**FORMATO PARA EL DESARROLLO DE COMPONENTE FORMATIVO**

|  |  |
| --- | --- |
| PROGRAMA DE FORMACIÓN | ADSO |

|  |  |  |  |
| --- | --- | --- | --- |
| COMPETENCIA | 220501093. Evaluar requisitos de la solución de *software* de acuerdo con metodologías de análisis y estándares. | RESULTADOS DE APRENDIZAJE | 220501093-01. Planear actividades de análisis de acuerdo con la metodología  seleccionada. |

|  |  |
| --- | --- |
| NÚMERO DEL COMPONENTE FORMATIVO | 03 |
| NOMBRE DEL COMPONENTE FORMATIVO | Metodologías de desarrollo de *software* |
| BREVE DESCRIPCIÓN | Las metodologías de desarrollo de *software* proponen un conjunto de procesos y actividades que deben ser desarrolladas por el equipo de desarrollo de *software* para realizar un trabajo organizado que sea fácil hacerle seguimiento y de esta forma establecer planes de mejora en busca de una mejor calidad de los productos y servicios que se desarrollan. |
| PALABRAS CLAVE | Metodologías, *scrum,* cascada, *software,* tecnología |

|  |  |
| --- | --- |
| ÁREA OCUPACIONAL | 2 - CIENCIAS NATURALES, APLICADAS Y RELACIONADAS |
| IDIOMA | Español |

1. **TABLA DE CONTENIDOS:**

1. Metodologías de desarrollo de *software*

2. Marcos de trabajos tradicionales

2.1. Cascada

2.2. Proceso Racional Unificado – RUP

3. Marcos de trabajo ágiles

3.1. Programación Extrema – XP

3.2. Desarrollo rápido de aplicaciones – RAD

3.3. *Scrum*

4. Planeación de proyectos de software

1. **INTRODUCCIÓN**

El mundo del desarrollo de *software* es complejo y variado, con una amplia gama de metodologías y marcos de trabajo diseñados para enfrentar los retos que supone la creación de sistemas efectivos y eficientes. En este componente, se exploran las principales metodologías y marcos de trabajo que han moldeado la ingeniería de *software* moderna.

Con estas bases establecidas, se adentrará en el corazón de cada metodología y marco de trabajo, buscando no solo entender su propósito y prácticas, sino también cómo y cuándo implementarlas de manera efectiva en los proyectos de *software.*

1. **DESARROLLO DE CONTENIDOS:**

# **Metodologías de desarrollo de *software***

Es importante aclarar que existe un gran número de definiciones sobre lo que es una metodología para evitar cualquier confusión en este material se utilizará la definición dada por Maida y Pacienzia (2015), quienes indican que:

|  |  |
| --- | --- |
| conjunto de iconos de mezquita ilustración vectorial | Una metodología hace referencia a un conjunto de procedimientos genéricos y lógicos que se usan para alcanzar un objetivo particular utilizando un conjunto de habilidades y conocimientos. |

Las metodologías de desarrollo de *software* siempre parten de un componente teórico y, cuando son usadas por los equipos de trabajo, conllevan a la utilización de un conjunto de técnicas y métodos que al final determinarán las tareas generales y específicas que se deberían realizar para alcanzar un objetivo.

Existen dos grandes clasificaciones de metodologías de desarrollo de *software* que se agrupan generalmente como marcos de trabajo tradicionales o marcos de trabajo ágiles que se presentan a continuación.

# **Marcos de trabajos tradicionales**

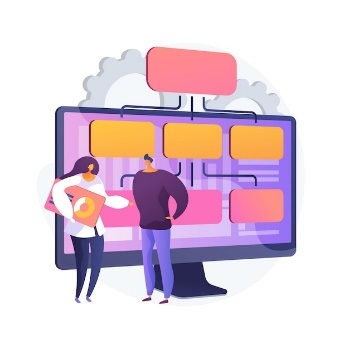
Para el desarrollo de un buen producto de *software* se debe iniciar con un excelente proceso de planificación y gestión durante todas las etapas y actividades que involucran transformar una idea o requerimiento en un producto o servicio que será utilizado por un cliente particular.

|  |  |
| --- | --- |
| Diseño de estilo plano de flujo de trabajo Ilustración vectorial Ilustración de stock | **“**Los marcos de trabajo o metodologías tradicionales se caracterizan por centrar la mayor parte de su esfuerzo en la planificación y control del proceso, lo que conlleva a una documentación exhaustiva y precisa de los artefactos que describen los requisitos y los modelos del sistema en las etapas iniciales del desarrollo del proyecto.**”**  (Maida y Pacienzia, 2015). |

Lo anterior supone que este tipo de enfoques son óptimos en proyectos en los cuales los requisitos están plenamente identificados y delimitados, y donde no se espera que se produzca ningún cambio en lo establecido mientras el proyecto es finalizado.

A continuación, se describen algunas metodologías que se enmarcan en los marcos tradicionales de desarrollo de software.

## **2.1. Cascada**



Este es uno de los modelos genéricos más ampliamente conocidos en la ingeniería de *software* y se deriva de procesos de ingeniería de sistemas más generales (Royce, 1970). Este modelo plantea un proceso lineal donde las actividades de desarrollo de un producto o servicios de *software* se agrupan en un conjunto de fases sucesivas. Estas son desarrolladas una única vez y los resultados de cada fase son la entrada requerida para la siguiente, ninguna fase puede iniciar si la fase anterior no ha sido finalizada, generalmente mediante un formalismo que puede ser un documento.

Según Sommerville (2005), el modelo en cascada se compone de cinco etapas principales que se asocian con actividades fundamentales en el proceso de desarrollo de *software*, las cuales son:

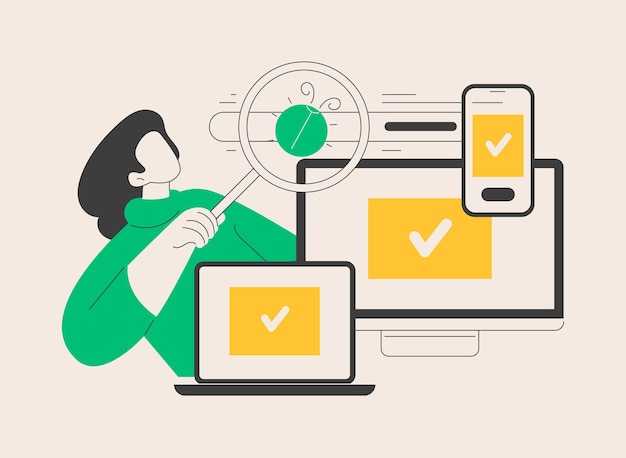
|  |
| --- |
| **Infografía interactiva**  **CF03\_2.1. Cascada (1)** |

Continuando con el tema, se presenta un comparativo que resume las principales ventajas y desventajas de este modelo.

El modelo en cascada define cuatro grupos de roles típicos los cuales se mencionan a continuación:

|  |
| --- |
| **Tarjetas**  **CF03\_2.1. Cascada (2)** |

## **2.2. Proceso Racional Unificado – RUP**

RUP es una sigla en inglés que corresponde a *Rational Unified Proces*s, el cual es un proceso de desarrollo de *software* tradicional que se basa en el modelo en cascada y fue desarrollado por la empresa Rational Software, actualmente propiedad de IBM. Esta metodología se enfoca en la arquitectura y es guiada por casos de uso (requerimientos) (Kruchten, 2003).

RUP divide el proceso de desarrollo en cuatro grandes fases. Dentro de ellas, se realizan iteraciones donde se desglosan, con mayor o menor intensidad, un conjunto de disciplinas según la fase en la que se está.

A continuación, se presentarán las fases y disciplinas propuestas por RUP.

**Figura 1.** Fases y disciplinas de RUP

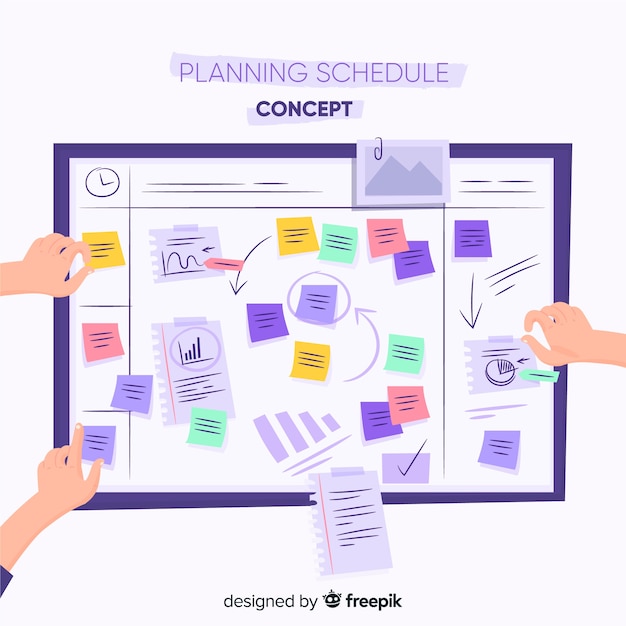
En la figura Fases y disciplinas del RUP se definen 9 flujos de trabajo de procesos como son: Modelado de negocios, Requerimientos, Análisis y diseño, Implementación, Prueba, Desarrollo, Configuración y administración del cambio, Administración del proyecto y Ambiente. Combinado con 4 fases: Incepción, Elaboración, Construcción y Transición. Cada una con interacciones respectivas.


A continuación, se describen las fases y su impacto en el proceso, por lo que se le invita a revisar el siguiente recurso de aprendizaje.

|  |
| --- |
| **Pestañas**  **CF03\_2.2\_Proceso Racional Unificado (1)** |

|  |  |
| --- | --- |
|  | Es importante señalar que, dentro de cada una de las fases del RUP, se desarrollan varias disciplinas, y el esfuerzo y tiempo invertidos en cada una tendrán un enfoque diferente dependiendo de la fase del proyecto en la que se esté. |

Otro aspecto relevante de esta metodología es que se sustenta en un conjunto de artefactos que se elaboran con el objetivo de especificar el proceso de análisis y diseño que respalda la construcción del *software.* Entre los artefactos más comunes se incluyen:

1. Documento de visión.
2. Documento de especificación de requisitos.
3. Diagramas de casos de uso.
4. Modelos conceptuales (clases y entidad relación).
5. Diagramas que representan la vista de implementación como:

* Diagramas de secuencia.
* Diagramas de estados.
* Diagramas de colaboración, entre otros.

Las disciplinas, por otra parte, representan un conjunto de actividades relacionadas con un área específica del proyecto y están inspiradas en el modelo en cascada. RUP establece las siguientes disciplinas según Kruchten (2003):

A continuación, se describen cada una de las disciplinas mencionadas y cómo estás aportan en el proceso.

RUP propone una categorización de roles encargados de la realización de actividades dentro de cada una de las disciplinas que son:

|  |
| --- |
| **Slides**  **CF03\_2.2\_Proceso Racional Unificado (2)** |

Existen también metodologías catalogadas dentro de los marcos de trabajo tradicionales. Sin embargo, el propósito de este texto es resaltar las características principales de las más destacadas en la industria del software. Para aquellos interesados en profundizar sobre las ventajas y desventajas de estas y otras metodologías, se recomienda revisar la sección de materiales complementarios.

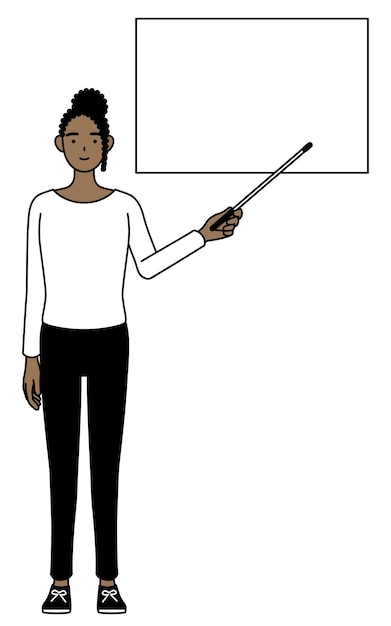
# **Marcos de trabajo ágiles**

Los marcos de trabajo ágiles, también conocidos como metodologías ágiles para el desarrollo de *software*, surgen como alternativa para gestionar proyectos en los que no es posible definir completamente los requerimientos y sus estimaciones en la fase inicial, o cuando se requiere adaptabilidad a lo largo del desarrollo del *software.* (Maida y Pacienzia, 2015).

Estas metodologías ofrecen un conjunto de pautas y principios diseñados para facilitar y dar prioridad a la entrega de productos por encima de los procesos de documentación exhaustiva, simplificándolos y promoviendo la participación activa del cliente desde las primeras etapas del proyecto.

El origen de las metodologías ágiles se remonta al año 2001 con la publicación del Manifiesto Ágil para el Desarrollo de *Software,* que establece cuatro valores fundamentales. (Manifiesto Ágil, 2001):

Adicionalmente a los valores ágiles anteriormente listados, el manifiesto ágil establece 12 principios ágiles para materializar los valores definidos (Manifiesto Ágil, 2001):

1. Nuestra principal prioridad es satisfacer al cliente a través de la entrega temprana y continua de *software* de valor.
2. Son bienvenidos los requisitos cambiantes, incluso si llegan tarde al desarrollo. Los procesos ágiles se doblegan al cambio como ventaja competitiva para el cliente.
3. Entregar con frecuencia *software* que funcione, en periodos de un par de semanas hasta un par de meses, con preferencia en los períodos breves.
4. Las personas del negocio y los desarrolladores deben trabajar juntos de forma cotidiana a través del proyecto.
5. Construcción de proyectos en torno a individuos motivados, dándoles la oportunidad y el respaldo que necesitan y procurándoles confianza para que realicen la tarea.
6. La forma más eficiente y efectiva de comunicar información de ida y vuelta dentro de un equipo de desarrollo es mediante la conversación cara a cara.
7. El *software* que funciona es la principal medida del progreso.
8. Los procesos ágiles promueven el desarrollo sostenido. Los patrocinadores, desarrolladores y usuarios deben mantener un ritmo constante de forma indefinida.
9. La atención continua a la excelencia técnica enaltece la agilidad.
10. La simplicidad como arte de maximizar la cantidad de trabajo que se hace es esencial.
11. Las mejores arquitecturas, requisitos y diseños emergen de equipos que se auto organizan.
12. En intervalos regulares, el equipo reflexiona sobre la forma de ser más efectivo y ajusta su conducta en consecuencia.

## **3.1 Programación Extrema – XP**

XP, abreviatura de *Extreme Programming,* es un marco de desarrollo de *software* ágil diseñado para producir *software* de alta calidad en entornos con requisitos muy cambiantes. Se enfoca en contextos que presentan riesgos asociados a plazos fijos, la implementación de tecnologías emergentes y equipos de trabajo reducidos localizados en el mismo lugar.

En XP se definen cinco valores según Beck y Andrés (2004):

|  |
| --- |
| **Pestañas**  **CF03\_3.1 Programación Extrema (1)** |

Además de sus valores, XP se distingue por establecer un conjunto de 12 prácticas de desarrollo de software. Aunque estas prácticas pueden adoptarse de manera independiente, su importancia se magnifica cuando se implementan conjuntamente (Jeffries, 2011).

|  |
| --- |
| **Acordeón**  **CF03\_3.1 Programación Extrema (2)** |

Los roles fundamentales establecidos por este marco de trabajo ágil son los siguientes: cliente, programador, *coach, tester y mánager.*

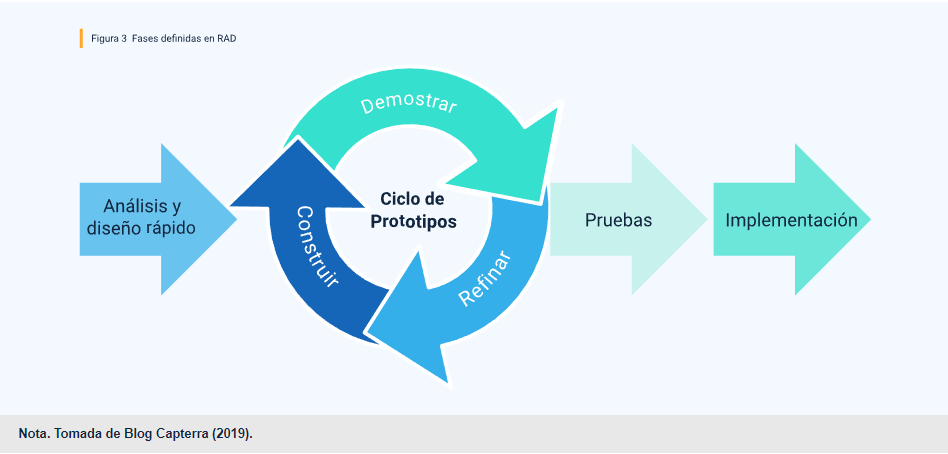
## **3.2. Desarrollo rápido de aplicaciones – RAD**

RAD (Desarrollo Rápido de Aplicaciones) es una metodología ágil de desarrollo de software que se enfoca en la rápida creación de aplicaciones a través de iteraciones frecuentes y retroalimentación constante. Fue propuesta por James Martin en 1991. Algunas características fundamentales de RAD son:

1. Mayor flexibilidad y adaptabilidad a cualquier ajuste que deba realizarse durante el proceso de desarrollo.
2. Iteraciones rápidas que reducen el tiempo de desarrollo y mantienen un ritmo de entrega acelerado.
3. Se fomenta la reutilización de código.
4. Mejor gestión del riesgo, ya que las partes interesadas discuten y abordan cualquier vulnerabilidad mientras se construyen los productos.

A continuación, se describen las fases propuestas en RAD.

**Figura 2.** Fases definidas en RAD



Nota: Tomada de Blog Capterra (2019).

El proceso de desarrollo en cinco fases clave: desde la definición precisa de los requisitos, pasando por la creación y afinamiento de prototipos, hasta las pruebas rigurosas y el lanzamiento cuidadosamente preparado del producto final.

|  |
| --- |
| **Slide**  **CF03\_3.2. Desarrollo rápido de aplicaciones (1)** |

Según HKSAR (2009), los roles más importantes en la metodología RAD incluyen: el facilitador, el escriba, el equipo SWAT, el administrador del modelo, el administrador de la base de datos, el equipo de planificación del taller, el equipo de diseño de usuario, el equipo de soporte de construcción y el equipo de transición. A continuación, se describen las características más destacadas de cada rol.

|  |
| --- |
| **Acordeón**  **CF03\_3.2. Desarrollo rápido de aplicaciones (2)** |

## **3.3. *Scrum***

Scrum es un marco de trabajo ágil de muy amplio uso en la industria del software que se fundamenta en los valores y principios ágiles definidos en (Manifiesto Ágil, 2001) y donde se definen tres pilares fundamentales según (SCRUMstudy, 2013) los cuales se describen a continuación:

Adicionalmente, este marco de trabajo ágil está estructurado por un conjunto de roles, eventos y artefactos que pueden ser presentados a continuación:

**Figura 3. Marco de trabajo *Scrum***

En la figura marcos de trabajo Scrum, aparecen entradas como ejecutivos, equipo, interesados, clientes y usuarios. Pasan al Dueño del producto quien se encarga de emitir una lista priorizada de lo que es requerido en las entradas. 
El equipo selecciona el conjunto de elementos de la pila del producto a la que se compromete durante el sprint. 
El sprint pasa por diferentes tareas de las cuales están a cargo cada uno de los equipos que conforman los roles, puede tomar un tiempo de 1 a 4 semanas, para entregar el producto finalizado. Finalmente se realiza reunión de retrospectiva del Sprint

Dentro de los roles hay una división en dos categorías fundamentales:

Por otro lado, hay tres roles centrales dentro del marco de trabajo de *Scrum* (SCRUMstudy, 2013) que se describen a continuación:

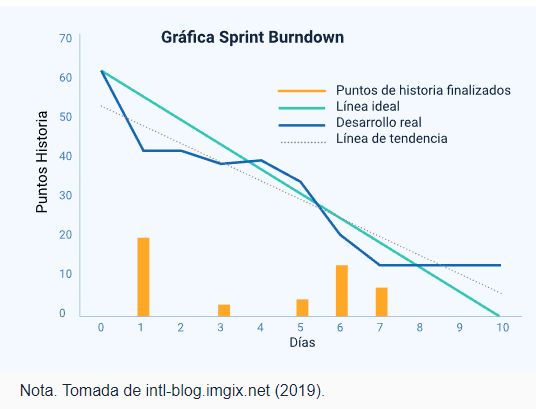
Además de los roles, *scrum* define un conjunto de eventos con participantes y objetivos claros que se desarrollan en momentos particulares del flujo general de *scrum,* a continuación, se detalla cada uno de estos:

|  |
| --- |
| **Slide**  **CF03\_3.3. Scrum** |

Finalmente, el marco de trabajo *scrum* define un conjunto de artefactos que permiten registrar y gestionar información clave para asegurar los tres pilares fundamentales y proveen información valiosa durante todo el proceso de desarrollo de *software.* Entre los artefactos representativos de este se encuentran los siguientes:

Por otro lado, también es posible usar este mismo gráfico para representar el avance general del proyecto ubicando en el eje Y la cantidad total de horas o esfuerzo del *Product Backlog* y en el eje X el número de *Sprints* proyectados. Cada punto representa el total acumulado hasta ese *Sprint* y se une mediante una línea, lo que permite determinar visualmente si el flujo de trabajo está en una situación óptima o no respecto al tiempo restante para completar el *Sprint* actual.

**Figura 4.** Burndown Chart



Nota. Tomada de intl-blog.imgix.net (2019).

**Tablero de *Scrum (Scrumboard)*:** es un elemento visual donde se integra la mayor parte de los elementos del marco de trabajo *Scrum*, en él se indica la carga de trabajo, el estado actual de cada una de las actividades y sus respectivos responsables. Este es un elemento que se sincroniza de manera permanente y facilita la implementación de los pilares de transparencia, inspección y adaptabilidad. Si bien se aconseja el uso de un tablero, existen diferentes tipos de herramientas digitales que permiten la implementación de un tablero de *Scrum.*

**Figura 5.** Tablero de *Scrum*



Adicionalmente es de vital importancia mencionar que entre los principales beneficios del marco de trabajo *Scrum* se encuentran los siguientes elementos:



1. Es posible gestionar las expectativas del cliente de manera regular, ya que, este puede y debe participar en las reuniones de revisión, por lo que, está enterado todo el tiempo del estado actual del proyecto.
2. El cliente puede obtener resultados importantes y utilizables desde las primeras iteraciones, ya que, la lista de producto está priorizada para ofrecer mayor valor en el menor tiempo posible y porque cada finalización de Sprint debe tener como resultado una versión totalmente funcional.
3. El proyecto puede iniciar con requerimientos de muy alto nivel y es fácil administrar los cambios.
4. La participación constante del cliente permite mitigar riesgos del proyecto desde sus primeras etapas.
5. Los procesos de retrospectiva permiten establecer actividades permanentes de mejora continua en función de las experiencias del equipo.

Un elemento clave en su formación es la planeación de proyectos de *software* que aunado con los contenidos ya vistos le permite tener un panorama amplio sobre el tema.

# **Planeación de proyectos de *software***

La planificación de proyectos de *software* comprende varias actividades destinadas a obtener una visión preliminar del *software* a desarrollar, lo que permite realizar estimaciones para determinar la viabilidad del proyecto en función de los recursos asignados.

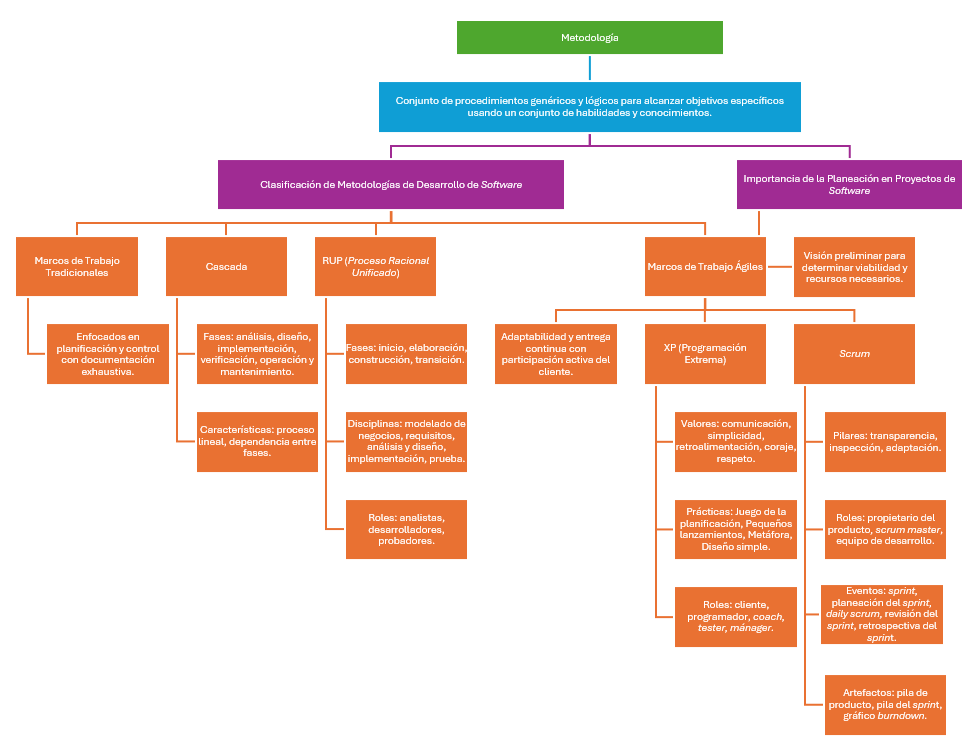
De acuerdo con SCRUMstudy (2013), las actividades de planificación estipuladas en el marco de trabajo de *Scrum* se detallan en el siguiente recurso de aprendizaje:

**Dejar espacio para video** Planeación de proyectos de software



1. **SÍNTESIS**

A continuación, se presenta una síntesis de la temática estudiada en el componente formativo.



1. **ACTIVIDADES DIDÁCTICAS (Se debe incorporar mínimo 1, máximo 2)**

|  |  |
| --- | --- |
| DESCRIPCIÓN DE ACTIVIDAD DIDÁCTICA | |
| Nombre de la Actividad | Evaluación de términos en metodologías de desarrollo de *software* |
| Objetivo de la actividad | Asociar conceptos clave con sus descripciones correspondientes para reforzar el conocimiento de las metodologías de desarrollo de *software.* |
| Tipo de actividad sugerida |  |
| Archivo de la actividad  (Anexo donde se describe la actividad propuesta) | *El ejercicio siempre debe tener realimentación positiva sobre las respuestas que del aprendiz… si queda mal o bien* |

1. **MATERIAL COMPLEMENTARIO:**

|  |  |  |  |
| --- | --- | --- | --- |
| Tema | Referencia APA del Material | Tipo de material  (Video, capítulo de libro, artículo, otro) | Enlace del Recurso o  Archivo del documento o material |
| ¿Qué son las metodologías tradicionales en el desarrollo de software? | Henao, C. (2018). #1. ¿Qué son las metodologías tradicionales en el desarrollo de software?. | Video | <https://www.youtube.com/watch?v=i8CPD1dW88k> |
| ¿Qué son las metodologías ágiles en el desarrollo de Software? | Henao, C. (2018b). #2. ¿Qué son las metodologías ágiles en el desarrollo de Software? | Video | <https://www.youtube.com/watch?v=i8CPD1dW88k> |
| SCRUM en 6 minutos | Metodologías Ágiles | Henao, C. (2018d). #3. SCRUM en 6 minutos | Metodologías Ágiles. | Video | <https://www.youtube.com/watch?v=i8CPD1dW88k> |

1. **GLOSARIO:**

|  |  |
| --- | --- |
| TÉRMINO | SIGNIFICADO |
| Iteración: | hace referencia a un ciclo limitado por tiempo donde se ejecutan actividades de análisis, diseño, construcción y pruebas. |
| *Product Owner*: | rol central de *Scrum* encargado de la gestión de la pila de producto y representante del cliente dentro del grupo de trabajo. |
| *Scrum Master:* | rol central de *Scrum* encargado de facilitar el trabajo del equipo de desarrollo y de mantener la aplicación del marco de trabajo *SCRUM.* |
| *Stakeholders:* | persona, organización o empresa interesada en el desarrollo del proyecto. |

1. **REFERENCIAS BIBLIOGRÁFICAS:**

Agilemanifesto.org. (2021). Manifiesto por el desarrollo ágil de software. Agilemanifesto.org. <https://agilemanifesto.org/iso/es/manifesto.html>

Beck, K., & Andrés, C. (2004b). Extreme Programming Explained: Embrace Change, 2nd Edition (The XP Series). Addison-Wesley.

Intl-blog.imgix.net. (2019). Burndown-chart. <https://intl-blog.imgix.net/wp-content/uploads/2019/12/Burndown-chart.png?auto=format%2Cenhance>

Jeffries, R. (2011). What is Extreme Programming? Ronjeffries.com. <https://ronjeffries.com/xprog/what-is-extreme-programming/>

Kruchten, P. (2003). The Rational Unified Process: An Introduction. Addison-Wesley Professional.

Maida, E, G., Pacienzia, J. (2015). Metodologías de desarrollo de software [en línea]. Tesis de Licenciatura en Sistemas y Computación. Facultad de Química e Ingeniería Fray Rogelio Bacon. Universidad Católica Argentina. <https://bit.ly/3hJMwXP>

Man, M., Hafriz, M., Nural, A., Mohd, H., Maizura, N., Noor, M., Wan, W., Bakar, A., & Man, M. (2008). eWorks: Development of a Web Based Site Assessment Software for Construction Progress Project. Communications of the IBIMA, (5), 93-99.

Martin, J. (1991). Rapid Application Development. Macmillan Coll.

Pngwing.com (s.f.) Marco de trabajo Scrum. <https://www.pngwing.com/es/free-png-xqgjv>

Royce, W. W. (1970). Managing the Development of Large Software Systems. Proceedings of IEEE WESCON, 26, 328-388.

SCRUMstudy. (2013). A Guide to the Scrum Body of Knowledge (SBOK Guide). VMEdu Inc.

Sommerville, I., Galipienso, M. I. A., y Martínez, A. B. (2005). Ingeniería del software. Pearson Educación.

1. **CONTROL DEL DOCUMENTO**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Nombre | Cargo | Dependencia | Fecha |
| Autor (es) | Jonathan Guerrero Astaiza | Experto Temático | Regional Cauca - Centro de teleinformática y producción industrial | Abril de 2024 |
| Paola Alexandra Moya | Evaluadora instruccional | Regional Antioquia - Centro de Servicios de Salud | Abril de 2024 |
|  | Olga Constanza Bermúdez Jaimes | Responsable Línea de Producción Antioquia | Regional Antioquia - Centro de Servicios de Salud | Abril de 2024 |

1. **CONTROL DE CAMBIOS**

**(Diligenciar únicamente si realiza ajustes a la Unidad Temática)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Nombre | Cargo | Dependencia | Fecha | Razón del Cambio |
| Autor (es) |  |  |  |  |  |