mantenga limpio, modular y fácil de mantener. La refactorización es parte integral de la construcción de calidad desde el principio.
- **Automatización de pruebas:** La automatización de las pruebas permite una retroalimentación rápida sobre la calidad del software sin retrasar el proceso. Se realizan pruebas continuas para verificar que no se introduzcan defectos y que el software esté funcionando correctamente.

#### 8.5.4 Beneficios de Lean en el Desarrollo de Software





Mejora de la eficiencia

**Mejora de la eficiencia:** Al reducir el desperdicio y optimizar el flujo de trabajo, Lean ayuda a los equipos a trabajar de manera más eficiente y con menos recursos.



Mayor flexibilidad y adaptabilidad

**Mayor flexibilidad y adaptabilidad:** Lean permite a los equipos adaptarse rápidamente a los cambios y a los nuevos requisitos del cliente, asegurando que se entregue valor de forma continua.



Mayor calidad

**Mayor calidad:** Al enfocarse en la prevención de defectos y la integración continua de calidad, los productos entregados son más confiables y estables.



Mejor colaboración

**Mejor colaboración:** Lean fomenta una mayor colaboración entre equipos multidisciplinarios y entre los desarrolladores y los clientes, asegurando que las soluciones estén alineadas con las necesidades del cliente.

#### 8.5.5 Desafíos de Implementar Lean

- **Requiere una cultura organizacional sólida:** La implementación de Lean puede ser difícil en organizaciones con una cultura tradicional jerárquica o de control estricto, ya que Lean promueve la autonomía del equipo y la toma de decisiones descentralizada.
- **Resistencia al cambio:** Algunos equipos pueden resistirse a los cambios que Lean requiere, especialmente si están acostumbrados a procesos más tradicionales o a trabajar de manera aislada.

- **Necesita una gestión efectiva del flujo:** Para que Lean funcione bien, es necesario tener un buen control sobre el flujo de trabajo, lo que puede ser complicado en equipos grandes o con procesos complejos.

## 8.6 Metodología Crystal

Crystal es una familia de metodologías ágiles basada en el concepto de que cada proyecto es único, por lo que el proceso debe ser adaptado a las características específicas de ese proyecto. A diferencia de otras metodologías ágiles, como Scrum o Kanban, Crystal no sigue un conjunto rígido de prácticas, sino que propone diferentes enfoques dependiendo del tamaño del equipo y la criticidad del proyecto.

Es una de las metodologías ágiles más flexibles y adaptables. Fue creada por Alistair Cockburn (uno de los firmantes del Manifesto Ágil) y se enfoca en la importancia de la gente y la comunicación dentro de los equipos, así como en la adaptación del proceso a las necesidades específicas del proyecto.



Crystal enfatiza la importancia de las personas en el equipo y la necesidad de una comunicación continua, tanto dentro del equipo como con los stakeholders del proyecto. Los principios clave de Crystal están orientados a la colaboración, la entrega continua de valor y la flexibilidad.

### 8.6.1 Principios de la Metodología Crystal



La gente es lo más importante

- **La gente es lo más importante:** En Crystal, se considera que las personas son el factor clave para el éxito de un proyecto. La metodología hace hincapié en que los equipos pequeños y altamente colaborativos son más efectivos que los grandes equipos. Se enfatiza la necesidad de contar con personas comprometidas, comunicativas y capaces de adaptarse a los cambios.
- **Entrega frecuente de software funcional:** Como en otras metodologías ágiles, Crystal propone la entrega continua de software funcionando. El ciclo de entrega es corto y frecuente, lo que permite recibir retroalimentación temprana y ajustar el rumbo del proyecto según las necesidades cambiantes.

Entrega frecuente de software funcional

## Reflexión constante sobre el proceso

- **Reflexión constante sobre el proceso:** Crystal fomenta la práctica de retroalimentación continua no solo a nivel de producto, sino también en el proceso de desarrollo. Los equipos deben revisar y ajustar su forma de trabajar regularmente para mejorar continuamente. Esto puede implicar sesiones regulares de retrospectiva donde el equipo identifica áreas de mejora en su dinámica y en la forma en que gestionan el trabajo.

- **Simplicidad:** Crystal promueve la simplicidad tanto en el diseño del producto como en los procesos utilizados. Se busca eliminar todo lo que no agrega valor al cliente o al equipo. La simplicidad también se extiende a la documentación: Crystal recomienda crear solo la documentación mínima necesaria, priorizando la comunicación directa.

## Simplicidad

## Transparencia y comunicación abierta

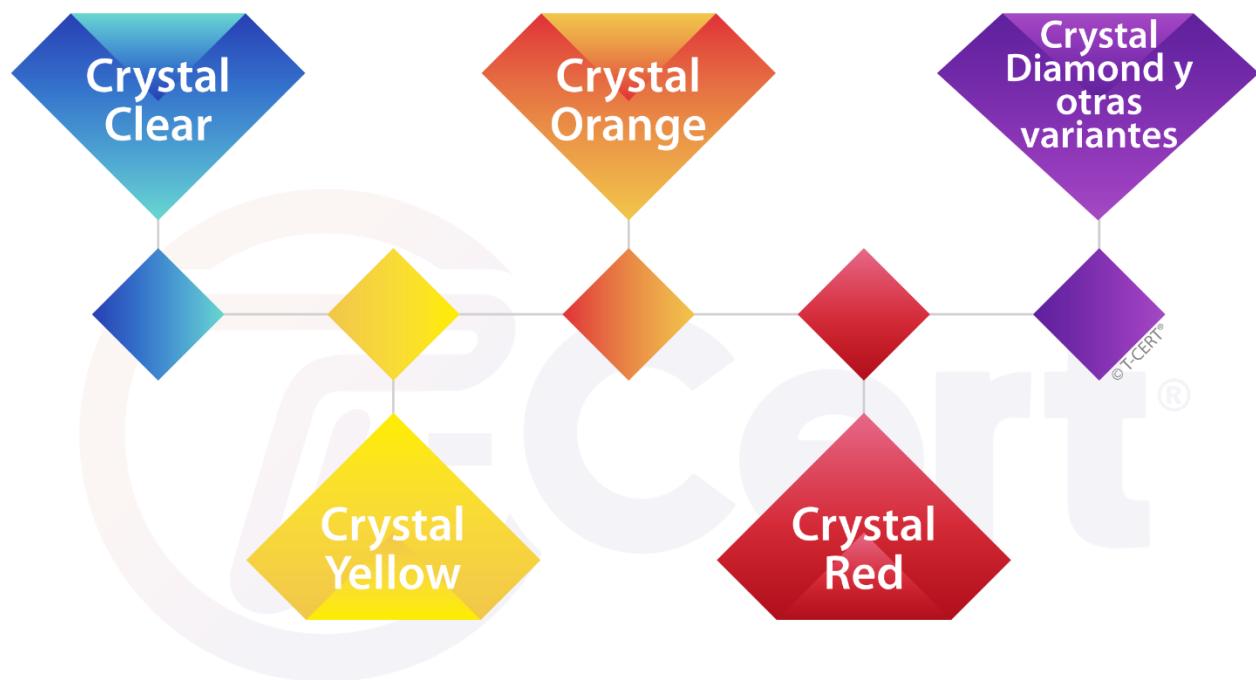
- **Transparencia y comunicación abierta:** La comunicación continua y abierta entre los miembros del equipo y con los stakeholders es esencial en Crystal. Se fomenta la interacción cara a cara y la toma de decisiones colaborativas para asegurar que todos estén alineados con los objetivos del proyecto. Se utilizan herramientas visuales (tableros, gráficos, etc.) para facilitar la comunicación.

- **Adaptación del proceso al contexto:** En Crystal, no existe un proceso único para todos los proyectos. La metodología se adapta a las características del proyecto, como el tamaño del equipo, la complejidad del producto y la criticidad del sistema que se está desarrollando. Esto significa que un proyecto grande o de alta criticidad puede necesitar un proceso más formal y estructurado, mientras que un proyecto pequeño o menos crítico puede seguir un proceso más ágil y sencillo.

## Adaptación del proceso al contexto

## 8.6.2 Las Variantes de Crystal

Una característica interesante de Crystal es que no es una única metodología, sino que es una familia de métodos que varían según el tamaño del equipo y la criticidad del proyecto. A medida que el equipo crece o el proyecto se vuelve más complejo, se pueden aplicar versiones más pesadas de Crystal.



- **Crystal Clear:** Para proyectos pequeños con equipos de 1 a 10 personas. Es una variante de Crystal muy ligera y centrada en la comunicación constante y la entrega rápida de software funcional. Se enfoca en la simplicidad y en la colaboración directa entre los miembros del equipo.

- **Crystal Yellow:** Para proyectos de hasta 20 personas. Es un poco más estructurada que Crystal Clear, pero sigue siendo ligera y ágil. Se introduce la gestión de los requisitos y algunos procesos de control, pero mantiene un enfoque flexible.



Crystal  
Orange

- **Crystal Orange:** Para proyectos de hasta 50 personas. Crystal Orange tiene un poco más de formalidad y un enfoque más claro en la gestión de riesgos y en la gestión de calidad, aunque sigue siendo ágil y enfocado en la entrega continua de valor.

- **Crystal Red:** Para proyectos grandes y críticos, con equipos de más de 100 personas. En este caso, la metodología Crystal se vuelve más formal y estructurada. Incluye gestión de calidad, gestión de requisitos, y un enfoque más detallado para asegurar que los productos cumplen con estándares muy altos.



Crystal  
Red

Crystal  
Diamond y  
otras  
variantes

- **Crystal Diamond y otras variantes:** Existen otras variantes para proyectos aún más complejos o críticos. Estas versiones incluyen prácticas de gestión de la calidad más avanzadas y procesos de control más detallados.

### 8.6.3 Ventajas de Crystal



- **Flexibilidad:** Crystal se adapta fácilmente a las características de cada proyecto, lo que lo convierte en un enfoque muy flexible. Puede ser utilizado tanto en proyectos pequeños y simples como en grandes proyectos críticos.
- **Énfasis en la gente:** Crystal pone a las personas en el centro del proceso. Se enfoca en mejorar la comunicación, el trabajo en equipo y el compromiso de todos los involucrados en el proyecto.
- **Reducción de desperdicios:** Como todas las metodologías ágiles, Crystal promueve la eliminación de desperdicios. Se enfoca en simplificar los procesos y en la entrega continua de valor.
- **Entrega temprana de valor:** La metodología fomenta entregas frecuentes de software funcional, lo que permite una



retroalimentación continua y un ajuste rápido a las necesidades cambiantes del cliente.

- **Adaptabilidad:** La metodología se ajusta tanto al tamaño del equipo como a la complejidad del proyecto, lo que la hace adecuada para una amplia gama de escenarios.

#### 8.6.4 Desventajas y Desafíos de Crystal



No es adecuada para proyectos grandes y complejos sin adaptación



Requiere una fuerte comunicación



Poca formalización en los procesos

© T-CERT®



No es adecuada para proyectos grandes y complejos sin adaptación

- **No es adecuada para proyectos grandes y complejos sin adaptación:** Aunque hay variantes más formales de Crystal (como Crystal Red), la metodología en sí está diseñada para equipos pequeños a medianos. Implementar Crystal en proyectos muy grandes o con estructuras jerárquicas complejas puede ser un desafío.

- **Requiere una fuerte comunicación:** El enfoque de Crystal en la comunicación constante puede ser una ventaja, pero también puede ser un desafío en equipos dispersos geográficamente o en proyectos con comunicación deficiente.



Requiere una fuerte comunicación



Poca formalización en los procesos

- **Poca formalización en los procesos:** Para equipos que buscan una guía más rígida o estructurada, el enfoque flexible de Crystal podría sentirse menos seguro o difícil de seguir.

#### 8.6.5 Comparación con Otras Metodologías Ágiles

**Scrum vs Crystal:** Scrum tiene roles, eventos y artefactos bien definidos, mientras que Crystal es más flexible y se adapta según el tamaño y la complejidad del proyecto. Scrum puede ser más adecuado para proyectos de tamaño medio o grande, mientras que Crystal es ideal para equipos pequeños y proyectos que requieren más flexibilidad.

**Kanban vs Crystal:** Kanban se enfoca en el flujo continuo de trabajo y la gestión visual, mientras que Crystal tiene un enfoque más holístico que incluye la comunicación y la entrega frecuente. Kanban es más adecuado para gestionar el trabajo en curso (WIP), mientras que Crystal se adapta a la metodología de trabajo general y a la entrega de software.

**XP vs Crystal:** Ambas metodologías promueven la entrega frecuente de software funcional y la calidad, pero XP (Extreme Programming) tiene más prácticas técnicas definidas (como pair programming y test-driven development), mientras que Crystal se enfoca más en la flexibilidad del proceso y la importancia de las personas en el equipo.

La metodología Crystal es una excelente opción para equipos pequeños y proyectos en los que se requiere alta flexibilidad. Al centrarse en la comunicación y la entrega continua de valor, permite que los equipos adapten su proceso de trabajo según las necesidades y características del proyecto. Aunque puede ser menos formal que otras metodologías ágiles como Scrum o XP, su adaptabilidad y enfoque en las personas hacen que sea adecuada para proyectos con equipos comprometidos que busquen un proceso ágil, ligero y eficaz.

## 8.7 Metodología FDD (Feature-Driven Development)

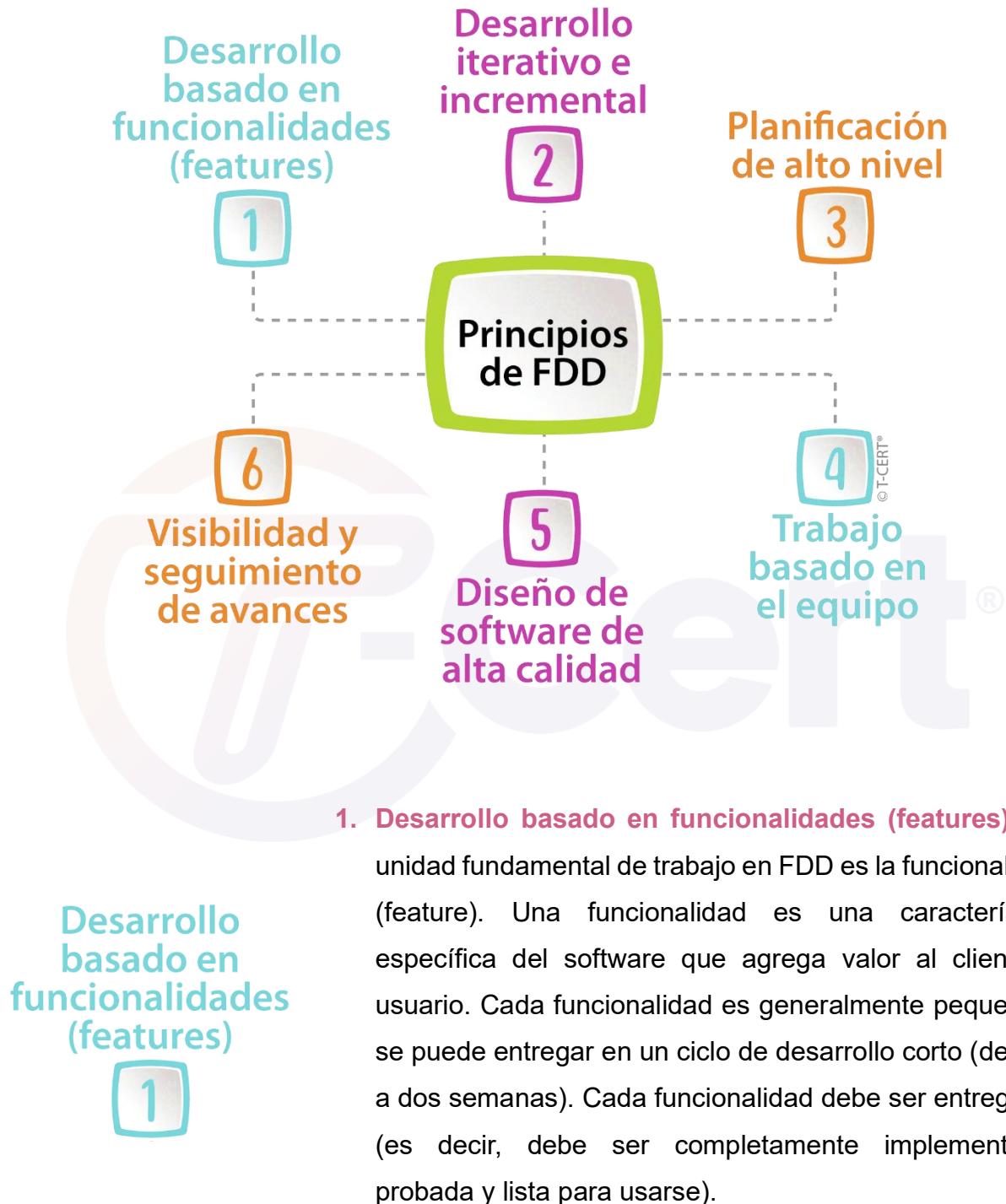
Es un enfoque que se centra en la entrega continua de funcionalidades específicas y bien definidas a lo largo del ciclo de vida del proyecto. Es una de las metodologías ágiles más orientadas al diseño y la planificación estructurada, y es especialmente adecuada para proyectos grandes, con equipos distribuidos y cuando se necesita un enfoque más centrado en el desarrollo de características específicas del producto.

FDD es una metodología ágil creada por Jeff De Luca y Peter Coad a finales de la década de 1990. Su enfoque principal es la entrega incremental de funcionalidades (features), con un fuerte énfasis en el diseño y la planificación a nivel de alto nivel, pero manteniendo la flexibilidad para adaptarse a los cambios a medida que se avanza en el desarrollo.



A diferencia de otros enfoques ágiles, como Scrum o Kanban, que tienden a ser más centrados en el proceso y en la gestión del trabajo, FDD se concentra en el desarrollo de funcionalidades bien definidas que pueden ser entregadas de manera continua y eficiente. Es muy adecuado para proyectos grandes, donde el enfoque en las características ayuda a mantener una estructura clara y el seguimiento de los avances.

### 8.7.1 Principios de FDD



### Desarrollo basado en funcionalidades (features)

1

1. **Desarrollo basado en funcionalidades (features):** La unidad fundamental de trabajo en FDD es la funcionalidad (feature). Una funcionalidad es una característica específica del software que agrega valor al cliente o usuario. Cada funcionalidad es generalmente pequeña y se puede entregar en un ciclo de desarrollo corto (de una a dos semanas). Cada funcionalidad debe ser entregable (es decir, debe ser completamente implementada, probada y lista para usarse).

## Desarrollo iterativo e incremental

2

**2. Desarrollo iterativo e incremental:** FDD sigue un enfoque iterativo e incremental, entregando funcionalidades de forma regular y predecible. Las funcionalidades se desarrollan en ciclos cortos y se entregan continuamente, lo que permite una retroalimentación temprana y la adaptación a los cambios del cliente.

## Planificación de alto nivel

3

**3. Planificación de alto nivel:** En lugar de planificar en detalle para todo el proyecto de una vez, FDD utiliza una planificación de alto nivel basada en el conjunto de funcionalidades que deben desarrollarse. Esto permite a los equipos tener una visión clara de lo que se espera en términos de funcionalidades, pero sin perder la flexibilidad para adaptarse a los cambios conforme avanza el proyecto.

**4. Trabajo basado en el equipo:** FDD promueve una estructura de trabajo en equipo, donde los desarrolladores trabajan en equipos pequeños especializados para desarrollar funcionalidades específicas. Cada funcionalidad generalmente tiene su propio equipo responsable. Los equipos colaboran estrechamente para asegurar que cada característica se complete de manera eficiente.

4  
© T-CERT®  
Trabajo basado en el equipo

## Diseño de software de alta calidad

5

**5. Diseño de software de alta calidad:** FDD pone énfasis en el diseño previo. Antes de comenzar con el desarrollo de las funcionalidades, se realiza un diseño de alto nivel del sistema completo, lo que ayuda a establecer una base sólida y asegura que las funcionalidades que se desarrollen estén bien integradas en la arquitectura del sistema. Esto evita que el sistema se

convierta en un conjunto desarticulado de funcionalidades inconexas.

**6. Visibilidad y seguimiento de avances:** FDD tiene un enfoque fuerte en la medición y seguimiento del progreso. Utiliza una serie de métricas para garantizar que las funcionalidades se entreguen según lo prometido. El progreso se monitorea en función de las funcionalidades completadas, lo que facilita la visualización del avance en tiempo real y asegura que el equipo esté en camino de cumplir con los plazos y objetivos.

6

## Visibilidad y seguimiento de avances

### 8.7.2 Etapas del Desarrollo en FDD



**Desarrollar una visión global del modelo (Develop an Overall Model):** Esta etapa implica crear un modelo de alto nivel del sistema o producto. En esta fase, se lleva a cabo un análisis y diseño general del sistema. Se identifican las áreas principales del sistema, los módulos y

las interacciones entre ellos, lo que proporciona una visión general y permite a los equipos comprender el panorama general antes de comenzar a desarrollar características individuales.

**Construir una lista de características (Build a Feature List):** Una vez que el modelo global ha sido creado, el siguiente paso es desglosarlo en funcionalidades o características. Cada funcionalidad representa una parte funcional del sistema que se puede desarrollar, probar e implementar de manera independiente. Las funcionalidades se definen en términos de lo que debe hacer el sistema (por ejemplo, "Registrar un nuevo usuario" o "Generar un reporte de ventas").



**Planificar las características (Plan by Feature):** Durante esta fase, se planifica cómo y cuándo se implementarán las funcionalidades. Se asignan prioridades a las características y se establecen plazos y recursos necesarios para completarlas. Las funcionalidades se asignan a los desarrolladores, quienes se encargan de su implementación en los siguientes pasos del proceso.

**Diseñar por característica (Design by Feature):** Aquí, se realiza el diseño detallado de cada funcionalidad. Los diseñadores trabajan en conjunto con los desarrolladores para asegurar que cada funcionalidad se pueda integrar correctamente con el sistema. El diseño está enfocado en las características específicas, pero asegurando que encjen con la arquitectura general del sistema, desarrollada en la etapa 1.



**Construir por característica (Build by Feature):** Esta es la fase de implementación. Cada funcionalidad se desarrolla, prueba y entrega de forma independiente. Los desarrolladores trabajan en ciclos cortos, desarrollando y verificando la funcionalidad en su totalidad. Después



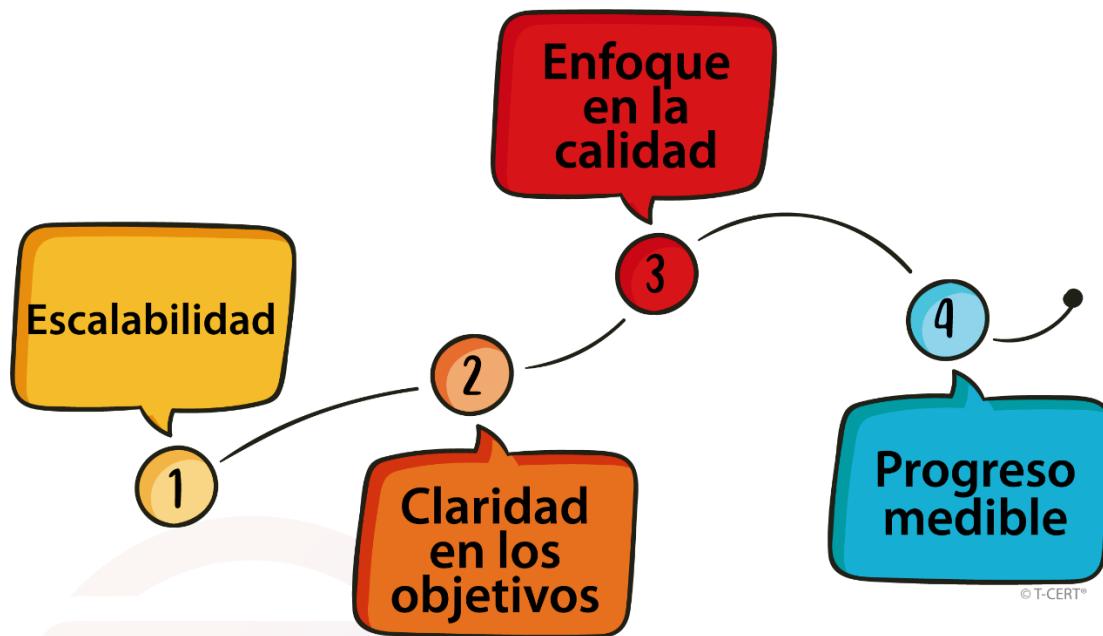
de completar la implementación de cada funcionalidad, se realiza una revisión para garantizar que se haya cumplido con los requisitos establecidos y que la funcionalidad esté lista para entregarse.

### 8.7.3 Roles en FDD

FDD no es una metodología con roles rígidos como en Scrum, pero tiene algunos roles clave que son fundamentales para su éxito:

- **Chief Architect:** Es responsable de la creación del modelo global del sistema y de asegurar que el diseño a nivel de características esté alineado con la arquitectura general.
- **Feature Leaders:** Son los encargados de liderar el desarrollo de características específicas. Cada líder de funcionalidad se encarga de la planificación, el diseño y la construcción de una funcionalidad en particular. Los Feature Leaders son expertos en el área de la funcionalidad que lideran y son responsables de la calidad y el progreso de la misma.
- **Desarrolladores:** Los desarrolladores se encargan de la implementación de las funcionalidades, siguiendo las especificaciones del diseño y asegurando la calidad del código entregado.
- **Project Manager:** El Project Manager en FDD no tiene un papel tan prescriptivo como en otras metodologías ágiles, pero sigue siendo responsable de supervisar el progreso del proyecto y coordinar los esfuerzos entre los distintos equipos.

#### 8.7.4 Ventajas de FDD



**1 Escalabilidad:** FDD es adecuado para proyectos grandes y complejos, ya que está diseñado para gestionar un número elevado de funcionalidades que se pueden entregar de forma incremental.

**2 Claridad en los objetivos:** Las funcionalidades están bien definidas desde el principio, lo que proporciona una clara dirección y enfoque durante el ciclo de vida del proyecto.



**3 Enfoque en la calidad:** FDD promueve la calidad a través de la planificación y el diseño de alto nivel, lo que minimiza el riesgo de tener un sistema desorganizado o defectuoso.

**Progreso medible:** El enfoque de entregas incrementales basado en funcionalidades permite medir fácilmente el progreso del proyecto, lo que proporciona visibilidad tanto para el equipo como para los stakeholders.



#### 8.7.5 Desventajas y Desafíos de FDD



© T-CERT®



- **Requiere un esfuerzo inicial considerable:** La fase de planificación y diseño de alto nivel puede ser más intensiva al principio, lo que requiere un esfuerzo significativo antes de comenzar el desarrollo real.
- **Menos flexible en la fase de diseño:** Si bien FDD es ágil, la metodología tiene un enfoque más estructurado en la fase de diseño y planificación, lo que puede ser percibido como más rígido en comparación con metodologías como Scrum o XP.





Dependencia de los líderes de características

- **Dependencia de los líderes de características:** FDD pone un énfasis considerable en los Feature Leaders, lo que puede ser un problema si no se cuenta con personas adecuadas o si el equipo no tiene suficiente experiencia.
- **No es ideal para equipos pequeños:** FDD es mejor para proyectos grandes con equipos distribuidos, por lo que puede no ser la mejor opción para equipos pequeños o proyectos muy simples.



No es ideal para equipos pequeños

#### 8.7.6 Con relación a otras Metodologías Ágiles

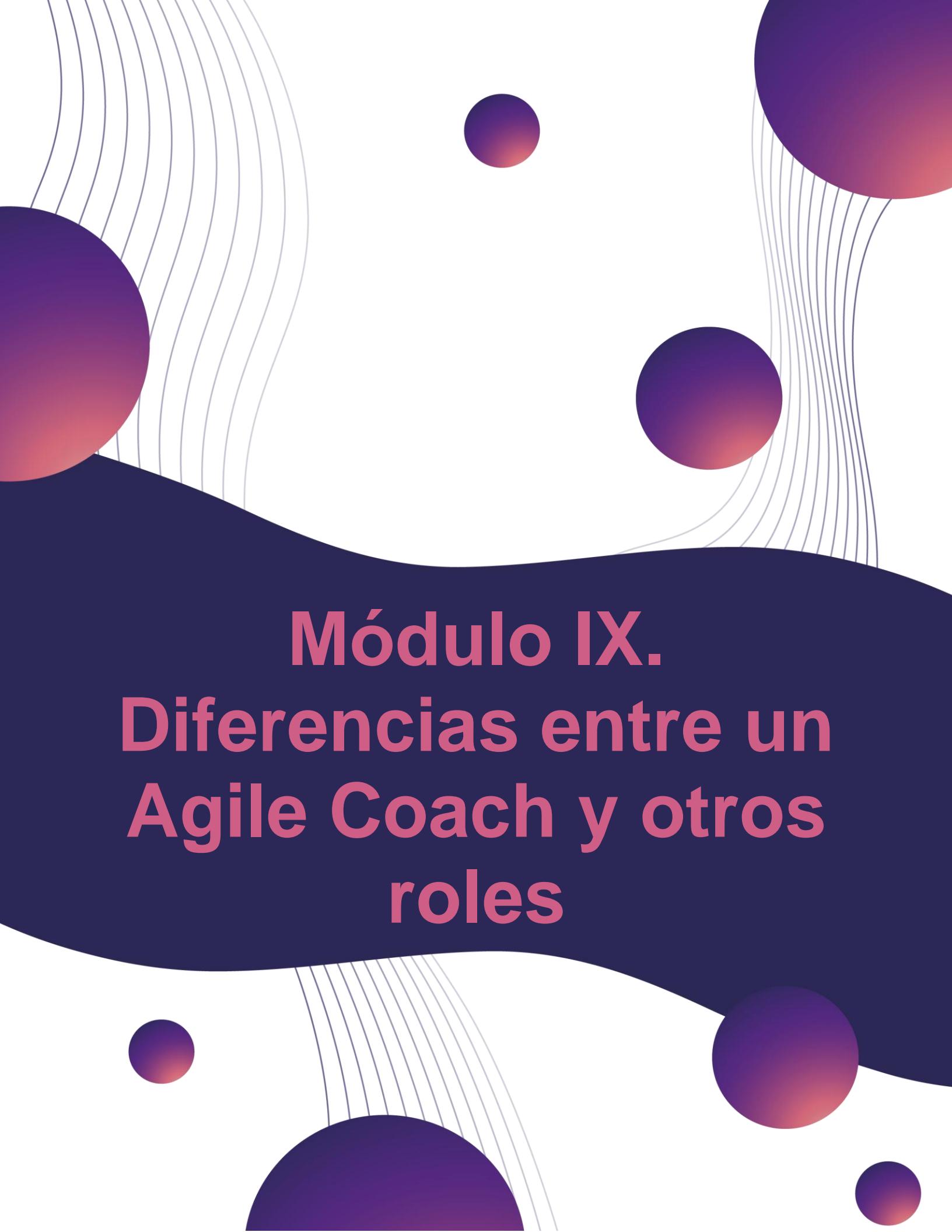
**Scrum vs FDD:** Mientras que Scrum tiene un enfoque más flexible y adaptativo con ciclos de trabajo llamados sprints, FDD es más estructurado y se enfoca en la entrega de funcionalidades específicas. FDD tiende a ser más adecuado para proyectos grandes y complejos, mientras que Scrum es ideal para proyectos más pequeños o menos complejos.

**XP vs FDD:** XP (Extreme Programming) es muy técnico y pone mucho énfasis en la calidad del código (a través de prácticas como el pair programming y test-driven development), mientras que FDD está más orientado al diseño y entrega incremental de funcionalidades. FDD también tiene un enfoque más estructurado en el análisis y planificación.

FDD (Feature-Driven Development) es una metodología ágil que pone un fuerte énfasis en la entrega continua de funcionalidades específicas del sistema, con un enfoque sólido en la planificación, diseño y seguimiento.

Es especialmente efectiva para proyectos grandes y complejos, donde se necesita claridad en los objetivos y un enfoque organizado para manejar múltiples funcionalidades. Si bien puede no ser tan flexible como otras metodologías ágiles como Scrum o XP, su enfoque en las funcionalidades bien definidas y la calidad del diseño lo convierte en una excelente opción para proyectos que requieren un control claro y continuo sobre el progreso y la calidad del producto.





# Módulo IX. Diferencias entre un Agile Coach y otros roles

## Módulo IX. Diferencias entre un Agile Coach y otros roles

### 9.1 Agile Coach vs Scrum Master

Aunque ambos roles están relacionados con la implementación de metodologías ágiles, el Scrum Master se enfoca específicamente en la aplicación de Scrum, mientras que el Agile Coach tiene una perspectiva más amplia y puede trabajar con diferentes marcos ágiles y en diferentes áreas de la organización.

### 9.2 Agile Coach vs Product Owner

Mientras que un scrum master se centra en facilitar el proceso scrum para un solo equipo, y un Product Owner se enfoca en maximizar el valor del producto, un Agile Coach suele trabajar a un nivel más alto, ayudando a múltiples equipos y a la organización en su conjunto a trabajar de forma ágil. Debe hacerlo de forma sostenible en el tiempo y adaptada a las circunstancias de la organización.

### 9.3 Agile Coach vs Project Manager

Mientras que el Project Manager tradicional se centra en la planificación, ejecución y control de proyectos, el Agile Coach se enfoca en el cambio cultural y la mejora continua en toda la organización, trabajando para crear un entorno ágil que permita a los equipos alcanzar su máximo potencial.

