**COMPONENTE FORMATO PARA EL DESARROLLO DE COMPONENTE FORMATIVO**

| PROGRAMA DE FORMACIÓN | Análisis y desarrollo de *software* |
| --- | --- |

| COMPETENCIA | 220501093 - Análisis de la especificación de requisitos del *software.* | RESULTADOS DE APRENDIZAJE | 220501093\_01 Planear actividades de análisis de acuerdo con la metodología seleccionada. |
| --- | --- | --- | --- |

| NÚMERO DEL COMPONENTE FORMATIVO | 6 |
| --- | --- |
| 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 | Scrum, Xp, RUP, agíl, marcos de trabajo. |

| ÁREA OCUPACIONAL | 2 - Ciencias naturales, aplicadas y relacionadas |
| --- | --- |
| IDIOMA | Español |

1. **TABLA DE CONTENIDOS:**

**Introducción**

1. **Metodologías de desarrollo de *software***
2. **Marcos de trabajo tradicionales**
   1. Cascada
   2. Proceso Racional Unificado - RUP
3. **Marcos de trabajo ágiles**
   1. Programación extrema - XP
   2. Desarrollo rápido de aplicaciones - RAD
   3. SCRUM
4. **Planeación de proyectos de *software***
5. **DESARROLLO DE CONTENIDOS:**

**Introducción**

| FORMATOS DI con el código: DI\_CF6\_VIDEO\_INTRODUCCIÓN |
| --- |

1. **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 utiliza la definición dada en Maida y Pacienzia (2015), quienes indican que una metodología hace referencia a un conjunto de procedimientos genéricos y lógicos que se utilizan para alcanzar un objetivo particular usando 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 diferentes tipos de metodologías de desarrollo de *software* que fueron ideadas pensando en problemas particulares presentados en la industria en contextos específicos, por lo cual es importante conocer sus diferentes características y contrastarlas con las necesidades particulares a las que se enfrenta a la hora de desarrollar un producto y servicio. Es decir, cada una de estas tiene ventajas y enfoques que pueden ser reutilizados en diferentes momentos.

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 verán a continuación.

1. **Marcos de trabajo tradicionales**

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

Los marcos de trabajo tradicionales o metodologías tradicionales se caracterizan por centrar la mayor parte de su esfuerzo en la planeación 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, donde no se producirá 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 conocido 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 donde estas son desarrolladas una única vez y los resultados de cada fase son la entrada requerida para cada fase subsiguiente, 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 (5) etapas principales que se asocian con actividades fundamentales en el proceso de desarrollo de *software*, las cuales son:

| RECURSO GAMA MEDIA 1\_ MAPA MENTAL  DI\_CF6\_2.1\_Cascada |
| --- |

Las anteriores fases se pueden visualizar en la siguiente figura donde se sintetiza la información anteriormente mencionada.

**Figura 1***.*

*Etapas del modelo cascada*



Siguiendo con el tema se presenta un cuadro comparativo donde se resumen las principales ventajas y desventajas de este modelo.

***Tabla 1***

*Ventajas y desventajas del modelo cascada.*

| Ventaja | Desventaja |
| --- | --- |
| La definición clara de fases permite el desarrollo de una estructura sencilla que es ideal para proyectos sencillos y cortos. | No se acopla muy bien a proyectos complejos con múltiples equipos trabajando en paralelo ya que las fases normalmente se traslapan y es difícil diferenciarlas. |
| Siguiendo este modelo se genera una muy buena documentación del proceso y es posible definir hitos claros. | Es difícil introducir cambios en el transcurso del proyecto. |
| Es más fácil realizar la estimación de costos al inicio del proyecto. | Los usuarios finales y clientes normalmente son integrados al final del proceso lo que impide tener realimentación y ajustes en etapas tempranas. |
| Es fácil elaborar cronogramas de trabajo en base al desarrollo de las actividades de cada fase. | Existen fallos que solo son detectados cuando el sistema entra en funcionamiento, lo que puede ser desastroso para un proyecto que sigue este modelo. |

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

* **Desarrolladores**: es el rol más importante en la metodología cascada ya que son los encargados directos de la creación de código.
* ***Testers*:** encargados de encontrar fallas en los productos finales y retornar el *software* a los desarrolladores para arreglar todos los defectos.
* **Analista del negocio**: encargado de la realización de estrategias de negocio que le permitan al producto *software* alcanzar popularidad en el mercado digital.
* **Administrador del proyecto**: es responsable de la calidad final del *software*. Administra el proyecto y lo subdivide en tareas entre los miembros del equipo.

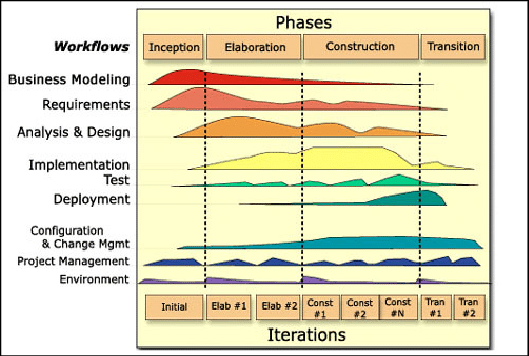
**2.2 Proceso Racional Unificado - RUP**

RUP es una sigla en inglés equivalentes a Proceso Racional Unificado, el cual es un proceso de desarrollo de *software* tradicional basado en el modelo cascada y que fue desarrollado por la empresa Rational Software que es propiedad de IBM, esta metodología se centra en la arquitectura y es guía por casos de uso (requerimientos) (Kruchten, 2003).

RUP divide el proceso de desarrollo en cuatro grandes fases, dentro de las que se realizan algunas iteraciones donde se desglosan, en mayor o menor intensidad, un conjunto de disciplinas según la fase que se está abordando. A continuación, se observan las fases y disciplinas propuestas por RUP.

**Figura 2**

*Fases y disciplinas de RUP*



Nota. Tomada de Man, Hafriz, Nural, Maizura, Noor, Wan, Bakar y Man (2008).

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.

| RECURSO DE APRENDIZAJE GAMA MEDIA N. 2: INFOGRAFÍA  DI\_CF6\_2.2\_RUP\_FASES |
| --- |

Es importante aclarar que dentro de cada una de las fases se desarrollan varias de las disciplinas, pero el esfuerzo y tiempo que se invierte en ellas tendrá mayor o menor enfoque dependiendo de la fase en la que se encuentre el proyecto.

Otro aspecto importante de esta metodología es que se apoya en un conjunto de artefactos que son desarrollados con el fin de especificar el proceso de análisis y diseño que soporta la construcción del *software*, entre los artefactos más comunes se encuentran:

* Documento de visión.
* Documento de especificación de requisitos.
* Diagramas de casos de uso.
* Modelos conceptuales (clases y entidad relación).
* 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):

* Modelado de negocios.
* Requerimientos.
* Análisis y diseño.
* Implementación.
* Pruebas.
* Despliegue.
* Configuración.
* Administración del cambio.
* Administración de proyectos y ambiente.

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

| RECURSO DE APRENDIZAJE GAMA MEDIA N. 3: LÍNEA DE TIEMPO- INTERACTIVA  DI\_CF6\_2.2\_RUP\_DISCIPLINAS |
| --- |

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

* Analistas.
* Desarrolladores.
* Probadores.
* Encargados y otros actores.

A continuación se describen cada uno de los roles:

En los **roles específicos** clasificados en la categoría de analistas se encuentran:

* Analistas de procesos de negocio.
* Diseñadores del negocio.
* Analistas del sistema.
* Especificador de requisitos.
* Diseñadores de interfaces de usuario o similares.

Dentro de los roles específicos clasificados en la categoría de **desarrolladores** se encuentran:

* Arquitectos de *software.*
* Diseñador de bases de datos.
* Desarrollador *backend* y *frontend*.
* Cualquier otro rol relacionado con procesos de codificación o integración de código.

En los roles específicos clasificados en la categoría de **probadores** se encuentran:

* Los diseñadores de pruebas.
* Los implementadores de pruebas.

De igual forma, en los roles específicos clasificados en la categoría de analistas encontramos: analistas de procesos de negocio, diseñadores del negocio, analistas del sistema, especificador de requisitos, diseñadores de interfaces de usuario.

Dentro de los roles específicos clasificados en la **categoría de otros** se encuentran:

* Artistas gráficos.
* Administradores de sistemas.
* Especialista en herramientas.
* *Stakeholders*.
* Cualquier otro rol no especificado anteriormente.

Existen otras metodologías que son categorizadas como marcos de trabajo tradicionales, sin embargo, la idea de este espacio es mostrar las características principales de los más reconocidos en la industria de *software*. Si se desea profundizar en las ventajas y desventajas de estas metodologías y otras, se invita a explorar el apartado de materiales complementarios. A continuación, se detallarán las metodologías enmarcadas dentro de la categoría de marcos de trabajo ágiles.

1. **Marcos de trabajo ágiles**

Los marcos de trabajo ágiles o metodologías ágiles para el desarrollo de *software* nacen como otra opción para abordar proyectos donde no es posible tener un detalle completo de los requerimientos y sus estimaciones en la primera fase del proyecto o donde es necesario hacer procesos de adaptabilidad a lo largo del proceso de desarrollo de *software* (Maida y Pacienzia, 2015).

Las metodologías ágiles proveen un conjunto de pautas y principios que buscan facilitar y priorizar la entrega de producto sobre procesos de documentación exhaustiva, haciéndolos más simples, donde interactúa el cliente final desde las primeras etapas del proyecto. El inicio de las metodologías ágiles nació en el año 2001 a partir del manifiesto ágil de *software* donde se establecen cuatro valores fundamentales (Manifiesto Ágil, 2001):

* Individuos e interacciones sobre procesos y herramientas.
* *Software* funcionando sobre documentación extensiva.
* Colaboración con el cliente sobre negociación contractual.
* Respuesta ante el cambio sobre seguir un plan.

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”.

A continuación, se invita a profundizar en el tema revisando la sección de material complementario donde encontrará un video que le permitirá ampliar el conocimiento, por otro lado, en los numerales siguientes podrá identificar otra serie de factores importantes que debe tener en cuenta.

**3.1 Programación Extrema - XP**

XP es la abreviación comúnmente utilizada para referirse a *Extreme Programming,* que es un marco de desarrollo de *software* ágil que busca producir *software* de alta calidad en contextos con requisitos altamente cambiantes, riesgos que involucran tiempos fijo con tecnologías nuevas y equipos de trabajo pequeños ubicados en un mismo sitio.

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

* **Comunicación**: el desarrollo de *software* requiere de trabajo en equipo por lo cual es importante la transferencia de conocimientos y utilizar medios de comunicación apropiados, se propone la discusión cara a cara con herramientas que permitan dibujar o escribir, como, por ejemplo: un tablero.
* **Simplicidad:** esto se refiere a hacer solo las cosas que sean absolutamente necesarias evitando el desperdicio, elaborar las cosas de forma que sea fácil entender por otros y abordar solo requisitos conocidos.
* **Retroalimentación**: esto permite identificar áreas de mejora y revisión constante de las prácticas implementadas en el proceso de forma que se puedan establecer procesos de mejora permanentes.
* **Coraje**: se refiere al actuar sobre situaciones que pueden ser retadoras para el equipo, como, por ejemplo:
  + Enfrentar problemas organizacionales,
  + Intentar implementar funcionalidades de formas diferentes cuando lo convencional no funciona y aceptar comentarios, etc.
* **Respeto**, los miembros del equipo deben respetarse entre sí para comunicarse y trabajar en equipo. Cada persona contribuye en favor de lograr el objetivo del equipo.

Adicional a los valores XP se caracteriza por la definición de un conjunto de 12 prácticas de desarrollo de *software* que aunque pueden ser adoptadas de forma aislada tiene mayor relevancia cuando son desarrolladas en conjunto (Jeffries, 2011):

| RECURSO DE APRENDIZAJE GAMA MEDIA N4. INFOGRAFÍA INTERACTIVA.  DI\_CF6\_3.1\_XP |
| --- |

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

* Los clientes son los encargados del establecimiento de las prioridades del proyecto y las necesidades puntuales.
* Los programadores se encargan de transformar esos requerimientos en bloques de código funcional, son el centro del marco de Extreme Programming*.*
* Los *testers* se encargan de la aplicación de pruebas para garantizar la calidad de los productos y servicios desarrollados.
* El *coach* es una figura encargada de brindar asesoría a los miembros del equipo y son los que definen el rumbo del proyecto.
* El manager se encarga de la coordinación de actividades, del aseguramiento de los recursos requeridos para el proyecto y quien tiene la responsabilidad de la comunicación externa.

**3.2 Desarrollo rápido de aplicaciones - RAD**

RAD (Desarrollo Rápido de Aplicaciones) es otra metodología de desarrollo de *software* ágil que se centra en el desarrollo rápido de aplicaciones mediante la realización de iteraciones frecuentes y realimentación constante y fue inventado por James Martin en 1991.

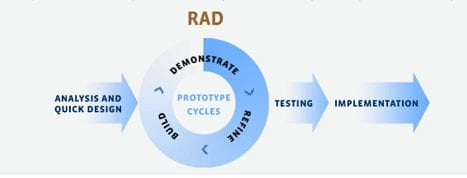
Algunas características fundamentales de RAD son:

* Mayor flexibilidad y adaptabilidad a cualquier ajuste que deba realizarse durante el proceso de desarrollo.
* Iteraciones rápidas que reducen el tiempo de desarrollo y mantienen un ritmo de entrega acelerado.
* Se fomenta la reutilización de código.
* 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 3**

*Fases definidas en RAD*



Nota. Tomada de Blog Capterra (2019).

La **primera fase** definida corresponde a la **definición y finalización** de los requisitos del proyecto. En esta fase las partes interesadas se reúnen para definir los requisitos del proyecto definiendo entre otros casos:

* Los objetivos.
* Las expectativas.
* Los plazos.
* Presupuesto del proyecto.

La **segunda fase** aborda la construcción de **prototipos** los cuales son construidos, validados y mejorados a partir de la validación con los usuarios y una vez son aprobados pasan a la **tercera fase** donde estos prototipos son transformados en modelos funcionales.

En la **cuarta fase** se enfoca en la realización de **pruebas** exhaustivas para garantizar que todos los elementos construidos funcionan bien individualmente y también de forma colectiva. Finalmente, en la **última fase** se realizan todas las actividades de lanzamiento del producto lo que involucra el cargue inicial de datos y entrenamiento a los usuarios.

Según HKSAR (2009) los roles más importantes de la metodología RAD son: facilitador, escriba, equipo *Swat*, administrador del modelo, administrador de bases de datos, equipo de planificación de *Workshop*, equipo de diseño de usuario, equipo de soporte de construcción y equipo de transición, a continuación se mencionan las características más importantes de cada rol.

* **Facilitador:** se encarga del aseguramiento de los objetivos, es responsable de los *Workshops* de captura de requerimientos, se asegura del alistamiento de materiales, es un mediador en procesos de resolución de conflictos.
* **Escriba:** es responsable de la documentación de todas las salidas producto de los *Workshops* de captura de requerimientos y registra información de todo el proceso.
* **Equipo *Swat***: equipo encargado del diseño y construcción del sistema.
* **Administrador del modelo**: encargado de coordinar el desarrollo y mantenimiento de las arquitecturas modelos resultantes del proyecto.
* **Administrador de bases de datos**: responsables del rendimiento, integridad y seguridad de los datos de la organización.
* **Equipo de planificación de los *Workshops***: asisten en la definición de requerimientos y el alcance propuesto para el sistema propuesto.
* **Equipo de diseño de usuario:** proveen información detallada describiendo las funciones del negocio y procesos que serán afectados por el sistema propuesto.
* **Equipo de soporte de construcción**: equipo que se asegura que las necesidades del usuario final sean alcanzadas en el sistema completo mediante la entrega de conocimiento detallado de actividades de negocio y realimentación en la usabilidad del sistema al equipo *Swat*.
* **Equipo de transición:** desarrolla las tareas necesarias para preparar y llevar el sistema a entornos de producción.

**3.3 Scrum**

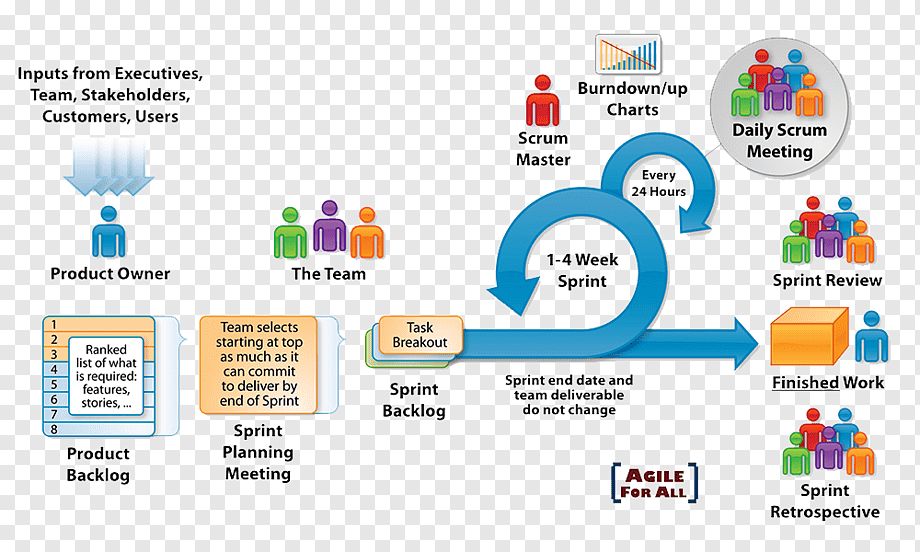
Scrum es un marco de trabajo ágil de muy amplio uso en la industria del *software* que se fundamenta en los valores y principios ágiles definidos en (Manifiesto Ágil, 2001) y donde se definen tres pilares fundamentales según (SCRUMstudy, 2013) los cuales se describen a continuación:

* **Transparencia**: hace referencia a que cualquier proceso de Scrum puede ser conocido por cualquiera. Esto es posible por medio de eventos como:
  + Las reuniones de revisión y reuniones diarias.
  + Artefactos como la pila de producto.
  + Cronogramas de lanzamiento.
  + Documentos de visión del proyecto.
  + Instrumentos de seguimiento, como: el *burndown* chart o el tablero de Scrum *(Scrum board).*
* **Inspección:** permite que cualquiera pueda estar enterado de las actividades realizadas por otros y en general conocer el estado actual de los procesos.
* **Adaptación:** por medio de la transparencia y la inspección es posible fijar actividades de mejoras que permitan modificar todo tipo de proceso en pro de lograr más altos estándares de calidad.

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

**Figura 4**

*Marco de trabajo Scrum*



Nota. Tomada de pngwing.com (s.f.)

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

* + Los roles centrales que hacen referencia a los requeridos obligatoriamente para la creación de un producto, están altamente comprometidos y de los cuales depende el éxito o no de un proyecto.
  + Los roles no centrales que se refieren a todo el personal interesado en el proyecto, pueden interactuar con el equipo, pero no son los responsables del éxito del mismo, dentro de esta categoría entran los *stakeholders*, directivos, gerentes, *marketing*, asesores, etc.

Para representar mejor estas dos categorías se hace referencia a la historia del cerdo y la gallina que se observa en la siguiente figura, los roles centrales son el cerdo en la historia y las gallinas los roles no centrales.

**Figura 5**

*Fábula roles de Scrum*

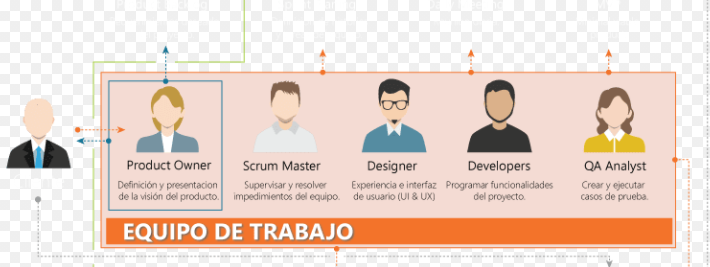


Nota. Tomada de implementingscrum.com (2006).

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

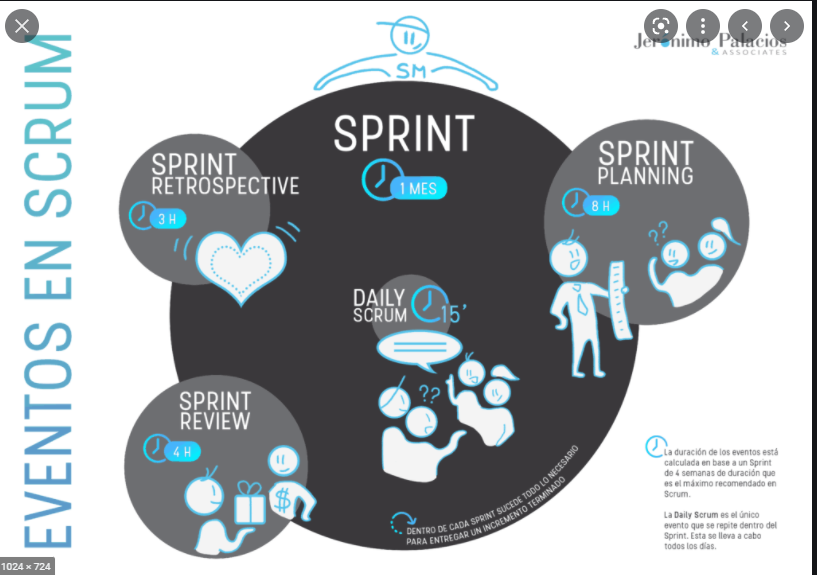
**Figura 6**

*Roles de Scrum*



* **Dueño del producto (*Product Owner*):** persona con amplio conocimiento en el negocio del cliente, sus necesidades y las tendencias del mercado para el área específica. Este rol está encargado de maximizar el valor de negocio entregado al cliente y es el único responsable del control del *Product* *Backlog* (requerimientos) y su priorización. Este también representa al cliente en algunos procesos de demostración de avances y determina cuándo aprobar o no una entrega.
* ***Scrum Master***: es un rol que se encarga de facilitar los procesos al interior del equipo de trabajo removiendo cualquier impedimento y apoyando procesos de empoderamiento personal, debe velar porque los elementos propios del marco de trabajo Scrum se apliquen de manera correcta.
* **Equipo de desarrollo (*Developer Team*):** son los responsables de la transformación de los requerimientos en código ejecutable a ser usado por el cliente, pero también son responsables de la planificación de las iteraciones, establecimiento de características para tener en cuenta en la verificación de un requerimiento terminado y presentación de avances a los clientes. Generalmente es un equipo autoorganizado y autogestionado.

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:



<https://jeronimopalacios.com/scrum/>

* ***Sprint***: es el corazón de Scrum y se refiere a una iteración que está acotada generalmente por un lapso entre 2 y 4 semanas donde se realiza un ciclo completo de actividades de análisis, diseño, construcción y pruebas para desarrollar una versión del producto potencialmente entregable al cliente.
* **Planeación del *Sprint*:** reunión realizada justo antes del inicio de un *Sprint* donde se definen el subconjunto de requerimientos (*Sprint Backlog*) a ser desarrollados en los siguientes y cómo será el proceso requerido para hacer la entrega, lo cual incluye detallar los requerimientos en tareas concretas, estimación de tiempos/esfuerzo y distribución inicial de responsabilidades. Dependiendo de la duración del *Sprint* este tiempo de planificación puede variar, pero la métrica establecida para un *Sprint* de 4 semanas corresponde a una planeación de *Sprint* de 8 horas.
* **Reunión diaria (*Daily Meeting***): reunión realizada generalmente al inicio de cada día donde el equipo de trabajo informa en que ha venido trabajando, qué cosas realizará en el día y qué problemas se le han presentado. Esta es corta, se realiza de pie y debe tener una duración alrededor de los 15 minutos, por lo que, se alinea con los pilares de transparencia e inspección.
* **Revisión del *Sprint*:** reunión realizada al finalizar el sprint donde el equipo de desarrollo muestra los resultados. Para un *Sprint* de 4 semanas se usan reuniones de revisiones 4 horas.
* **Reunión de retrospectiva:** esta es la última, se realiza luego de la revisión del *Sprint* y tiene como objetivo la autoevaluación personal y del grupo sobre el desempeño del *Sprint* que acaba de finalizar. En esta se identifican y documentan los aprendizajes por medio de diferentes técnicas en las que generalmente se busca dar respuesta a las siguientes preguntas:
  + ¿Qué funcionó bien y se debe seguir haciendo?
  + ¿Qué no funcionó bien y se debe dejar de realizar?
  + ¿Qué debemos empezar a mejorar?

Para un *Sprint* de 4 semanas se utilizan 3 horas para esta reunión.

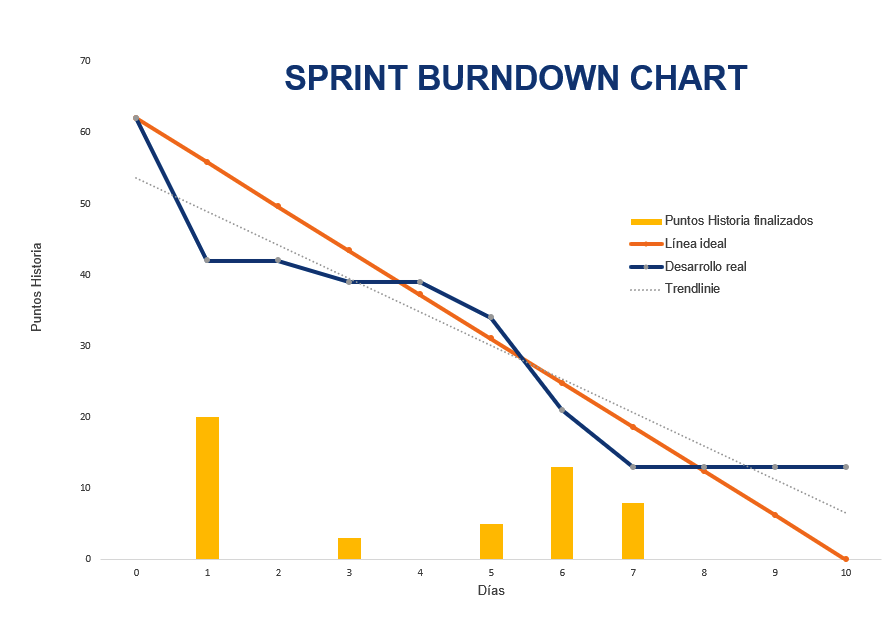
Finalmente, el marco de trabajo Scrum define un conjunto de **artefactos** que permiten registrar y gestionar información clave para asegurar los tres pilares fundamentales y proveen información valiosa durante todo el proceso de desarrollo de *software*. Entre los artefactos representativos de este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

* **Pila de producto (*Product Backlog***): lista priorizada de requerimientos generalmente descritos en formato de historias de usuarios que representa todas las características del sistema a construir.
* **Pila del *Sprint* (*Sprint Backlog***): lista de requerimientos seleccionados desde el *Product backlog* por el equipo de trabajo para ser desarrollados durante un *Sprint* particular. Este es uno de los artefactos generados en la reunión de planeación del *Sprint.*
* ***Burndown Chart***: es un gráfico visual de dos ejes que muestra a los equipos la cantidad de trabajo pendiente por completar (eje Y) y el tiempo disponible para hacerlo (eje X). Este generalmente se realiza por cada Sprint ubicando la cantidad de trabajo a realizar del *Sprint Backlog* (usualmente medido por puntos de historia u de horas de trabajo) en un tiempo 0 y por cada día finalizado se resta la cantidad de puntos de historia u horas de cada tarea completada.

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 la cantidad de *Sprint* proyectados. Cada uno de estos puntos se une por medio de una línea y es posible determinar visualmente si el flujo de trabajo está en una situación óptima o no respecto al tiempo restante para completar el *Sprint*.

**Figura 7**

*Burndown Chart*

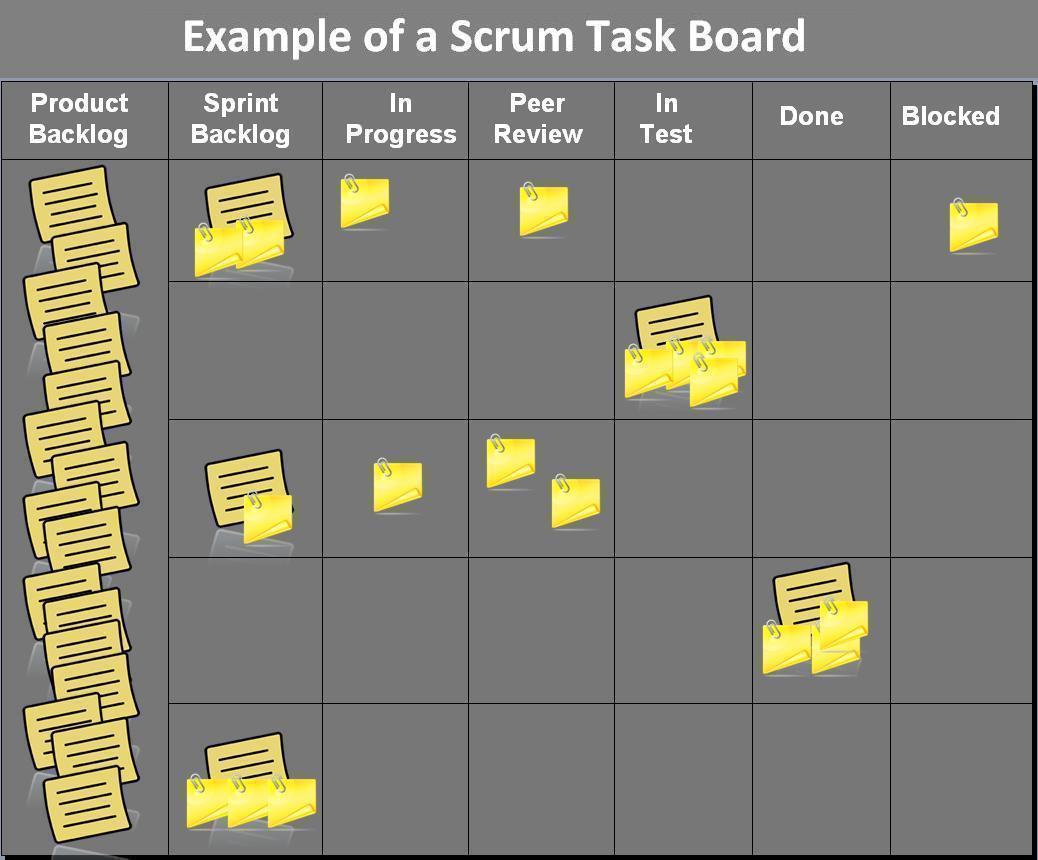


Nota. Tomada de intl-blog.imgix.net (2019).

* **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 8**

*Tablero de Scrum*



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

* 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.
* 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.
* El proyecto puede iniciar con requerimientos de muy alto nivel y es fácil administrar los cambios.
* La participación constante del cliente permite mitigar riesgos del proyecto desde sus primeras etapas.
* 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.

Para profundizar le invito a revisar la sección de material complementario donde encontrará un video sobre el tema.

1. **Planeación de proyectos de *software***

La planificación de proyectos de *software* involucra un conjunto de actividades que buscan tener una imagen preliminar del *software* a construir a partir de la cual se realizan estimaciones que permiten determinar la viabilidad o no de un proyecto de acuerdo con la cantidad de recursos asignados.

Según (SCRUMstudy, 2013), las actividades de planificación definidas dentro del marco de trabajo de Scrum son las que a continuación se mencionan en el recurso de aprendizaje:

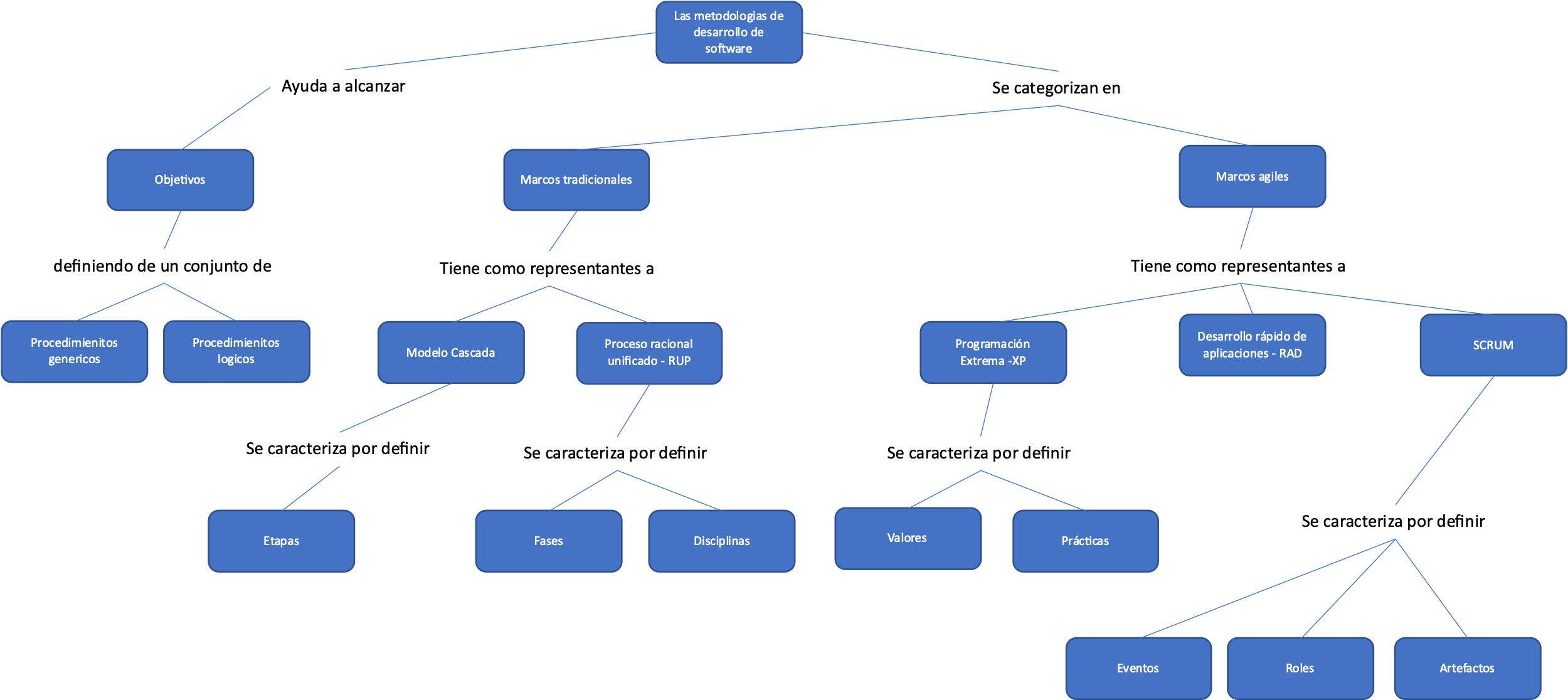
| Recurso de aprendizaje gama alta n. 2 video  DI\_CF6\_4\_PlaneaciónProyectos |
| --- |

1. **Síntesis**

Las metodologías de desarrollo de *software* definen un conjunto de elementos que facilita el proceso de gestión y construcción ordenada de un producto o servicio de *software*, pero dependiendo de sus particularidades puede ser categorizado como un marco de trabajo tradicional o marco de trabajo ágil.

Cada marco de trabajo define un conjunto de características particulares y es el equipo de trabajo el que al final decide, de acuerdo con los contextos particulares de cada proyecto, cuál se acomoda mejor a sus propias necesidades. Incluso es posible ver la aplicación de características de diferentes metodologías en un mismo proyecto, lo que generaría un marco de trabajo híbrido.

A continuación, se muestra un mapa conceptual de los elementos más importantes desarrollados en este componente:



1. **ACTIVIDADES DIDÁCTICAS**

| DESCRIPCIÓN DE ACTIVIDAD DIDÁCTICA 1 | |
| --- | --- |
| Nombre de la Actividad | Asociación de conceptos en metodologías de desarrollo de *software* |
| Objetivo de la actividad | Afianzar los conceptos vistos en el componente formativo más importantes y que están asociados a las metodologías de desarrollo de *software*. |
| Tipo de actividad sugerida | Arrastrar y soltar el término de la columna A con la definición que corresponde que se encuentra en la columna B. |
| Archivo de la actividad  (Anexo donde se describe la actividad propuesta) | Anexo documento en Word llamado Actividad didáctica 1 |

| DESCRIPCIÓN DE ACTIVIDAD DIDÁCTICA 2 | |
| --- | --- |
| Nombre de la Actividad | Características de Scrum |
| Objetivo de la actividad | Afianzar algunos de los conceptos más importantes asociados al marco de trabajo Ágil – Scrum. |
| Tipo de actividad sugerida | Arrastrar y soltar el término con la definición que corresponde. |
| Archivo de la actividad  (Anexo donde se describe la actividad propuesta) | Anexo documento en Word llamado Actividad didáctica 2 |

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 |
| --- | --- | --- | --- |
| 2. Marcos de trabajo tradicionales | Henao, C. (2018). *#1. ¿Qué son las metodologías tradicionales en el desarrollo de software?* [Video]. YouTube. [https://www.youtube.com/watch?v=i8CPD1dW88kyfeature=youtu.be](https://www.youtube.com/watch?v=i8CPD1dW88k&feature=youtu.be) | Video de YouTube | <https://youtu.be/i8CPD1dW88k> |
| 3. Marcos de trabajo Ágiles | Henao, C. (2018b). *#2. ¿Qué son las metodologías ágiles en el desarrollo de Software?* [Video]. YouTube. [https://www.youtube.com/watch?v=fHKsufzM7qQyfeature=youtu.be](https://www.youtube.com/watch?v=fHKsufzM7qQ&feature=youtu.be) | Video de YouTube | <https://youtu.be/fHKsufzM7qQ> |
| 3.3 SCRUM | Henao, C. (2018d). #3*. SCRUM en 6 minutos | Metodologías Ágiles*[Video]. YouTube. [https://www.youtube.com/watch?v=HhC75IonpOUyfeature=youtu.be](https://www.youtube.com/watch?v=HhC75IonpOU&feature=youtu.be). | Video de YouTube | [https://www.youtube.com/watch?v=HhC75IonpOUyfeature=youtu.be](https://www.youtube.com/watch?v=HhC75IonpOU&feature=youtu.be) |

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.

Blog Capterra.com. (2019). *RAD*. <https://blog.capterra.com/wp-content/uploads/2019/12/K_rOSJJEJ-Zl0G3VZGYWiGw3tFj1TuRQDvqPTpmVB3FrK4lnOONPG06gqWdwjBOagIbRUyCyD5R_WhvzZ4cHdglqaVyGyV4pWcn-PwIJn8JniGlQaeAQBJqLxqH_t4i7uCnBW9rp.png>

Implementingscrum.com (2006). *Scrum*. Fuente: [https://cdn-images-1.medium.com/max/1600/1\*znawlwBQPM\_xAcopRXjQHw@2x.jpeg](https://cdn-images-1.medium.com/max/1600/1*znawlwBQPM_xAcopRXjQHw@2x.jpeg)

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. <https://www.researchgate.net/profile/Mustafa-Man/publication/228820202/figure/fig6/AS:668418772578316@1536374862199/Phases-in-Rational-Unified-Process.png>

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 | Octubre de 2021 |
| Ana Vela Rodríguez Velásquez | Diseñadora Instruccional | Centro de Gestión Industrial  Regional Bogotá Capital | Octubre de 2021 |
| Silvia Milena Sequeda Cárdenas | Evaluadora Instruccional | Centro de Gestión Industrial  Regional Bogotá Capital | Octubre de 2021 |
| Rafael Neftalí Lizcano Reyes | Asesor pedagógico | Centro de Diseño y la manufactura  Regional Santander | Octubre de 2021 |
|  | José Gabriel Ortiz Abella | Corrector de estilo | Regional Distrito Capital – Centro para la Industria de la Comunicación Gráfica | Octubre del 2021. |

1. **CONTROL DE CAMBIOS**

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

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