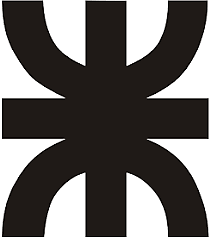
**Universidad tecnológica nacional**

Facultad Regional de Córdoba



Ingeniería en Sistemas de Información

Ingeniería de Software

***Docentes***

ING COVARO, Laura Inés (Adjunto);

ING. MASSANO, María Cecilia (JTP)

ING CRESPO, Mickaela (Ayudante 1ra)

***Integrantes***

ANGLADA, Martín Legajo: 58159 martinanglada@gmail.com;

CHECA, Nicolás Legajo: 58351;

MARRO, Yanina Legajo: 47594 yani\_marro@hotmail.com;

PICOSSI, Matias Legajo: 54941;

TOLEDO, Rodrigo Legajo: 75441.

**Grupo N°**: 7

***Curso:*** 4K3

2020

ÍNDICE

[Introducción 2](#_Toc55496192)

[Práctico 14 - Comparación De Enfoques Para Gestión Del Desarrollo De Productos De Software 3](#_Toc55496193)

[Consigna: 3](#_Toc55496194)

[Objetivo: 3](#_Toc55496195)

[Propósito: 3](#_Toc55496196)

[Entradas: 3](#_Toc55496197)

[Salida: 3](#_Toc55496198)

[Instrucciones: 3](#_Toc55496199)

[Observaciones: 3](#_Toc55496200)

[Comparación de Enfoques para Gestión del Desarrollo de Productos de Software 4](#_Toc55496201)

[Conclusión 9](#_Toc55496202)

[Bibliografía utilizada 10](#_Toc55496203)

INTRODUCCIÓN

En el siguiente informe se presenta el análisis realizado respecto a la evaluación de diferentes enfoques para la gestión del desarrollo de productos de software basados tanto en procesos definidos como en procesos empíricos.

La característica particular que poseen los procesos definidos es que se pueden aplicar una y otra vez y obtener los mismos resultados. Este tipo de proceso es predictivo por lo que es posible administrarlo y controlarlo.

El enfoque de gestión que se utiliza para desarrollar software bajo este proceso es el que denominamos enfoque tradicional, el cual permite realizar las fases de Análisis de requerimientos, Diseño, Desarrollo e Implantación secuencialmente sin iniciar la fase siguiente hasta que la anterior esté terminada.

Los procesos empíricos tienen la particularidad que no son predictivos, es decir si se aplica una y otra vez, los resultados obtenidos nunca son los mismos. En este caso la falta de predictibilidad ocasiona que no exista una administración y control tal como se haría en los procesos definidos, sino que se busca administrar este tipo de proceso a través de inspecciones frecuentes y adaptaciones.

El enfoque de gestión que se utiliza en los procesos empíricos es lo que denominamos enfoque ágil y aquí encontramos dos marcos de trabajo como SCRUM y KANBAN.

En este documento se presentó una comparación respecto a estos tres enfoques de gestión de desarrollo de software: Tradicional, Scrum y Kanban.

Para realizar esta comparación se consultó material bibliográfico (referenciado en la sección bibliografía) y se realizó una síntesis para cada enfoque.

Esta síntesis se plasmó en un cuadro de doble entrada donde para cada enfoque se presentaron diferentes características o aspectos de análisis de cada uno.

Finalmente, presentamos una conclusión del trabajo realizado.

PRÁCTICO 14 - Comparación de Enfoques para Gestión del Desarrollo de Productos de Software

Consigna:

Presentar los diferentes enfoques para la gestión del desarrollo de productos de software, considerando los enfoques de gestión basada en procesos definidos (Tradicional), Scrum y Kanban y realizar una tabla comparativa que explique cada enfoque.

Objetivo:

Que el estudiante sea capaz de abordar la temática, no solo a partir de lo brindado por el docente durante las clases, sino también a través de investigación bibliográfica. Que consiga identificar aspectos que puedan ser comparados para obtener una decisión acertada en la elección de un determinado enfoque de gestión, en el momento en que lo necesite.

Propósito:

Aplicar los conceptos de Gestión del Desarrollo de Productos de Software basados en procesos definidos y basados en procesos empíricos.

Entradas:

Conceptos teóricos sobre los temas desarrollados en clase. Bibliografía sobre el tema. Artículos científicos de interés.

Salida:

Se evaluará que:

• Introduzca brevemente el trabajo a presentar y los enfoques de gestión con la información de interés.

• Realice un cuadro comparativo de los diferentes aspectos a contrastar.

• Presente una conclusión de elaboración propia del grupo en relación al tema abordado.

Instrucciones:

En grupos trabajarán la consigna.

• Prepararán un informe que contenga el trabajo

• Subirán en el aula virtual en la sección indicada por el docente.

Observaciones:

Deben referenciar el material bibliográfico utilizado.

Comparación de Enfoques para Gestión del Desarrollo de Productos de Software

|  |  |  |  |
| --- | --- | --- | --- |
| Enfoque  Aspecto | Tradicional | Scrum | Kanban |
| Objetivo | * Entregar para la fecha de entrega definida el producto de software que cumpla con todos los requerimientos según lo planificado al comienzo del proyecto. | * Realizar entregas frecuentes e incrementales de funcionalidades de software a lo largo de lo que dure el proyecto. | * Entregar un producto de trabajo o servicio a un cliente que lo solicitó atendiendo el nivel de prioridad del servicio y limitando la cantidad de servicios que se encuentran en progreso para evitar cuellos de botella. |
| Planificación | * Rigurosa y detallada. * Se realiza un plan minucioso sobre todo el trabajo que se va a realizar antes de desarrollar el software. * Se definen todos los roles, actividades, artefactos, herramientas y notaciones para el modelado. * El desarrollo de trabajo se realiza por flujos de trabajo de manera secuencial. * En cada flujo de trabajo se definen roles, actividades, artefactos y herramientas. * Para iniciar un nuevo flujo es necesario terminar el anterior. | * Flexible y adaptable. * Se realizan contratos de alcances que pueden variar a lo largo del desarrollo del proyecto de software. * Para planificar se suele ir de lo más general a lo particular * Se identifican requerimientos a un alto nivel (Epics) que después se dividen en requerimientos más específicos (Stories). Las stories se las estima en conjunto asignando un valor de Story points. Estas stories (priorizadas) arman el Product Backlog. De allí, se toman stories por prioridad para planificar una iteración (Sprint). | * Se identifica claramente el proceso según las etapas por las que pasan los trabajos del mismo. * Luego se definen los tipos de trabajo que se realizarán en este proceso. En el desarrollo de software pueden ser casos de uso, historias de usuario, defectos, tareas de mantenimiento, entre otras. * Para cada etapa dentro del proceso se limita la cantidad de trabajo en proceso (WIP). Se asigna una capacidad de trabajo (en proporción) a cada uno de los tipos de trabajos definidos anteriormente, además de definir para cada uno las distintas políticas de trabajo apropiadas. |
| Requerimientos | * Para obtenerlos, se realiza un proceso enfocado a identificarlos. Para esto se utilizan distintas técnicas como la entrevista, la observación, cuestionarios, entre otras. * La información obtenida de la toma de requerimientos, se analiza para extraer los requerimientos con distintos niveles de detalles según la vista que se esté aplicando. * Los cambios en los requerimientos no son bienvenidos. Se debe seguir todo un proceso para poder aceptar o rechazar un cambio. * Se traduce en la especificación de una ERS | * Los requerimientos, se ven reflejados en épicas cuando estos tienen un nivel de detalle muy bajo. Las épicas se desarman en stories hasta llegar a un nivel medio de detalle. Estas stories especifican una forma en la que un rol determinado interactúa con el sistema y que espera obtener como resultado. * Ya dentro del sprint, estas stories se dividen en tareas que tienen el nivel de granularidad necesario para que el equipo de desarrollo las trabaje. * Los cambios son bienvenidos siempre y cuando no interfieran en los requerimientos de la iteración que esté activa en el momento. | * Los requerimientos se traducen en algún tipo de trabajo según lo definido. Por ejemplo, en el desarrollo de sofware se pueden dividir en 3 tipos de trabajos, “Casos de uso”, “Mantenimiento” o “Bugs”. Y a cada clase se le asigna una porción de la capacidad de trabajo. De esta forma se limita la cantidad de requerimientos que se tienen en un momento dado. * Los cambios en los requerimientos son aceptables siempre que haya capacidad de trabajo disponible |
| Gestión del cambio | * Resistencia a los cambios. * No hay retroalimentación ante los problemas. * Exceso de burocracia * Ocurren desfases entre lo solicitado y entregado y entre el producto y la situación real del mercado * Hacer un cambio al alcance requiere de un proceso formal de control de cambios | * El cambio forma parte del proceso * La única condición por la que no puede haber un cambio es durante la fase de ejecución del sprint | * Se definen clases de servicio que clasifican las tareas que deben ser gestionadas de manera similar aplicando diferentes políticas a cada tipo de trabajo. |
| Métricas | * Exhaustivas * Se debe saber elegir las que mejor representen al proyecto en curso, al proceso utilizado y al producto que se construye. * Definidas de acuerdo a si están en el dominio del proyecto, del producto o del proceso.   Se definen las básicas:   * Tamaño del producto * Esfuerzo: estimar la cantidad de Hs/Hombre invertidas en un proyecto * Tiempo calendario definido para el proyecto * Estimación de defectos para el producto | * Se mide sólo lo necesario. * Capacidad: Estimación del tiempo ideal que el equipo dispone para completar un trabajo en un período dado. Se mide en esfuerzo (horas) o story points por días disponibles para una iteración. * Velocidad: Observación empírica de la capacidad del equipo para completar el trabajo por iteración. * Running Tested Features: medir cuánta funcionalidad se ha entregado para la implementación en cada iteración. | * Se mide para evitar y disminuir el “cuello de botella” y el tiempo ocioso. * Cycle Time: Tiempo entre el inicio y el final del proceso. (Días de trabajo o esfuerzo) * Lead Time: Tiempo entre que se pide un ítem de trabajo y el de su entrega. (Días de trabajo) * Touch Time: Tiempo que pasó en columnas de trabajo en curso. (Días hábiles) * Eficiencia del ciclo del proceso: Se calcula dividiendo el Touch Time por el tiempo que paso. |
| Estimaciones | * Puede realizarse en horas de trabajo o líneas de código por hora/día. * Al comienzo del desarrollo suelen ser poco certeras, pero se van afinando a medida que avanza el proyecto, es decir a medida que se completa el desarrollo. | * Se estima en horas de trabajo o story points por día de trabajo. Es decir, se estima según la capacidad de trabajo del equipo. | * Las estimaciones no son un requisito * Se puede estimar en días hábiles de trabajo * El objetivo de las estimaciones es conocer el proceso y el trabajo en curso, limitar el trabajo en curso y optimizar el flujo de trabajo |
| Roles | Está relacionado al tipo de actividades que se realizan en cada etapa del desarrollo de software:   * Analista funcional * Arquitecto de software * Diseñador * Programador * Tester   Líder de proyecto: actúa de nexo entre el equipo de desarrollo y los diferentes mandos de la organización y stakeholders del proyecto. | Se definen roles especiales para Scrum orientados al seguimiento y control de los equipos de desarrollo (Scrum Master) seguimiento y control del desarrollo del producto de software (product owner) y desarrollo del producto de software en sí (Development team)   * Scrum Master * Product Owner * Development Team | * No prescribe roles.   En la práctica se define:   * Gestor de peticiones de servicio: responsable de entender las necesidades y expectativas de los clientes, y de facilitar, seleccionar y ordenar los elementos de trabajo * Gestor de prestación de servicios: es el responsable del flujo de trabajo entregando los elementos seleccionados a los clientes y facilitando la Reunión de Kanban y la planificación de la entrega |
| Equipo de trabajo | * Cada miembro del equipo tiene una especialización dentro del flujo de trabajo. * Las decisiones son tomadas de manera autocrática y luego son informadas a los demás miembros del equipo. * Hay una cadena de mando implícita y una relación de jerarquía entre cliente, empleado y jefe | * Equipos autoorganizados, multifuncionales, colaborativos. * Las decisiones son tomadas en conjunto luego de deba.tir soluciones * No existe una cadena de mando explícita. Tanto cliente como equipo trabajan conjuntamente | * Equipos multifuncionales y/o especializados. * El trabajo se realiza de forma independiente de acuerdo al grado de especialización que tiene algún miembro del equipo. * Las decisiones se toman en forma conjunta entre los miembros con el objetivo de evitar siempre cuellos de botella |
| Compromiso | * Se asume el compromiso por la totalidad del proyecto hasta que este se complete y cumpla con los requerimientos. | * El equipo asume un compromiso por cada iteración | * Se puede asumir un compromiso como no. Es algo opcional. |
| Sistema | * Predictivo. * Todo el proceso está planificado. * Se puede predecir a largo plazo las variables del proyecto | * Prescriptivo. * Para el proceso se definen ciertas pautas e información obtenidas del conocimiento experto. | * Adaptativo. * Sólo se define el WIP (Work in Progress) para gestionar el flujo de trabajo y evitar cuellos de botella. |
| Iteraciones de trabajo | * Se definen flujos de trabajo de duración variable. Cada flujo de trabajo posee fases compuestas por un conjunto de iteraciones también variables | * Se definen time-box fijos de duración máxima de 2 a 4 semanas dependiendo del tamaño del equipo de trabajo. | * Trabajo continuo. * Se realiza el trabajo cuando hay demanda |
| Eventos importantes | * Reunión para la planificación del proyecto: definición del plan de proyecto para definir el alcance del proyecto definición del proceso y ciclo de vida a utilizar, definición del plan de riesgos, la definición de asignación de recursos y de controles y métricas. * Reunión de relevamiento y análisis: investigar las solicitudes de requerimientos que necesita el cliente * Reunión de seguimiento y control: reuniones programadas por la gerencia para evaluar el seguimiento del proyecto y control de presupuesto | * Sprint: desarrollo del producto en iteraciones sucesivas. * Sprint Planning: tiene como propósito definir el objetivo del Sprint y negociar qué ítems del backlog pasarán a ser desarrollados. * Daily Scrum: es la reunión diaria fija. está orientada a realizar una inspección y adaptación del trabajo diario. * Sprint Review: se revisa todo el trabajo realizado por el Equipo de Desarrollo, y sirve para inspeccionar el incremento * Retrospective: reunión que busca mejorar la forma en que el Scrum Team desempeña su trabajo. | * No prescribe reuniones. * Replenishment meeting: reunión que busca establecer prioridades y seleccionar cuales son las siguientes prioridades de trabajo. * Daily Kanban: es una reunión en la que los miembros del equipo de un sistema o parte de un sistema Kanban se reúnen cada día para evaluar el trabajo en curso y acciones a tomar. * Service Delivery Meeting: está orientada a examinar y mejorar la efectividad del servicio. * Delivery Planning Meeting: reunión que se realiza para revisar y planificar entregas al cliente. * Strategy Review: durante esta reunión, se seleccionan los servicios que deben proveerse a nivel organizacional y así como establecer las condiciones necesarias y dar dirección al servicio. * Operations Review: El objetivo es entender el balance entre servicios, desplegar recursos para maximizar la entrega de valor de acuerdo con las expectativas de los clientes. * Risk Review: esta reunión se hace para entender los riesgos asociados a la entrega de valor. |

Conclusión

Hemos visto que los tres enfoques distan uno de otro, pero no hay ni buenos ni malos. La clave en la selección de uno u otro enfoque está en lo que realmente se necesita.

Recordando el modelo Cynefin que se vio en clase y que nos sirve para apuntar nuestras conclusiones, vemos que una metodología tradicional puede ser aplicada en un ambiente **complicado**, ya que en manos expertas y especializadas aplicando buenas prácticas es posible llevar a cabo el proyecto. Si bien se puede aplicar metodología ágil en este universo los proyectos que se encuentran aquí no tienen incógnitas desconocidas y para equipos de desarrollo que trabajan con metodología ágil puede resultar aburrido o abrumador porque se convierten en proyectos repetitivos, que no aportan creatividad ni representan un reto para el equipo.

Es que surge entonces el otro escenario, el **complejo** pues en este entorno es donde mejor funcionan los métodos ágiles, aquí la estructura no es fácil de entender para el equipo (primera variable) y no existe algún experto que pueda apoyar al equipo porque generalmente en este escenario se da la innovación, el conocimiento emerge mientras el proyecto avanza, muchas cosas no se han realizado antes, la incertidumbre es alta, el modelo tradicional se vuelve poco recomendado, pues si se piensa realizar requisitos detallados muy probablemente tiempo después tendrán que ser desechados porque poco a poco se va teniendo nuevo conocimiento que antes no se tenía y que termina modificando los requerimientos. Este escenario es el más cómodo para los equipos comprometidos y que les gusta enfrentar nuevos retos y aquí Scrum puede ser usado para atacar el proyecto de software.

Finalmente, el escenario caótico donde está el lema “tenemos que entregar como sea”, se necesita respuesta inmediata para regresar a tener un poco de orden y este claramente no es un escenario para usar Scrum y mucho menos un enfoque tradicional. Aquí, en este entorno creemos que puede ser usado Kanban.

Tres enfoques diferentes uno basado en procesos definidos (Enfoque tradicional) y otros dos en procesos Empíricos (Kanban y Scrum).

Las comparaciones realizadas nos son útiles para conocer cuál es el que más se adapta de acuerdo al contexto o universo de desarrollo en donde estemos parados.

Bibliografía utilizada

* Sommerville, Ian - ***Ingeniería De Software*** - Novena Edición -Editorial Addison-Wesley Año 2011.
* Steve Mc Connell., ***Desarrollo Y Gestión De Proyectos Informáticos*** Editorial McGraw Hill – Año 1996.
* Pressman, Roger - ***Ingeniería De Software, Un Enfoque Práctico***. Séptima Edición - Editorial McGraw Hill – Año 2010.
* Henrik Kniberg & Mattias Skarin, ***Kanban Y Scrum – Obteniendo Lo Mejor De Ambos,*** Editorial C4MediaInc – Año 2010.
* David J. Anderson and Andy Carmichael, ***Essential Kanban Condensed***, digital version - Editorial Lean Kanban University Express April - 2016.
* Filminas de clase teórica dispuestas por la cátedra
* Roles y Responsabilidades en Kanban. Disponible en: [Roles en Kanban](https://ipmoguide.com/kanban-roles/)
* Trabajo de posgrado: “Caracterización de Especificación de Requerimientos en entornos Ágiles: Historias de Usuario”. Autora María Paula Izaurralde, febrero 2013. Disponible en: [Tesis Requerimientos en metodología ágiles](https://www.institucional.frc.utn.edu.ar/sistemas/lidicalso/pub/file/Tesis/Anteproyecto_Requerimientos_en_Metodolog%C3%ADas_Agiles.pdf)