Universidad de la sierra sur

Metodologías de un proyecto

Informática 506-A

Índice

1. Modelo en cascada **4**
- 1.1. Modelos evolutivos. 5
  - 1.1.1. Hacer prototipos. 5
  - 1.1.2. El modelo espiral. 6
  - 1.1.3. Modelos concurrentes. 7

2. SCRUM **8**
- 2.1. Principales características de Scrum. 9
- 2.2. El ciclo de Scrum. 9
- 2.3. Ventajas de Scrum. 10
- 2.4. desventajas de Scrum. 11
- 2.5. Valores Ágiles. 12
- 2.6. Conclusión. 12

3. KANBAN **12**
- 3.1. Qué es el método Kanban y para qué sirve. 12
- 3.2. Definición de KANBAN. 13
- 3.3. Características del Método Kanban. 13
  - 3.3.1. PRINCIPIO 1: Visualización en el Modelo Kanban. 13
  - 3.3.2. PRINCIPIO 2: Tareas en proceso. 14
  - 3.3.3. PRINCIPIO 3: Priorización de tareas en método KANBAN . 14
  - 3.3.4. PRINCIPIO 4: Medir el tiempo 15
- 3.4. Ventajas del método kanban – beneficios. 15
- 3.5. Desventajas e inconvenientes de la metodología kanban. 15

4. Metodología ágil crystal. **16**
- 4.1. Características 16
- 4.2. Ventajas 16
- 4.3. Desventajas 17

Índice de ilustraciones

Ilustración 1 Modelo en cascada....4
Ilustración 2 Modelo en V 4
Ilustración 3 Hacer prototipos 6
Ilustración 4 Modelo en espiral 7
Ilustración 5 Modelo concurrente....8
Ilustración 6 Ciclo Scrum 10
Ilustración 7 Grafica de ventajas....17
Ilustración 8 Grafica de desventajas....17

1. Modelo en cascada

Ocurre en cierto número limitado de nuevos esfuerzos de desarrollo, sólo cuando los requerimientos están bien definidos y tienen una estabilidad razonable.

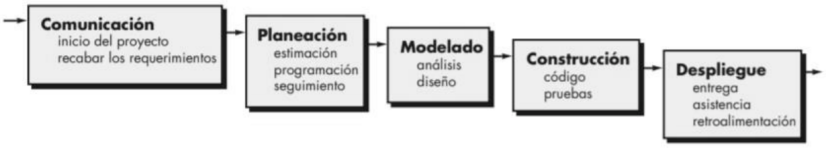

Ilustración 1 Modelo en cascada

Modelo de la cascada, o también llamado ciclo de vida clásico Sugiere un enfoque sistemático y secuencial para el desarrollo del software, que comienza con la especificación de los requerimientos por parte del cliente y avanza a través de planeación, modelado, construcción y despliegue.

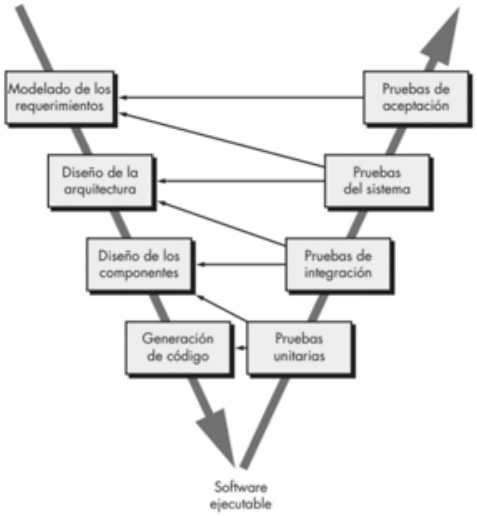

Ilustración 2 Modelo en V

A medida que el equipo de software avanza hacia abajo desde el lado izquierdo de la V, los requerimientos básicos del problema mejoran hacia representaciones técnicas cada vez más detalladas del problema y de su solución. Una vez que se ha generado el código, el equipo sube por el lado derecho de la V, y en esencia ejecuta una serie de pruebas (acciones para asegurar la calidad) que validan cada uno de los modelos creados cuando el equipo fue hacia abajo por el lado izquierdo. El modelo de la cascada es

el paradigma más antiguo de la ingeniería de software. Problemas que en ocasiones surgen al aplicar el modelo de la cascada:

- Los proyectos reales no siempre siguen el flujo secuencial propuesto por el modelo.
- Es difícil para el cliente enunciar en forma explícita todos los requerimientos.
- El cliente debe tener paciencia.

1.1. Modelos evolutivos.

Los modelos evolutivos son iterativos, Se caracterizan por la manera en la que permiten desarrollar versiones cada vez más completas del software, los dos modelos comunes de proceso evolutivo son los siguientes:

1.1.1. Hacer prototipos.

El paradigma de hacer prototipos ofrece un mejor enfoque, ayudara a los equipos a mejorar la comprensión de que se va a elaborar cuando los requerimientos no estén claros.
El paradigma de hacer prototipos tiene un proceso, el cual comienza con la comunicación, donde se reúne los participantes para definir los objetivo generales de software, se identifican los requerimientos que se conozca y detecta las áreas en las que es imprescindible una mayor definición, después de esto se plantea rápidamente una iteración para hacer el prototipo, y se lleva a cabo el modelado(diseño rápido), este se centra en la representación de los aspectos del software que serán visibles para los usuarios finales. El diseño rápido lleva a la construcción de un prototipo donde se entrega y es evaluado por los participantes, que dan retroalimentación para mejorar los requerimientos. Lo ideal es que el prototipo sirva como mecanismo para identificar los requerimientos del software, el prototipo sirve como "el primer sistema", algunos prototipos se construyen para ser "desechables", otros son evolutivos ya que poco a poco se transforman en el sistema real, a los ingenieros de software les gusta el paradigma de hacer prototipos porque ellos siente que así los usuarios adquieren la sensación del sistema real, y los desarrolladores logran construir algo de inmediato.
Algunas razones por las cual el hacer prototipos llega a ser algo problemático son las siguientes:

1. Los participantes ven lo que parece ser una versión funcional del software, sin darse cuenta de que el prototipo se obtuvo de manera caprichosa; no perciben que en la prisa por hacer que funcionara, usted no consideró la calidad general del software o la facilidad de darle mantenimiento a largo plazo.

2. Es frecuente que llegue a compromisos respecto de la implementación a fin de hacer que el prototipo funcione rápido. Quizá utilice un sistema operativo inapropiado.

Hacer prototipos es un paradigma eficaz para la ingeniería de software. La clave es definir desde el principio las reglas del juego, es decir, todos los participantes deben estar de acuerdo en que el prototipo sirva como el mecanismo para definir los requerimientos esto hará que la ingeniería del software real con la mirada puesta en la calidad.

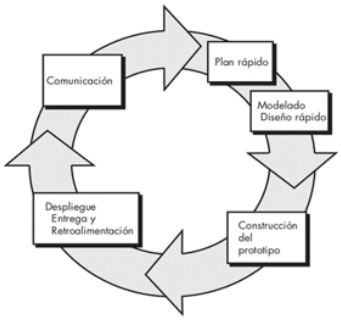

Ilustración 3 Hacer prototipos

1.1.2. El modelo espiral.

El modelo espiral es un modelo evolutivo del proceso del software y se acopla con la naturaleza iterativa de hacer prototipos con los aspectos controlados y sistémicos del modelo de cascada. Un modelo en espiral es dividido por el equipo de software en un conjunto de actividades estructurales.
En el proceso evolutivo el equipo de software realiza actividades implícitas en un circuito alrededor de la espiral en el sentido horario, partiendo del centro.
El primer circuito alrededor de la espiral da como resultado el desarrollo de una especificación del producto; las vueltas sucesivas se usan para desarrollar un prototipo y, luego, versiones cada vez más sofisticadas del software, cada paso por la región de planeación da como resultado ajustes en el plan del proyecto, el costo y la programación de actividades se ajustan con base en la retroalimentación obtenida del cliente después de la entrega, con esto el el gerente del proyecto ajusta el número planeado de iteraciones que se requieren para terminar el software.
El modelo espiral es un enfoque realista para el desarrollo de sistemas y de software a gran escala, el software evoluciona a medida que el proceso avanza, el desarrollador y cliente comprenden y reaccionan mejor ante los riesgos en cada nivel de evolución. El modelo espiral usa los prototipos como mecanismo de reducción de riesgos, permite aplicar el enfoque de hacer

prototipos en cualquier etapa de la evolución del producto.

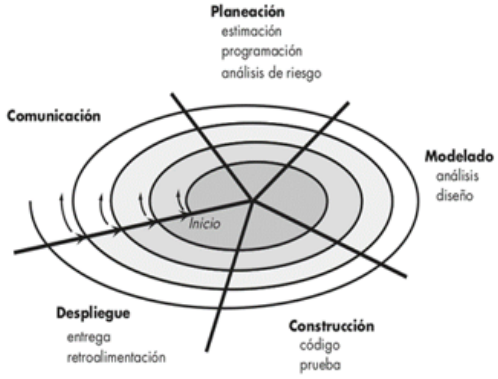

Ilustración 4 Modelo en espiral

1.1.3. Modelos concurrentes.

El modelo de desarrollo concurrente, en ocasiones llamado ingeniería concurrente, permite que un equipo de software represente elementos iterativos y concurrentes de cualquiera de los modelos de proceso antes mencionados. Por ejemplo, la actividad de modelado definida para el modelo espiral se logra por medio de invocar una o más de las siguientes acciones de software: hacer prototipos, análisis y diseño.
La figura muestra la representación esquemática de una actividad de ingeniería de software dentro de la actividad de modelado con el uso del enfoque de modelado concurrente. Cabe destacar que todas las actividades de ingeniería de software existen de manera concurrente, pero se hallan en diferentes estados.

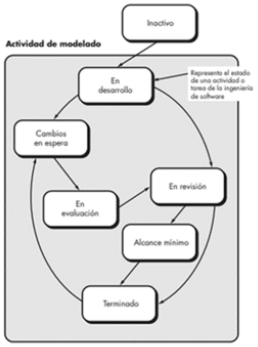

Ilustración 5 Modelo concurrente

El modelado concurrente define una serie de eventos que desencadenan transiciones de un estado a otro para cada una de las actividades, acciones o tareas de la ingeniería de software. Por ejemplo, durante las primeras etapas del diseño (acción importante de la ingeniería de software que ocurre durante la actividad de modelado), no se detecta una inconsistencia en el modelo de requerimientos. Esto genera el evento corrección del modelo de análisis, que disparará la acción de análisis de requerimientos del estado terminado al de cambios en espera.
El modelado concurrente es aplicable a todos los tipos de desarrollo de software y proporciona un panorama apropiado del estado actual del proyecto.

2. SCRUM

Scrum es un proceso en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente, en equipo, y obtener el mejor resultado posible de un proyecto. Estas prácticas se apoyan unas a otras y su selección tiene origen en un estudio de la manera de trabajar de equipos altamente productivos.
En Scrum se realizan entregas parciales y regulares del producto final, priorizadas por el beneficio que aportan al receptor del proyecto. Por ello, Scrum está especialmente indicado para proyectos en entornos complejos, donde se necesita obtener resultados pronto, donde los requisitos son cambiantes o poco definidos, donde la innovación, la competitividad, la flexibilidad y la productividad son fundamentales.

2.1. Principales características de Scrum.

Las características de Scrum listadas abajo pueden interpretarse y aplicarse sin problema en cualquier industria.

1. El método apoya la colaboración y la autoorganización. Los miembros del equipo trabajan y desarrollan el proyecto en conjunto, compartiendo ideas y descubrimientos.

2. Los equipos Scrum autogestionan sus actividades, y todas se enmarcan dentro de un margen de tiempo específico.

3. En Scrum, el cliente recibe versiones funcionales del producto final continuamente a lo largo de ciclos incrementales (sprints) que transcurren semanalmente o, como máximo, de mes en mes.

4. Los sprints se repiten hasta que el equipo Scrum consigue desarrollar todas las características solicitadas para el proyecto.

5. En Scrum, la adaptación a las condiciones del mercado tiene especial importancia. Para lograrlo, se incorporan cambios durante el ciclo de desarrollo del producto e incluso más tarde.

6. Scrum se enfoca en brindar respuestas rápidas y eficientes a los cambios, diseñando soluciones creativas y funcionales en el menor tiempo posible.

7. Uno de los principales objetivos del método es ofrecer altos valores comerciales al cliente en poco tiempo.

8. Una vez al día, se efectúa una reunión de seguimiento que no debe durar más de 15 minutos. La finalidad es obtener feedback sobre el avance del equipo y los posibles obstáculos.

9. Gracias a la participación que tienen todas las partes involucradas en el proyecto, con Scrum se puede ahorrar dinero, mitigar riesgos y ganar terreno en el mercado rápidamente.

10. El tamaño del equipo Scrum es un factor decisivo. Un equipo pequeño puede no tener las habilidades necesarias para desarrollar el producto o servicio solicitado, y un equipo grande puede arruinar el trabajo, ya que la colaboración será difícil. Por eso, el tamaño óptimo suele ser de seis a diez personas.

2.2. El ciclo de Scrum.

1. El Product Owner crea un Product Backlog, de acuerdo con el cliente. Básicamente una lista de requerimientos a completar.

2. El Equipo realiza un Sprint Planning Meeting. El proyecto se divide en tareas y se reparten.

3. Se crea un Sprint Backlog, donde se definen los Springs con objetivos detallados.

4. Se define un tiempo asignado a cada Spring, normalmente entre una y cuatro semanas.

5. El equipo se reune a diario en un Daily Standup, para que cada miembro detalle su progreso y que el Scrum Master y al resto del equipo estén informados.

6. Al final de cada Spring se llevan a cabo una Revisión y una retrospectiva del Spring.

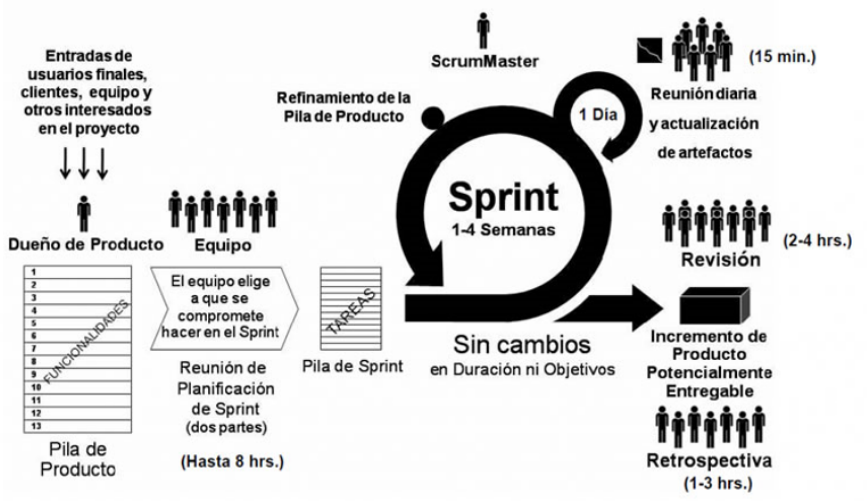

Ilustración 6 Ciclo Scrum

2.3. Ventajas de Scrum.

Scrum está muy de moda en los últimos tiempos, a pesar de ser una metodología que lleva años existiendo.
A nivel de resultados, Scrum funciona si está correctamente aplicado, y a nivel desde dentro de los equipos también es muy interesante, ya que tiene muchas ventajas, de las que destacamos algunas a continuación:

- Los equipos están muy motivados, porque las personas están contentas y felices.
- Su trabajo luce y es un producto de calidad.
- Existen muchas reuniones de planificación, de refinamiento, para estimar, etc.

- Se tienen en cuenta las opiniones de los trabajadores.
- Es un proceso en el que da tiempo a adaptar, reflexionar e inspeccionar y adaptar o cambiar las cosas que no han ido bien, mantener las cosas que han ido bien y abandonar las cosas que no hayan gustado.
- Se pueden obtener resultados anticipados.
- Se pueden regular las expectativas.
- Se tiene un alto grado de transparencia, ya que el cliente sabe en todo momento en que está trabajando en equipo, cuánto está tardando y los resultados que se tienen en cada momento, es decir, cada vez que se termina un Sprint el resultado puede ser lanzado.

2.4. desventajas de Scrum.

No hay ninguna solución que pueda satisfacer a todo el mundo. Por eso existen numerosos marcos de trabajo y metodologías de desarrollo de productos, generalistas o específicas. En el caso de Scrum, las desventajas más notables podrían ser:

- Puede dar la impresión de que los proyectos basados en Scrum tienden a durar más tiempo, al no tener una fecha de finalización definida. Esto es discutible, porque a menudo no se está comparando el resultado final obtenido con cada metodología.
- Se dice que las posibilidades de fracasar son mayores que en otras metodologías. Principalmente porque se apoya en el trabajo de pocos miembros, muy cualificados, la salida del proyecto de uno de ellos supone un serio contratiempo. Si la cooperación y motivación son esenciales en cualquier proyecto, en Scrum más.
- Por la misma razón Scrum solo funciona si contamos con profesionales muy experimentados, los juniors necesitan referencias, veteranos que les ayuden a desarrollarse.
- No es posible aplicar Scrum con grandes equipos, digan lo que digan no funciona. Hace falta una adaptación del framework o usarlo en combinación con otras metodologías.
- Las reuniones diarias pueden crear estrés y frustración entre los miembros menos experimentados. Hay que ayudarles a aprovecharlas para su crecimiento personal y profesional. Esta es una de las tareas fundamentales del Scrum Master.
- Lograr estándares altos de calidad solo es posible con una gran disciplina. El seguimiento de las tareas realizadas y la implementación de métodos de prueba estrictos son obligatorios. Algo que, por otra parte, debería ser aplicable a cualquier otra metodología.

2.5. Valores Ágiles.

Los valores ágiles que tiene Scrum son los siguientes:

1. Valorar a los individuos y las iteraciones sobre el proceso y las herramientas: La prioridad son las personas, ya que las personas no son recursos, un concepto que a veces se olvida.

2. El software funcionando por encima de la documentación extensiva: Se sustituye una documentación extensa y de lectura abstracta, por una visión más dinámica, por una serie de reuniones que se van adaptando a las necesidades del negocio.

3. Valorar más la colaboración con el cliente que la negociación de un contrato: Cambia una negociación más burocrática por una colaboración entre los miembros del equipo e incluso con el cliente.

4. Valorar más la respuesta ante el cambio que seguir un plan: Cada Sprint es un reto, y resulta muy importante ver como el equipo reacciona, como ha sabido buscar soluciones y como se ha adaptado al tiempo límite, más que seguir un plan con unas pautas marcadas, muy rígidas y con muy poca flexibilidad.

2.6. Conclusión.

Se llame Scrum al marco, Scrum Master al responsable del proyecto, o de cualquier otra manera, lo cierto es que nos sitúa ante una problemática referente al desarrollo del software que ninguna metodología puede resolver fácilmente: personas, profesionales, que hacen uso de tecnologías en constante evolución, con una enorme presión, bajo demandas que generan mucho estrés.
Un buen Scrum Master deberá resolver problemas muy importantes. Pero los directivos de las organizaciones que viven del desarrollo de software o de su uso, deberán encontrar el equilibrio entre los objetivos empresariales y de negocio. Contar con profesionales tan competentes como productivos, es tan importante como conseguir retener el talento. Scrum puede ayudarles a que trabajen de una forma más dinámica, pero también más participativa, que les puede motivar para crecer profesionalmente.

3. KANBAN

3.1. Qué es el método Kanban y para qué sirve.

El método Kanban, al igual que el método Scrum, es un tipo de sistema basado en la metodología ágil. En el caso del método Kanban, se busca conseguir un proceso productivo, organizado y eficiente a la hora de poder llevar a cabo las diferentes tareas en un departamento. Puede tratarse de

un proyecto en concreto o bien de un departamento interno en sí. Fomenta el trabajo en equipo, funcional y con un flujo permanente de tareas.

3.2. Definición de KANBAN.

Kanban es una palabra japonesa, que significa panel / Cartel. La idea de este sistema de organización es clasificar las tareas en 4 apartados; Por Hacer, En proceso, Pendiente de revisión, Hecho. Dentro de estos apartados, cada miembro clasifica sus tarjetas en función de las prioridades acordadas.

3.3. Características del Método Kanban.

Para poder agilizar las tareas, Kanban cuenta con cuatro principios o características básicas que lo fundamentan.

3.3.1. PRINCIPIO 1: Visualización en el Modelo Kanban.

Para poder comprender en qué momento del desarrollo se encuentra tu proyecto o revisar los temas tratados, Kanban es totalmente transparente, de forma que tienes acceso a todas las tareas en cualquier momento. Lo cual te permite organizarte en los diferentes bloques (Por Hacer, En proceso, Pendiente de revisión, Hecho) y hacer modificaciones para el buen funcionamiento del equipo.

La Tarjeta Kanban es el elemento clave en el tablero Kanban, cada tarjeta contiene información como el nombre de la tarea, la descripción, la persona asignada, la prioridad, comentarios, enlaces externo así como las subtareas asociadas.
Podemos separar estos elementos de la tarjeta en visibles o detalle en función de si es necesario abrir la tarjeta o no para visualizarlo. De esta forma tendríamos
Partes visibles tarjeta:

- Título Tarea: El nombre que le hemos dado a la tarea siempre será visible en la tarjeta Kanban.
- Descripción Tarea: Aquí introduciríamos los detalles de la tarea. Suele ser visible en todos los modos de visualización.
- Tiempo ciclo: Entendemos como tiempo de ciclo como la duración estimada de la tarea.
- Persona asignada.
- Prioridad.

Estas partes suelen ser vistas solo si pinchas en el detalle de la tarjeta, siendo «invisibles» en el modo tablero Kanban. Su función es dar más información, pero suelen permanecer más escondidas por no ser tan vitales como los anteriores.

- Comentarios miembros equipo.
- Enlaces adjuntos.
- Archivos adjuntos.
- Histórico estados tarjeta.

3.3.2. PRINCIPIO 2: Tareas en proceso.

Dice que no se debería de trabajar con más de una tarjeta a la vez, Durante la realización de las tareas, surgen dudas, o se ven fallos/mejoras. Kanban fomenta la continua modificación de las tareas, ya que se trata de un sistema de trabajo inmediato, compuesto por pequeñas tareas de corta duración. De esta forma ponemos freno a uno de los principales problemas en un entorno de trabajo: la focalización. Centrarse en tareas de corta duración y etiquetarlas según su estado no solo te permite ser más productivo, sino que te permite afrontar el problema de una forma más eficaz que con una tarea más grande en la que tienes la sensación de que no avanzas.

3.3.3. PRINCIPIO 3: Priorización de tareas en método KANBAN

Gracias a este concepto, cuando vas al bloque de las tareas pendientes, ya tienes claro cuál es el siguiente tema a tratar. La trasparencia que te permite ver todos los ejercicios a realizar hace que sea posible una gestión mejor del tiempo y colocar las tareas con un orden coherente para facilitar el trabajo propio y el del equipo.
La mayor parte de herramientas para implementar la metodología Kanban te permiten o bien ordenar con drag and drop las tareas de acuerdo a la prioridad para el equipo o dependencias o incluso asignarles un valor.
Además, una buena práctica, suele ser la incorporación de filas de acuerdo a diferentes prioridades/conceptos dentro de una misma columna. Por ejemplo, podemos crear filas para agrupar tareas como:

- Agrupación tareas por tipología de trabajo
- Agrupaciones en base al estado de desarrollo
- Agrupación por prioridades equipo/dependencias

Otra opción es utilizar una escala de colores para priorizar.

3.3.4. PRINCIPIO 4: Medir el tiempo

Gracias al sistema de situar las tareas en "Haciendo", durante el tiempo que trabajamos en ellas y etiquetar las tarjetas según el tema tratado, podemos hacer un seguimiento del tiempo invertido en cada función, departamento o campo.

3.4. Ventajas del método kanban – beneficios.

- Procesos innecesarios: Debido a la comunicación y las facilidades de las diferentes herramientas Kanban, evitamos procesos que reducen la eficiencia o maximizan el tiempo de trabajo.
- Conciencia sobre el desarrollo; Gracias a la comunicación y al control que se tiene en todo momento sobre el estado el producto, es mucho más sencillo mantener al equipo actualizado y con conocimiento del proceso que les envuelve.
- Equipo motivado; Debido a las características de Kanban y su metodología de trabajo, el equipo se ve más único, y con una mayor facilidad para plantear y resolver dudas al mismo tiempo que van saliendo. Se crea un vínculo mayor que con otros procedimientos y sirve como incentivados para los miembros del equipo.
- Flexibilidad; Ya que vivimos tiempos rápidos, tenemos que saber cómo enfrentarnos a los cambios, y con kanban, podemos hacer variaciones al producto tan pronto como se detecte un error o un giro en el mercado. Estos errores se localizan en breves periodos de tiempo debido a la corta duración de las tareas asignadas. Esto permite ofrecer productos que puedan competir con mercados emergentes y ofrece una flexibilidad de adaptación mucho mayor.
- Más eficaces; Kanban nos permite optimizar los tiempos, lo cual se traduce en eficiencia. Mayor calidad en menor tiempo. Además de las múltiples actualizaciones que podemos hacer durante el proceso evolutivo de un proyecto, es otros de los factores que nos permiten no tener que volver a empezar prácticamente desde cero al detectar errores o querer implementar mejoras.

3.5. Desventajas e inconvenientes de la metodología kanban.

- Costoso en determinados casos. Al ser un sistema basado en tareas si lo tratamos de aplicar en departamentos de producción muy grandes o en equipos multidiciplinares puede ser imposible de aplicar puesto que acabaríamos con tableros difíciles de gestionar.
- Tipo de Proyecto. El proceso Kanban llevado a la gestión de equipos, proyectos o procesos industriales asume que se sigue un ciclo repetitivo (por hacer, en proceso, en revisión, hecho). Si te encuentras ante tareas que no siguen un

patrón o que no es relevante conocer el estado puedo que no te resulte útil seguir esta metodología o flujo de trabajo.

- Anticipación: Aplicando la metodología Kanban resulta complicado anticiparse ante cambios de un proyecto.

4. Metodología ágil crystal.

El método de desarrollo Crystal, es una colección de enfoques de desarrollo de software ágil, se centra principalmente en las personas y la interacción entre ellas mientras trabajan en un proyecto de desarrollo de software. También hay un enfoque en la criticidad comercial y la prioridad comercial del sistema en desarrollo. A diferencia de los métodos de desarrollo tradicionales, Crystal no repara las herramientas y técnicas de desarrollo, sino que mantiene a las personas y los procesos en el centro del proceso de desarrollo. Sin embargo, lo importante no son solo las personas o los procesos, sino la interacción entre los dos.

4.1. Características

Una de sus características principales es la vital importancia que se les da a los desarrolladores que componen el grupo de trabajo, por lo cual sus puntos de estudio están destinados a:

- Aspecto humano del equipo.
- Tamaño de un equipo.
- Comunicación entre los desarrolladores.
- Políticas a seguir.
- Espacio físico de trabajo.

4.2. Ventajas

- Son apropiadas para entornos ligeros
- Al estar diseñada para el cambio experimenta reducción de costo.
- Presenta una planificación más transparente para los clientes.
- Se definen en cada iteración cuales son los objetivos de la siguiente.
- Permite tener una muy útil realimentación de los usuarios.

Ilustración 7 Grafica de ventajas

4.3. Desventajas

- Delimita el alcance del proyecto con el cliente.
- Puede no ser factible para proyectos muy grandes.
- Aún está en desarrollo.

Ilustración 8 Grafica de desventajas

Referencias

Pressman, R. S. (2010). *Ingeniería del software.* New York: y McGraw-Hill.

A. (2021, 22 junio). *¿Cómo Aplicar el Método Kanban? Características y Ventajas*. Apiumhub. https://apiumhub.com/es/tech-blog-barcelona/metodo-kanban-ventajas/

Briones, A. (2019, 24 abril). *Metodología de Desarrollo de Software Crystal*. Informes - Abraham Briones. https://www.clubensayos.com/Tecnolog%C3%ADa/Metodolog%C3%ADa-de-Desarrollo-de-Software-Crystal/4697721.html

G. (2015, 18 septiembre). *ROLES*. METODOLOGÍA AGIL CRYSTAL. https://iswugaps2crystal.wordpress.com/2015/09/17/roles/

Juancarruthers, V. A. P. B. (2018, 10 abril). *Metodología Agil: Crystal*. Ingenieria de Software I. https://isi2018.wordpress.com/2018/04/09/metodologia-agil-crystal/