Propuesta de nuevo método para desarrollo de aplicativo MSDS (Método simplificado en desarrollo de software)

John Stiven Moscoso Zapata

INSTITUCIÓN UNIVERSITARIA PASCUAL BRAVO
FACULTAD DE INGENIERÍA
TECNOLOGÍA EN DESARROLLO DE SOFTWARE
MEDELLÍN
2022

Propuesta de nuevo método para desarrollo de aplicativo MSDS (Método simplificado en desarrollo de software)

John Stiven Moscoso Zapata

Trabajo de grado para optar al título de Tecnólogo en desarrollo de software

Asesor
Liliana María García Aguirre

INSTITUCIÓN UNIVERSITARIA PASCUAL BRAVO
FACULTAD DE INGENIERÍA
TECNOLOGÍA EN DESARROLLO DE SOFTWARE
MEDELLÍN
2022

Tabla de contenido

Lista de figuras

Lista de tablas

1. Resumen

MÉTODO SIMPLIFICADO EN EL DESARROLLO DE SOFTWARE (MSDS)

John Stiven Moscoso Zapata

Hoy en día, el ritmo frenético y el dinámico de la industria del software ha hecho que se replanteen los fundamentos en los que se basa el desarrollo de proyectos de software tradicional. Las investigaciones recientes, así como el mercado existente están marcando las pautas para la aplicación de *metodologías en el desarrollo de software*, teniendo como cualidades primordiales cumplir con las necesidades de rapidez, fluidez y flexibilidad para cumplir con los criterios y las necesidades de los clientes en el menor tiempo posible.
Ante esta circunstancia, el nivel de adaptabilidad de las metodologías tradicionales a los nuevos espacios de trabajo no eran eficaces y por lo tanto no abarcaban las necesidades de un mercado que iba creciendo rápidamente, por lo que en este proyecto se propone una nueva metodología llamada *MÉTODO SIMPLIFICADO EN EL DESARROLLO DE SOFTWARE (MSDS)*, donde se categoriza como una *metodología ágil* y eficiente y se tratara al igual sobre los dos grandes grupos existente para el desarrollo de software que son las tradicionales o pesadas y ágiles.
El objetivo de esta investigación es exponer e introducir sobre esas metodologías existentes y dar a conocer la propuesta de una nueva metodología.

2. Abstract

MÉTODO SIMPLIFICADO EN EL DESARROLLO DE SOFTWARE (MSDS)

John Stiven Moscoso Zapata

Nowadays, the frenetic and dynamic pace of the software industry has led to rethink the fundamentals on which the development of traditional software projects is based. Recent research, as well as the existing market are setting the guidelines for the application of methodologies in software development, having as primary qualities to meet the needs of speed, fluidity and flexibility to meet the criteria and needs of customers in the shortest possible time. Given this circumstance, the level of adaptability of traditional methodologies to new workspaces were not effective and therefore did not cover the needs of a market that was growing rapidly, so this project proposes a new methodology called SIMPLIFIED METHOD IN SOFTWARE DEVELOPMENT (MSDS), where it is categorized as an agile and efficient methodology and will be treated equally on the two major existing groups for software development which are the traditional or heavy and agile.
The objective of this research is to expose and introduce these existing methodologies and to present the proposal of a new methodology.

3. Glosario

Software: termino informático para un conjunto de programas que comprenden componentes lógicos interpretados en una computadora que hacen posible la realización de tareas específicas

Metodología: es una serie de técnicas, pasos y métodos racionales utilizados para concluir objetivos propuestos dentro de un proyecto, investigación, exposición o tarea propuesto.

Método: es la técnica o el procedimiento de un modo ordenado que se sigue para alcanzar uno o varios objetivos

Metodología tradicional: este tipo de metodologías como el nombre lo indica son las que se han utilizado desde un comienzo, son métodos robustos, que se convirtieron en ineficientes.

Metodología ágil: son metodologías eficientes adaptativas y muy flexibles que involucran a cada actor de un proyecto de manera iterativa.

Framework: es un campo de trabajo o una estructura prediseñada para desarrollar un proyecto

Stakeholders: es un término propuesto y designado en los años 80s como el grupo de interés que están involucrados de una u otra manera sobre las acciones que determina un proyecto grupo o empresa.

Iteración: tiene como definición la repetición de un proceso para alcanzar una meta, objetivo o resultado.

Codificación: es el método que permite representar la **información** utilizando un conjunto de símbolos que se combinan siguiendo determinadas reglas.

Ciclo de vida desarrollo de software: se designa como las diferentes etapas en el desarrollo de un sistema o proyecto informático o de software

4. Introducción

El desarrollo de software es una de las industrias en el sector de la tecnología con más competitividad en el mundo por lo que ha tenido una evolución constante en lo que se refiere a las metodologías, las formas en que se realiza la preparación para el diseño y desarrollo del programa de software, principalmente con el fin de mejorar, maximizar los procesos y ofrecer una mejor calidad.

Hoy en día encontramos múltiples propuestas metodológicas que son aplicables en poca o gran medida en el desarrollo de los procesos llevados a cabo en un proyecto. El objetivo es presentar una propuesta de una nueva metodología, después de hacer un análisis concienzudo de las que ya existen, identificando las debilidades y fortalezas. Esta metodología busca que se adapte a las necesidades de los diferentes grupos de desarrollo, quienes buscan entregar a sus clientes excelentes desarrollos, utilizando los tiempos adecuados en cinco fases de desarrollo que se expondrán en el documento.

Entre las metodologías que se analizaron están los modelos tradicionales en cascada, modelo en espiral, modelo en fuente y metodologías ágiles como el scrum, la gestión ágil de proyectos, el desarrollo orientado a pruebas o TDD, el Extreme Programming (XP), el Método de Desarrollo de Sistemas Dinámicos (DSDM) y la metodología de cristal.

5. Planteamiento Del Problema

Descripción

Actualmente en el campo del desarrollo de software existen metodologías que se han utilizado durante años, esto con el fin de realizar trabajos que deben llegar a ser confiables, pero sobre todo de calidad, esto se complementa con un equipo con experiencia.
Estas metodologías son una herramienta muy importante para tener la capacidad de realizar una tarea de desarrollo de software especializado de manera eficiente y así poder terminar con éxito un proyecto y obtener los resultados previstos.

Aunque existen muchas metodologías, todavía muchos equipos de desarrollo no logran adaptarse a ellas, algunas veces realizan unos híbridos donde unen unas metodologías con otras y así experimentan y esto hace que se demoren en las entregas, la propuesta del método simplificado de desarrollo nos define las bases para que ese problema de tiempos y calidad sean superados en cinco fases de desarrollo, Visión, Preparación, Avances, Estabilidad e Implementación.

5.1 Formulación

¿Puede una nueva metodología resolver las necesidades específicas de desarrollo en las que la calidad y el tiempo de entrega sean las prioridades, y en la que el grupo de desarrollo sea la base para la solución?

6. Justificación

Los productos de software de alta calidad tienen que utilizar un método adecuado para su construcción, que se centra en las etapas de uso metódico y también en las acciones para la gratificación de los propósitos propuestos para completar el proyecto. Dicha opción implica un conjunto de conceptos existentes y propuestos en el presente proyecto que deben ser seguidos y acatados.

Estos consisten en actividades específicas para entender el problema, así como para conectar con el cliente, técnicas específicas para representar un diseño, las mejores prácticas para implementar el servicio, en técnicas y tácticas sólidas para la selección.

Como se expondrá en el documento se dará claridad sobre la importancia fundamental que nos otorga las metodologías en el desarrollo de software para dar soluciones a las distintas necesidades de los clientes en un mercado creciente de las tecnologías a nivel mundial, la toma de decisiones y la evolución en los proyectos se definen con la importancia de adoptar los métodos necesarios para desarrollarlos no solo en el proyecto si no también en todo el equipo de trabajo involucrado.

En este proyecto se hace un análisis sobre las metodologías existentes y a partir de este, se presenta una propuesta de una nueva metodología definida como Método Simplificado En El Desarrollo De Software que basado en un desarrollo iterativo e incremental fortifican las capacidades en un equipo o grupo colaborativo y multidisciplinario de desarrollo con tareas cooperativas que simplifican y nos llevan a una entrega de solución efectiva a nuestros clientes.

7. Objetivos

7.1 Objetivo general

Realizar un análisis de las metodologías de desarrollo de software que existen al día de hoy y proponer una nueva metodología que cumpla con las necesidades de los proyectos modernos

7.2 Objetivos específicos

- Identificar todas las metodologías aplicables en uno o varios proyectos solicitados.
- Realizar paralelos de las metodologías donde se identifiquen las ventajas y desventajas dependiendo del contexto en el que se desarrolla el proyecto.
- Demostrar por medio de consultas casos reales sobre metodologías aplicadas en empresas reconocidas en el medio tecnológico.
- Proponer una nueva metodología que ayude a los proyectos modernos en programación de software.

8. Marco Teórico

Definición desarrollo de software

El desarrollo de software se refiere a un conjunto de tareas informáticas dedicadas al procedimiento de producción, creación, publicación y soporte de aplicaciones de software.
El software en sí mismo es el conjunto de instrucciones o programas que indican a un sistema informático lo que debe hacer. Es independiente del equipo y hace que los sistemas informáticos sean programables. Hay tres tipos básicos:
Software de sistema, software de programación y software de aplicación.

El desarrollo de software lo realizan en gran medida los diseñadores, los ingenieros de software y los programadores. Estos deberes se conectan y también se superponen, y la dinámica entre ellos difiere considerablemente entre los departamentos de desarrollo y también las comunidades **(IBM, 2010).**

Metodologías

Una metodología se compone principalmente de hacer uso de diferentes herramientas, técnicas, métodos y también versiones para el desarrollo. Frecuentemente este tipo de técnica necesita ser documentada, para asegurar que los diseñadores que ciertamente permanecerán en la preparación de la tarea, reconozcan perfectamente la metodología y también en muchos casos el ciclo de vida de la aplicación de software que se planea cumplir.
Aunque actualmente existe una gran variedad de metodologías de programación. El hecho es que todas ellas se basan en ciertas estrategias generalistas que se produjeron hace años, algunos tipos de enfoques de crecimiento de software que se hicieron uso, así como inventado en el inicio de nuestra era tecnológica.

Las metodologías aportan al proceso lo siguiente:

Optimiza el proceso y el producto software.
Métodos que guían en la planificación y en el desarrollo del software.
Define qué hacer, cómo y cuándo durante todo el desarrollo y mantenimiento de un proyecto.

Hay diferentes metodologías y métodos que han permanecido en los años como herramientas de apoyo para el desarrollo de aplicaciones de software y se debe diferenciar entre metodologías y métodos de desarrollo de software.
(Rojas, marzo 2017)

Modelo de desarrollo de software

Es una representación simplificada del procedimiento de avance del programa de software, presentado desde un punto de vista particular.

Metodología de desarrollo de software

Es un enfoque estructurado para el desarrollo de aplicaciones de software que consiste en modelos de sistemas, notaciones, políticas, punteros de diseño y también estándares de procesos.
(Carlemany, 2011)

¿Qué es una metodología y para que se utiliza?

Las metodologías de desarrollo de software son una colección de métodos organizativos que se componen en el desarrollo de soluciones de programas de sistemas informáticos. El propósito de los diferentes enfoques es tratar de organizar los grupos de trabajo para asegurar que crean las características de un programa en el mejor medio posible.

Las metodologías son una de las fases específicas de un trabajo o proyecto que parte de un emplazamiento académico, así como provoca una opción de estrategias o enfoques concretos respecto al procedimiento para la gratificación de los objetivos. Es el conjunto de métodos utilizados en una actividad prevista para definirla y potenciarla. Indica los pasos que hay que seguir y cómo hacerlos para completar un trabajo.

Metodología tradicional

La gestión tradicional es un método bien establecido en el que un proyecto pasa por un conjunto de procesos de vida, que incluyen las etapas de iniciación, planificación, ejecución, seguimiento y cierre.

Todo el proyecto se prepara de antemano sin posibilidad de transformar los requisitos/necesidades. Este enfoque supone que tanto el tiempo como el precio son variables y que, por otro lado, las necesidades son atendidas. Por ello, los gestores de tareas estándar suelen enfrentarse a problemas de presupuesto y también de calendario.

(ARVIZU BÁRCENAS LAURA, 2015)

Metodologías ágiles

Los sistemas convencionales se concentran en una planificación agresiva en la que son necesarios factores como el precio, la extensión y también el tiempo, sin embargo, el seguimiento ágil de los proyectos prioriza la sinergia, la colaboración con los clientes, así como la versatilidad. Se trata de un proceso repetitivo que se concentra en los comentarios de los clientes y en los lanzamientos continuos en cada iteración de la tarea de crecimiento de la aplicación de software.

La esencia del crecimiento ágil de software no es un proceso establecido, sino un énfasis en la colaboración para lograr resultados y también el ajuste evolutivo. Las metodologías Ágiles son extra adaptables a los cambios de especificaciones, así como a los desarrollos, por lo que se invierte menos tiempo en la planificación preliminar y también en la priorización.

(McCarthy, 2018)

Diferencias entre la gestión ágil y tradicional

Flexibilidad

La gestión tradicional no es rentable para ejecutar las modificaciones de los proyectos. Se trata de un proceso agotador y, una vez terminada la estrategia, el supervisor se encarga de conectarla con el grupo y de garantizar que cada miembro del personal lleve a cabo el plan de acuerdo con sus necesidades. Hay mucha resistencia a realizar cambios, ya que pueden obstaculizar el progreso de la tarea. En cambio, la gestión ágil ofrece mucha más flexibilidad a la hora de realizar cambios. Los miembros del personal pueden intercambiar sugerencias para ayudar a mejorar el producto. Los enfoques ágiles se centran mucho más en el desarrollo de los productos adecuados que en la supervisión de una estructura rígida.

Propiedad y transparencia

En la gestión de proyectos tradicionales la posesión viene del supervisor de la tarea, por lo que está a cargo de la intención y también el registro del viaje del producto, no hay participación del cliente, el miembro del personal por lo general no tiene voz en los resultados de sus iniciativas o el progreso de la tarea. En la técnica de la destreza, el trabajo proviene del equipo. Todos trabajan juntos para encontrar un plan, así como para ver el desarrollo del artículo en todo momento. Toda esta apertura desempeña una función vital en el mantenimiento de un entorno de trabajo eficaz y extremadamente cohesionado.

Solución de problemas

En caso de que se produzca un fallo inesperado, los miembros del personal deben informar del problema al director. En el método ágil, los grupos están facultados para tomar sus propias decisiones. De este modo, resuelven el problema internamente y no pierden tiempo innecesariamente. Al iniciar y atender el proceso, las sugerencias ayudan a solucionar muchos de los problemas que impiden el desarrollo. Por lo tanto, los miembros del equipo rara vez aumentan los problemas menores o irrelevantes con los administradores, a menos que se tomen decisiones extremas.

Se puede concluir entonces que la administración tradicional en el nuevo mercado es un proceso rígido e inadaptable, centrado en la gestión jerárquica y la interacción unidireccional. Por otro lado, la gestión ágil tiene un proceso de trabajo mucho más versátil, así como polivalente, centrado en la resolución de cuestiones en lugar de seguir un proceso extenuante.
(Rodelgo)

Metodología ágil	Metodología tradicional
Pocos artefactos. El modelo es prescindible, modelos desechables	Más artefactos. El modelo es esencial, mantenimiento en los modelos
Pocos roles, más genéricos y muy flexible	Más roles y totalmente específicos.
El cliente posee un rol en el equipo de trabajo	El cliente interactúa con el equipo de desarrollo solo en ocasiones denominadas reuniones.
Orientada a proyectos ágiles y en ocasiones pequeños, y en el mismo lugar	Designada a proyectos de cualquier tamaño, pero suelen ser especialmente de gran trabajo y robustos
La arquitectura se va definiendo y mejorando a lo largo del proyecto	Se promueve que la arquitectura se defina tempranamente en el proyecto
Énfasis en los aspectos humanos: el individuo y trabajo en equipo	Énfasis en la definición de los procesos: tareas, roles y herramientas
Basada en prácticas de elaboración de códigos	Basadas en estándares y normas de un entorno de desarrollo
Se producen pequeños cambios en el desarrollo del proyecto	No se espera que Ocurran grandes cambios durante el desarrollo

Diferencias entre las metodologías ágiles y tradicionales de desarrollo de software.

El concepto de calidad en el software

Una faceta crucial del desarrollo de software es la calidad superior. Inicialmente, el avance del desarrollo era manejar el proceso de construcción de un aplicativo o un sistema que cumple con los requisitos de los clientes, probarlo, instalarlo en el entorno de trabajo, preservarlo y hacerlo evolucionar con las modificaciones de la empresa. Como resultado, un aspecto de la calidad del desarrollo de software es llevar a cabo este proceso de la mejor manera en la que permite que la tarea conectada se complete en el tiempo programado, así como dentro del plan de gastos asignado. Aunque esta idea, así como las actividades relacionadas, se han establecido como parte de las nuevas competencias de diseño, todavía se considera un gasto en algunas empresas. No obstante, se han alcanzado importantes acuerdos en la comunidad sobre la garantía de calidad como factor determinante en las tareas. Existen estándares de trabajo, así como versiones vinculadas que comprenden una recomendación en la mejora de los procesos relacionados con el avance de los procesos de desarrollo de software.

Algunos aspectos de lo que las metodologías no se ocupan son:

- Las personas se conectan hablando cara a cara y a partir de diálogos con preocupaciones y respuestas.
- Las personas no son consistentes en su creencia de forma continua en el tiempo.
- Las personas son variables de un día a otro y de un lugar a otro.
- Los individuos generalmente desean sentirse integrados en la empresa, se preguntan, toman iniciativas y están dispuestos a aplicarse para hacer efectivos los proyectos.
- Los individuos necesitan tiempo para asumir roles y también posibilidades de interactuar.
- Las personas funcionan mejor, por ejemplo.
- Los individuos prefieren dejar de trabajar con cautela en lugar de prosperar asumiendo riesgos.
- Los individuos tienen hostilidad a la transformación.
- Las personalidades individuales dominan los equipos.

- La personalidad personal afecta a la capacidad de realizar determinados trabajos.

Además, en conjunto los resúmenes de los diferentes métodos son:

- Largos y también complicados.
- Mezclan resúmenes de procesos con productos elaborados.
- No manejan pequeñas reglas que generan grandes impactos.
- No se ocupan de las personas y también de su lugar de trabajo, sino de las funciones.

Estas observaciones nos dan un enfoque para acertar en la importancia de la existencia de ciertas condiciones en el ambiente de trabajo que demuestran estos comportamientos. Sin estas condiciones ninguna metodología es asumida como la mejor o la que se debe seguir por encima de una u otra, ya que están perdidas o nulas las bases que aseguran a las personas desarrollarse como profesional.

Proyectos de software y las condiciones de trabajo

Trabajo iterativo

Es válida si la organización ha decidido trabajar en forma iterativa. Está compuesta de los siguientes ingredientes:

Cultura organizacional:

Debe ser consistente la forma iterativa de desarrollo con el modo en que los proyectos son vendidos. No pueden venderse proyectos de alcance, presupuesto y planificación fijos. Los acuerdos deben explicarse y ser entendidos por todos los involucrados. Estos aspectos relacionados a la forma de trabajo deben ser el elemento de cohesión del trabajo compartido por la gerencia, el departamento de ventas, marketing y área de desarrollo.

Grupo de desarrollo:

De parte del grupo de desarrollo se espera actitud y compromiso para con el proyecto. Esto se traduce en el líder planeando y siguiendo según los riesgos, analistas y desarrolladores

conduciendo su trabajo con los requerimientos, una definición de la arquitectura como basamento y monitoreo de la calidad de procesos y productos en cada momento del proyecto.

Organización cliente:

Dispuesta a establecer acuerdos centrados en la confianza que genera el conocimiento de la forma de trabajo y el trabajo compartido más que en conflicto de intereses y multas por incumplimientos. Actitud para participar del proceso de desarrollo además de soportar el proyecto.

Si bien son factibles diferentes formas de trabajo con distintas dinámicas, como veremos en los capítulos próximos, el desarrollo de software ha obtenido una madurez que indica a la iterativa como la forma más adecuada. Los ingredientes anteriores son los necesarios para que esta forma de trabajo se convierta en una propiedad deseable.
(Wrike)

9. Metodologías en el desarrollo de software

La ingeniería de software aplica los principios de las ciencias de la computación, así como las matemáticas, para lograr opciones rentables a los problemas del desarrollo de software. También es la aplicación organizada, disciplinada y cuantificable al crecimiento, funcionamiento y mantenimiento del software.

Al principio, el desarrollo de software era un programa realmente pequeño como resultado de las limitaciones del hardware disponible en aquellos días. A medida que aumentaba la capacidad de cálculo, crecía el tamaño y la complejidad del software establecido. Surgieron varios métodos para ayudar a gestionar esta complejidad: estrategias relacionadas con la configuración de lenguajes; estudios exhaustivos sobre el diseño de aplicaciones de software; estilo de software y dispositivos CASE (ingeniería de software asistida por ordenador). Después de un período de éxito, la situación del programa de software se determinó en los años sesenta, pero sus efectos se siguen sintiendo realmente hoy en día. Básicamente, la situación del software se basa en los problemas de suministrar programas sin defectos o plagas, fáciles de entender y también verificables. Se han recomendado numerosos enfoques para intentar superar estos problemas, pero lo cierto es que todavía no existe ninguna técnica que nos permita conocer el precio real y la duración de un proyecto antes de que comience.

El primero de los efectos que todavía podemos ver hoy en día revela que de cada cuatro tareas de software se queda corta con respecto a la oferta. Además, las tasas de rotación de personal de alrededor del 20% se consideran regulares. Otro problema es que las grandes tareas alargan los periodos de crecimiento de tres a cinco años, con los problemas que ello conlleva, haciendo que varios de los programas queden desfasados antes incluso de ser implantados. Por último, el mantenimiento del software es uno de los mayores costes relacionados con la informática en muchas empresas.

Un proceso de avance de software es una estructura utilizada para el crecimiento de un software. Los sinónimos básicos son "proceso de vida" y "procedimiento de software". Existen muchos modelos para estos procedimientos, cada uno de los cuales define varios métodos para una selección de tareas, así como las tareas que deben realizarse a lo largo del proceso. Proceso de

crecimiento del programa de software Las secciones siguientes muestran las acciones que deben realizarse a lo largo del proceso.

•*Investigar los requisitos de los usuarios. Esto se lleva a cabo durante la fase de análisis.*

La mayoría de los clientes, si no todos, no saben exactamente lo que desean. Esto se debe a que la mayoría de ellos no saben exactamente cuáles son las actividades que realizan durante el proceso.
Por eso el análisis requiere que el desarrollador acabe siendo deliberadamente un especialista en el dominio del usuario para ayudarle y dirigirle en la definición de sus necesidades.
Podemos dividir esta fase en cuatro áreas: En primer lugar, el desarrollador debe prestar atención así como observar, tratando de descubrir la mayor cantidad de detalles posibles; después, tiene que dudar así como tratar de aclarar lo más alto posible la información recopilada; después, el desarrollador necesita comprobar la información y también recomendar servicios; finalmente, una vez que realmente ha entendido lo suficiente del asunto, debe escribir el archivo con la especificación de las demandas.

•*Definir las características necesarias para el sistema (especificación).*

La especificación de las necesidades es la última fase de la tarea de análisis. En ella hay que exponer de forma inequívoca cuáles son las acciones necesarias. En el documento se recogen anotaciones formales, archivos estructurados y ejemplos. El objetivo es lograr una especificación de necesidades que, de forma correcta y conecte con el desarrollador los atributos necesarios para el sistema.

•*Crear una solución adecuada al problema, es decir, la creación del proyecto.*

El proyecto busca desarrollar un servicio que satisfaga las necesidades, basándose en la experiencia acumulada basado en la experiencia acumulada (así como en las estrategias estandarizadas). Los proyectos normalmente requieren introducir trabajos que necesitan innovar en algún grado, produciendo una serie de servicios factibles y también utilizando alguna métrica

para seleccionar uno de ellos. El resultado final es un archivo de trabajo que comunica de forma inequívoca la tarea a los que la van a ejecutar.

•*Desarrollar la solución propuesta (implementación).*

Durante esta fase se lleva a cabo el desarrollo de la aplicación en sí misma. Donde se inicia con la estructuración del código, documentarlo, corregir cualquier error que se detecte, prueba del código, y enviar la información necesaria al analista, así como al testador. El objetivo es alcanzar en los tiempos establecidos la depuración del código para continuar con el siguiente proceso.

•*Garantizar que la solución responde al problema originalmente propuesto y que esta funciona correctamente en el contexto a ser ejecutada (integración). Para ello se llevan a cabo diferentes pruebas.*

En esta etapa se comprueba si la implementación corresponde al proyecto y si esta funciona correctamente y atiende a todos los requisitos planteados al inicio del proceso. Debe testear los módulos individuales y el sistema por completo y la interacción con el entorno, software, datos, etc. existentes.

•*Modificar las soluciones de trabajo cuando nuevos requisitos son presentados o identificados (mantenimiento).*

El hecho es que las necesidades individuales avanzan y se alteran con el tiempo. gradualmente. A pesar de la amplitud de los exámenes realizados, es posible que no descubran todos los problemas antes de que se suministre la aplicación de software. Como resultado, la aplicación de software se tiene que alterar adicionalmente con el tiempo.

Los ajustes en las demandas pueden dar lugar a aplicaciones y exámenes adicionales, o a un trabajo añadido, e incluso a una evaluación. Paralelamente al procedimiento anterior, hay que llevar a cabo la preparación y administración de todas las actividades. Para ello, es imprescindible realizar un programa o cronograma de los trabajos con la debida antelación, dando las fuentes necesarias para que las tareas tengan todos los problemas necesarios para

lograr sus propósitos. Además, hay que examinar el rendimiento de todas las actividades y descubrir los medios para optimizarlo. Otro punto esencial es acordar con el cliente los plazos y también las calidades del envío que se va a realizar.

10. METODOLOGÍAS TRADICIONALES

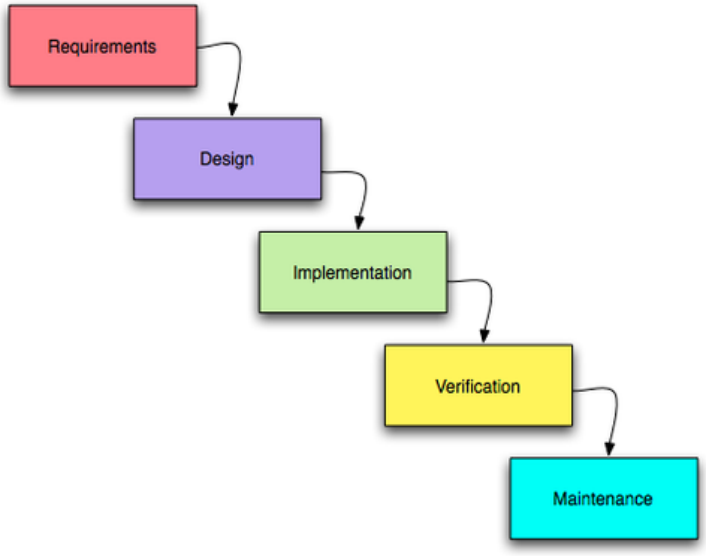
Las metodologías tradicionales o rígidas en el desarrollo de software son aquellas que desarrollan una técnica de trabajo en el procedimiento de crecimiento del software, con el propósito de lograr una aplicación de software extra eficiente.

Se definen especificando y desarrollando de forma absoluta y rígida todas y cada una de las necesidades al inicio de los trabajos de ingeniería de software. Estos métodos son poco flexibles y no permiten modificaciones.

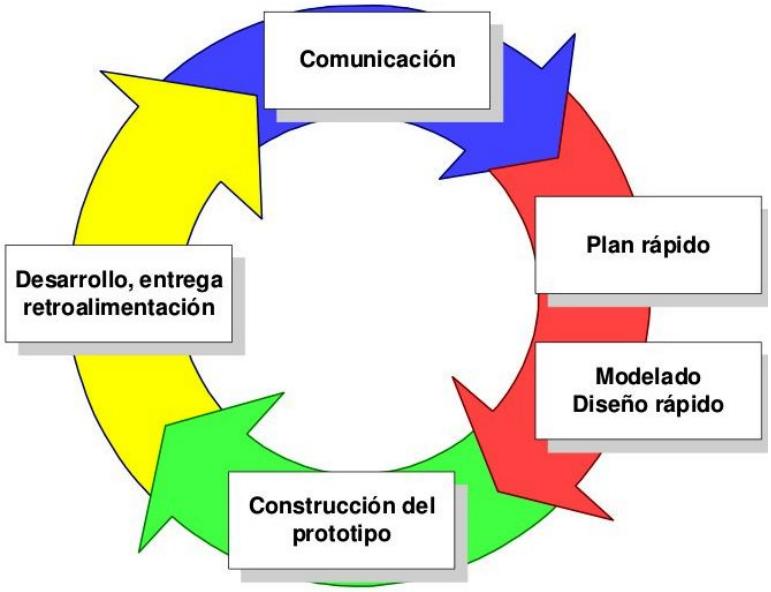
El método tradicional funciona aplicando un enfoque directo en el que las fases del procedimiento de crecimiento del programa de software deben coincidir entre sí de forma secuencial. Es decir, una fase tiene que estar terminada antes de que empiece la siguiente, estas fases recogen la recopilación de requisitos y la documentación.

Modelo tradicional en cascada:

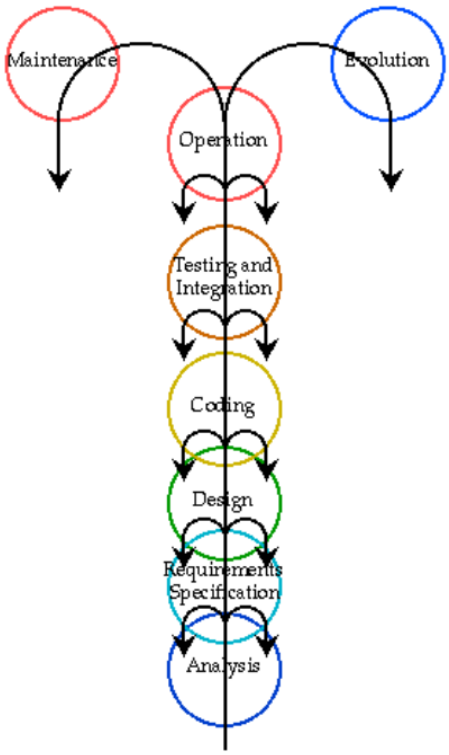
Es el modelo más sencillo posible, las fases son ejecutadas de forma secuencial.



Modelo en fuente: se basa en el modelo en cascada, pero observe que la secuencia siempre contiene ciclos. Refleja el hecho de que algunas fases no pueden comenzar antes que otras y que algunas de estas fases están intercaladas.



Modelo en espiral: este modelo fue sugerido en el año 1988. Las actividades se repiten y generan un incremento.



Modelo en fuente: se basa en el modelo en cascada, pero observe que la secuencia siempre contiene ciclos. Refleja el hecho de que algunas fases no pueden comenzar antes que otras y que algunas de estas fases están intercaladas.

Tipos de desarrollo

Desarrollo Basado En Prototipos

El desarrollo está basado en prototipos que parten en cierto modo de la base de "construya algo y vea si eso es lo que se pretende". Se pueden tratar de un proceso de desarrollo completo –Programación Exploratoria- o pueden ser simples bocetos anticipándose al ciclo de vida del proyecto o implementación, o incluso pueden ser parte de un abordaje evolutivo.

Desarrollo Interactivo E Incremental

El desarrollo interactivo defiende la construcción inicial de un pequeño pedazo de software, que va creciendo de forma gradual, ayudando a los involucrados en el proceso a descubrir lo más pronto posibles problemas o inconformidades antes de que puedan llevar al desastre al proyecto. Los procesos iterativos son los preferidos por los desarrolladores comerciales porque ofrecen el potencial de alcanzar los objetivos del proyecto para un cliente que no sabe cómo comunicar lo que él quiere.

Desarrollo Ágil

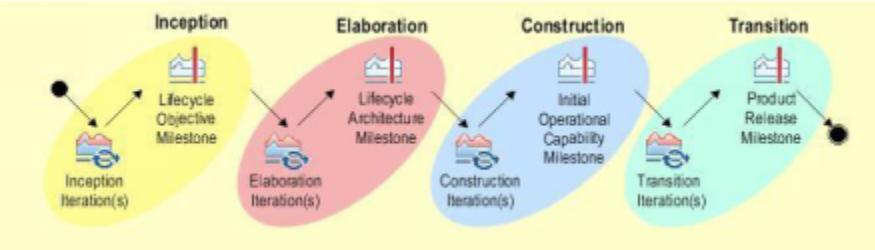
El desarrollo ágil de software defiende algunos puntos de vista en detrimento de otros:

•Individuos e interacciones X procesos y herramientas
•Un software funcionando X una documentación comprensible
•Colaboración con el cliente X negociación de contratos
•Respuesta al cambio X seguir un planteamiento

Los procesos ágiles utilizan el feedback, en lugar de la planificación, como su mecanismo de control primario. El feedback se orienta a través de pruebas y teste periódicos del software desarrollado.

11. Ciclo De Vida De Un Proceso De Desarrollo

Las fases del ciclo de vida de un proceso de desarrollo son: Concepción (iniciación), Elaboración, Construcción y Transición. A continuación, hablaremos de ellas con más detalle.



Fase de concepción:

El objetivo de esta fase es conseguir el entendimiento simultáneo entre todos los involucrados de los objetivos del ciclo de vida para el proyecto. Los objetivos son:

- Comprensión de lo que construir.
- Determinar la visión y el alcance de los sistemas y sus límites.
- Identificar quién está involucrado en el sistema y por qué
- Identificar las funcionalidades clave del sistema, decidir qué requisitos más críticos
- Determinar por lo menos una solución posible, identificando al menos una arquitectura candidata y su aplicación práctica
- Entender el coste, cronograma y los riesgos asociados al proyecto.

Los proyectos pueden tener una o más interacciones en la fase de concepción. Entre otras razones para tener múltiples interacciones podemos encontrarnos con que el proyecto es grande y su alcance de difícil definición, así como por tratarse de sistemas sin precedentes o con muchos

involucrados con necesidades que entran en conflicto y relaciones complejas. Otra posibilidad que nos podemos encontrar son grandes riesgos técnicos que demandan la creación de un prototipo o prueba de concepto.

Fase de elaboración

El propósito de esta fase es establecer una línea de base de la arquitectura del sistema y proporcionar una base estable para el volumen de esfuerzo de desarrollo en la próxima fase.

Los objetivos son:

- Obtener un entendimiento más detallado de los requisitos
- Proyectar, implementar, validar y establecer una línea de base para la arquitectura
- Mitigar los riesgos esenciales y producir un cronograma y una estimación de costes precisos.

El número de iteraciones en la fase de elaboración depende de, pero no está limitado por factores como el desarrollo de un nuevo producto versus ciclo de mantenimiento, sistema sin precedentes versus tecnología y arquitectura conocida, entre otros.

Habitualmente, en la primera iteración, se debe proyectar, implementar y probar un pequeño número de escenarios críticos para identificar qué tipo de mecanismos y arquitectura serán necesarios, una vez hecho esto es posible mitigar los riesgos más cruciales. También se deben detallar los requisitos de alto riesgo que deben ser resueltos anticipadamente en el proyecto. Se deben realizar pruebas suficientes para validar que los riesgos arquitecturales están mitigados.

En las siguientes iteraciones, se corrige todo aquello que no estaba bien en la iteración anterior. Se proyecta, implementan y prueban los escenarios arquitecturales significantes que quedarán garantizando que se identifican todas las áreas

principales del sistema (cobertura arquitectural) así los riesgos potenciales escondidos aparecen lo antes posible.

Fase de construcción

La finalidad de esta fase es determinar el desarrollo del sistema basado en la arquitectura colocada en la línea de base.

Los objetivos son:

- Desarrollar de forma iterativa un producto completo que esté listo para ser entregado a la comunidad de usuarios pronto.
- Describir los requisitos que restan, pre rellenando los detalles del proyecto, finalizando la implementación y la prueba del software, liberando la primera versión operativa (beta) del sistema y determinar si los usuarios ya están listos para que la aplicación pueda ser implantada
- Minimizar los costes de desarrollo y conseguir algún grado de paralelismo.
- Optimizar los recursos y aumentar el paralelismo de desarrollo entre los desarrolladores o los equipos de desarrollo, como, por ejemplo, atribuyendo los componentes que pueden ser desarrollados independientemente a desarrolladores diferentes.

Normalmente la fase de construcción tiene más iteraciones (de dos a cuatro) que las restantes fases dependiendo de los tipos de proyectos:

Proyecto simple: una iteración para construir el producto (para una liberación beta)
Proyecto más complejo: una iteración para exponer un sistema parcial y una para madurar para la prueba beta
Proyecto grande: tres o más iteraciones, dependiendo del tamaño del proyecto (cantidad de requisitos a implementar para una liberación beta).

Fase de transición

La finalidad de esta fase es asegurar que el software esté listo lo antes posible para ser entregado a los usuarios. Los objetivos de esta fase son: Ejecutar la prueba Beta para validar si las expectativas de los usuarios fueron atendidas. Estos normalmente requieren de algunas actividades de ajuste fino, tales como reparación de errores o mejoras en el rendimiento y la usabilidad; Obtener la concordancia de los involucrados de que la distribución está completa. Esto puede envolver varios niveles de pruebas para la aceptación del producto, incluyendo pruebas formales, informales y pruebas beta; Mejorar el desarrollo de proyectos futuros con las lecciones aprendidas. Documentar las lecciones aprendidas y mejorar el entorno de procesos y herramientas para el proyecto.

La fase de transición puede incluir la ejecución paralela de sistemas antiguos y nuevos, migración de datos, formación de usuarios y ajustes en los procesos de negocio. La cantidad de iteraciones en la fase de transición varía de una iteración para un sistema simple que necesita en primer lugar reparar pequeños errores, hasta muchas iteraciones para un sistema complejo, envolviendo la adición de características y la ejecución de actividades para hacer la transición, en el negocio, del uso del sistema antiguo al sistema nuevo.
Cuando los objetivos de la fase de transición han sido alcanzados el proyecto está listo para ser cerrado. Para algunos productos el fin del ciclo de vida actual del proyecto puede coincidir con el comienzo del ciclo de vida siguiente, conduciendo a la nueva generación del mismo producto.

Disciplinas

Las disciplinas son agrupamientos de tareas que comparten un mismo propósito. Existen varios tipos de disciplinas de las que hablaremos a continuación. para adaptarse continuamente a los cambios en los requisitos del negocio, en los requisitos de sistema y en las capacidades técnicas. La gestión de proyecto es una disciplina tipo paraguas que impacta y es impactada por todas las otras disciplinas.

Requisitos

Esta disciplina explica cómo licitar, analizar, especificar, validar y gestionar los requisitos para el sistema a ser desarrollado. El propósito de esta disciplina es: entender el problema a ser resuelto; entender las necesidades de los involucrados; definir los requisitos para la solución; definir los límites (alcance) del sistema; identificar las restricciones técnicas para el planeamiento de las iteraciones; proporcionar la base inicial para la estimación del coste y cronograma.
Para conseguir los objetivos es importante comprender la definición y el alcance del problema que estamos intentando resolver. Los involucrados son identificados y el problema a ser resuelto es definido. Concordando con el problema a ser resuelto, los requisitos para el sistema son provocados, organizados, analizados, validados y especificados. Durante todo el ciclo de vida, los cambios en los requisitos son gestionados.

Prueba

Esta disciplina explica cómo proporcionar un feedback sobre la madurez del sistema a través de la evaluación del proyecto, implementación, ejecución y de los resultados de las pruebas. El propósito de esta disciplina es: encontrar y documentar defectos; validar y probar las suposiciones hechas en el proyecto y requisitos especificados a través de demostraciones concretas; validar que el producto de software fue hecho como se proyectó; validar que los requisitos están apropiadamente implementados. Un esfuerzo de prueba correcta está basado en

la filosofía de pruebas breves y pruebas frecuentes. Como orientación: ¿Cómo se podría romper este software? ¿En qué posibles situaciones podría estar este software para trabajar de manera previsible?
Las pruebas desafían las suposiciones, riesgos y otras incertidumbres inherentes en el trabajo de otras disciplinas y trata esas preocupaciones y evaluación imparcial.

Desarrollar La Arquitectura

Desarrollar los requisitos arquitecturalmente significativos priorizados para la iteración. Refinar la arquitectura inicial de alto nivel en un software funcionando con el objetivo de producir un software estable que atienda los riesgos técnicos en el ámbito. Desarrollar la arquitectura, los arquitectos y desarrolladores trabajan juntos para refinar el boceto inicial de la arquitectura en elementos de proyecto concreto, garantizar que las decisiones arquitecturales son adecuadamente capturadas y comunicadas, garantizar que el equipo posee informaciones suficientes para permitir que el software sea desarrollado y garantizar que los requisitos que fueron priorizados para la iteración corriente fueran adecuadamente atendidos en el software.

Tareas A Realizar

- Refinar la arquitectura
- Implementar, evaluar y evolucionar con la arquitectura ejecutable.

Los pasos a seguir son: refinar las metas arquitecturales y los requisitos arquitecturalmente significantes

- Identificar elementos de proyecto arquitecturalmente significantes (componentes, clases y subsistemas)
- Refinar los mecanismos arquitecturales
- Definir el desarrollo y la prueba de la arquitectura
- Identificar oportunidades adicionales de reusó

- Validar la arquitectura
- Mapear el software en el hardware y comunicar las decisiones definiendo la arquitectura

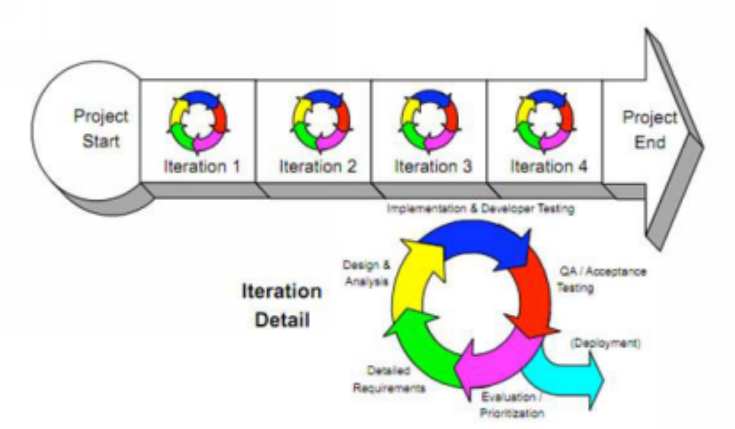
Razones para hacer en el inicio del proyecto un modelado ágil de la arquitectura:

•Aumentar la productividad
•Reducir el riesgo técnico
•Reducir el tiempo de desarrollo
•Mejorar la comunicación
•Subsidiar el desarrollo ágil de software
•Mejorar la organización del equipo.

12. Resultados

Como la finalidad de este proyecto es presentar una nueva metodología en desarrollo de software, es necesario hacer un recorrido por aquellas metodologías, que actualmente son de más uso en las empresas, con la finalidad de conocer su potencial.

12.1 Ciclo De Vida Del Scrum



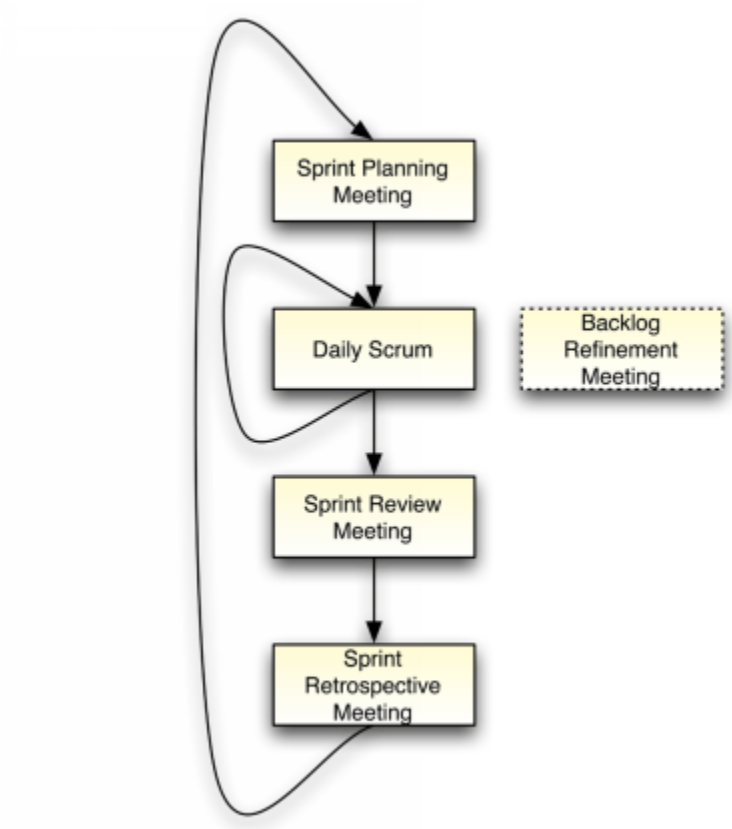
La estructura del framework consiste en grupos de Scrum conectados con los roles, eventos, los objetos, así como las reglas. Cada componente dentro de la estructura ofrece una función particular y es vital para el uso y el éxito de Scrum.

Scrum se basa en los conceptos empíricos de control de procedimientos o empirismo. El empirismo afirma que la comprensión proviene de la experiencia y de la elección basada en lo que se conoce. Además, hace uso de una técnica iterativa y también paso a paso para mejorar la previsión y el control de riesgos.
Las columnas de Scrum son la apertura, la evaluación y el ajuste. En cuanto a la transparencia, los elementos significativos del proceso tienen que ser visibles para los responsables de los resultados. visibles para los responsables de los resultados. Definir la norma común para que los

observadores compartan un entendimiento común de lo que se está viendo. Como ejemplo: un lenguaje común para referirse al proyecto y un significado común de "preparado". Cuando se trata de examinar, los elementos de Scrum necesitan ser inspeccionados y también proceder a la evaluación de las variaciones.

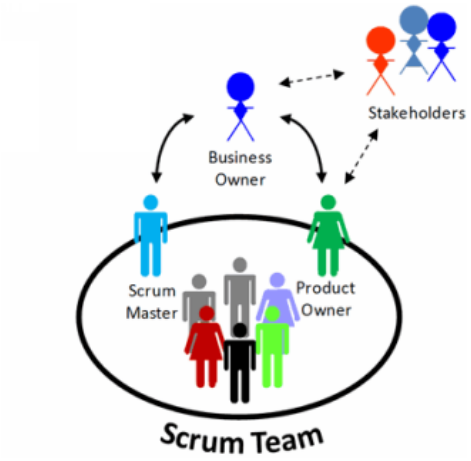
Ya en la personalización, esto supone que, si un inspector establece que una persona o más elementos de un procedimiento se desviaron fuera de los límites aceptables, el elemento resultante será inaceptable.

El artículo resultante será ciertamente inadecuado, el proceso o el producto en producción deberá ser reajustado.



12.1.1 El Equipo Del Scrum

Está formado por el gerente de proyecto, el equipo de desarrollo y el Scrum Master. Los equipos son auto-organizables y multifuncionales. El modelo de equipo en el Scrum es proyectado para perfeccionar la flexibilidad, creatividad y productividad.



El propietario de la tarea es responsable de aprovechar al máximo el valor del producto y también el trabajo del equipo de desarrollo. Es el único encargado de gestionar el stock de artículos. Entre sus funciones están: expresar claramente las cosas del stock de artículos; ordenar las cosas del stock para lograr de forma ideal los objetivos y las misiones; asegurarse de la valía del trabajo realizado por el equipo de desarrollo; asegurarse de que la cartera de productos es visible, transparente, clara para cada persona y también muestra que el equipo funcionará para seguirla; así como asegurarse de que el equipo de desarrollo comprende los productos de la cartera de productos al nivel ideal.

El equipo de desarrollo está compuesto por especialistas que realizan el trabajo de entregar una variación funcional que posiblemente para entregar una variación utilizable que posiblemente impulse el producto "todo preparado" al final de cada SLA.

"preparado" al final de cada Sprint. Sólo los participantes del equipo de desarrollo producen incrementos.

Entre sus características, el equipo de crecimiento es auto-organizado, así como interfuncional, teniendo todas las habilidades como un equipo para crear el incremento del producto. Scrum no define grupos para los miembros del equipo aparte del desarrollador; la responsabilidad pertenece al equipo en general. No contiene sub-equipos.

El Scrum Master es responsable de garantizar que el Scrum se entiende y también se utiliza; asegurando que el grupo pone el Scrum en actividad aplicada; asegurándose de que el equipo lleve a cabo el concepto, las técnicas, así como las normas de Scrum el concepto, las prácticas y las normas. El líder, trabaja tanto para el propietario del artículo como para el grupo de avance,

así como para el equipo de desarrollo.
En lo que respecta a su relación con el propietario del artículo, el Scrum Master trabaja para el propietario del producto mediante la búsqueda de estrategias para la supervisión eficaz de la pila de artículos, conectando claramente la visión, el objetivo y también los artículos de la pila con el equipo de crecimiento. Supervisa de mostrar el equipo para desarrollar elementos de backlog de artículos de una manera clara, así como sucinta, reconociendo la estrategia de producto duradera en el entorno empírico, además de la comprensión, así como el ejercicio de la destreza y la facilitación de eventos como se requiere o es necesario. En cuanto a su conexión con el grupo de desarrollo, supervisa de educar al equipo en la autogestión y también la interdisciplina. Muestra y lidera el grupo de desarrollo en la producción de artículos de alto valor, eliminando los desafíos para proceder y facilitando las ocasiones según sea necesario o requerido. Además, es responsable de educar al equipo de crecimiento en atmósferas empresariales donde Scrum no está totalmente asumido y comprendido.

12.1.2 Sprint

El corazón de Scrum es el Sprint, un período de tiempo de un mes o mucho menos, a lo largo de la cual un "todo listo", posiblemente útil variación paso a paso del producto, se produce. Los Sprint tienen periodos consistentes a lo largo del esfuerzo de crecimiento. Un nuevo Sprint se lanza después del veredicto del anterior.

Los Sprint se componen de una reunión de planificación del sprint, conferencias diarias, trabajo de avance, un testimonio del sprint y una retrospectiva del sprint. Durante el Sprint no se realizan cambios que puedan poner en peligro sus propósitos, los objetivos de calidad no se reducen y el alcance puede ser aclarado, así como renegociado entre el propietario del artículo y el grupo de avance a medida que se ha descubierto más.

Un Sprint puede terminarse antes de que se cumpla el plazo, y la única persona con autoridad para hacerlo es el propietario del producto. El único individuo con autoridad para hacerlo es el propietario del proyecto.

12.1.3 Conferencia De Preparación Del Sprint

El trabajo a realizar en el Sprint se planifica en la reunión de preparación del Sprint. Es un trabajo colectivo de todo el Grupo Scrum. La solución que se busca es lo que se puede proporcionar como resultado del siguiente incremento del Sprint o simplemente cómo se realizará el trabajo necesario para proporcionar el incremento.

12.1.4 Reunión Diaria

Se trata de una ocasión con una duración de 15 minutos. El tiempo de desarrollo debe integrar las tareas y también desarrollar un plan para las próximas 24h. Debe revisar el trabajo porque es la última reunión diaria y también prever el trabajo a realizar antes de la siguiente reunión diaria. Las preguntas a las que hay que responder son qué hice ayer que ayudó al grupo de crecimiento a alcanzar el objetivo del Sprint, qué haré hoy para ayudar al grupo de desarrollo, veo algún tipo de desafío que me impida a mí o al equipo de desarrollo alcanzar el objetivo del Sprint.

12.1.5 Revisión Del Sprint

La evaluación del Sprint se ejecuta al final del Sprint para inspeccionar el incremento y también ajustar el backlog del producto si es necesario. Es una conferencia casual, no una conferencia permanente, y la discusión del incremento está destinada a inspirar y obtener respuestas, así como anunciar la cooperación.
Los individuos son el grupo Scrum y también las partes interesadas. El propietario del artículo aclara
El propietario del artículo aclara qué productos de la pila de productos están "preparados" y cuáles no. El grupo de crecimiento revisa lo que ha funcionado a lo largo del Sprint y muestra el trabajo que está "preparado". El propietario del trabajo habla de la pila de productos tal y como está. Todo el grupo colabora en lo que debe hacer a continuación. Se analiza el calendario, el plan presupuestario, las capacidades potenciales y también el mercado para el siguiente

lanzamiento previsto del producto.

12.1.6 Retrospectiva Del Sprint

Ocurre después de la evaluación del Sprint y también antes de la conferencia de preparación para el siguiente Sprint. La duración de la caja de tiempo tiene que ver con 3 horas para un Sprint de un mes. El Scrum Master participa en la conferencia como empleado de apoyo debido a su responsabilidad en el proceso Scrum.
La retrospectiva del Sprint es una oportunidad para que el equipo se revise a sí mismo y desarrolle un plan de renovaciones para el siguiente Sprint.

12.2 Gestión ágil de proyectos

Este método fue creado por Jim Highsmith en 1999 a través de su publicación "Adaptive Software application Growth: A Collaborative Strategy to Managing Complex Equipments". En el libro, Jim dice que el proceso de avance debe estar alineado con los propósitos, para asegurar que, si los objetivos son repetibles y pueden ser totalmente predefinidos, un procedimiento autoritario sería ideal, pero si los propósitos de negocio son ajustables, y también de vanguardia en la naturaleza, después de que el marco debe ser ágil, con un alto grado de adaptabilidad y versatilidad.

12.3 Test Driven Development (TDD)

También conocida como “desarrollo orientado a pruebas” sería, sin duda similar a utilizar Extreme Programming. Esta técnica, por así decirlo, se da en el proceso de vida de la metodología XP, a lo largo de las fases de versión, producción y también mantenimiento. Al ser una estrategia orientada al avance de los trabajos con proyección, así como por su importancia, se considera actualmente un método ajeno a XP. Como señala Carvajal (Carvajal Riola 2008), en esta forma de establecer tareas se puede pensar en un enfoque, ya que ofrece requisitos y también tipos de avance, que condicionan el avance del trabajo, potenciando su calidad. Se puede utilizar individualmente o junto con otros enfoques como SCRUM, FDD o cascada. El procedimiento

contiene la repetición de 3 procedimientos que se duplican dentro de este enfoque:

1. Detallar las pruebas de la actuación a implementar.
2. Codificar la funcionalidad hasta que pase la prueba.
3. Recodificar ambos códigos, para estructurarlo. Al hacer esto, se itera en un bucle, construyendo las prestaciones del sistema a establecer.

Ilustración Test Driven Development (TDD)

12.4 Extreme Programming (XP).

Extreme Programs fue establecido por Kent Beck en 2000, y recomendado en la guía "Extreme Shows Explained: Embrace Adjustment" (Beck 2000), que se centra en el establecimiento de relaciones entre las personas como método para lograr el éxito en cualquier tipo de desarrollo. El XP busca la colaboración entre los diferentes miembros del equipo, bordeado por un gran ambiente de trabajo. XP se basa en las respuestas constantes entre los clientes, así como los desarrolladores, allanando el camino a la comunicación fluida entre los miembros, buscando la simplicidad de los remedios y la decisión a pesar de los ajustes. Su nombre está relacionado con los conceptos que defiende, utilizando el sentido común, pero de forma severa. Esta metodología es realmente útil en tareas en las que los requisitos pueden ser imprecisos, variables y también en

las que las amenazas son elevadas.

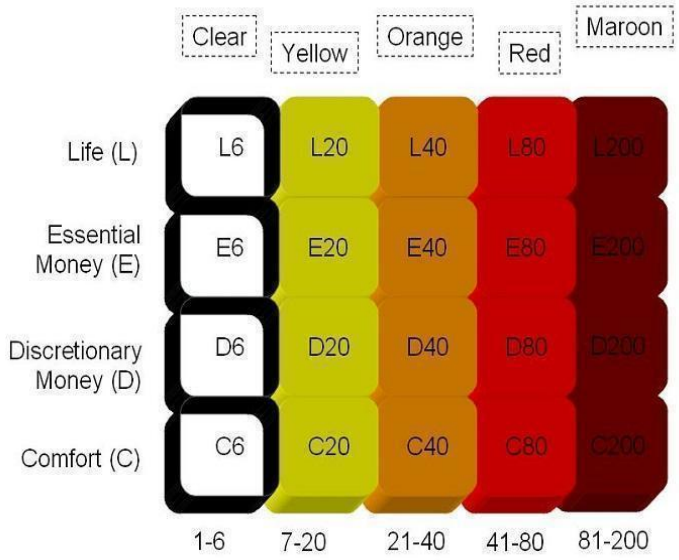
Extreme Programming (XP).

Es potencialmente el método de destreza mejor entendido y más comúnmente utilizado. El nombre XP fue acuñado por Beck (2000), ya que la técnica se desarrolló utilizando las mejores prácticas de crecimiento repetitivo y también con la participación extrema de los consumidores. El Extreme Programming (XP), que algunos piensan que es una metodología extraordinaria, así como otros creen que es negativa (Rakitin, 2001). En este metodo todas las necesidades se expresan como circunstancias (llamadas historias de usuario), que se aplican directamente como una serie de trabajos. Los programadores operan en parejas y desarrollan exámenes para todas y cada una de las tareas antes de crear el código. Todos los exámenes deben ejecutarse de forma eficiente cuando el nuevo código se integre en el sistema.

12.5 Metodología del cristal

Método basado en la categoría de la complejidad de una tarea, según una serie de colores, para asegurar que cuanto más oscuro sea el tono, más complicada será la tarea. Este método también recomienda utilizar un color por proyecto en función de la dimensión del trabajo y su criticidad. Por ello, no existe un requisito básico de Crystal, sino que hay una aplicación particular de la técnica para cada trabajo en el que se ha utilizado cada tipo de tarea.

Metodología basada en la clasificación de la complejidad de un proyecto, en función de una serie de colores, de forma que cuanto más oscuro más complejo será el proyecto. Este método, además propone utilizar un color por proyecto dependiendo del tamaño de este y de su criticidad.



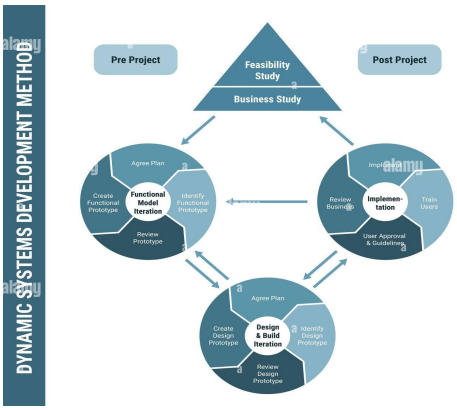
Metodología del cristal.

12.6 Método de Desarrollo de Sistemas Dinámicos (DSDM)

Este método se centró exclusivamente en la satisfacción de las necesidades del individuo. Este sistema fue lanzado en febrero de 1995, como un método genérico para individuos, procesos y también dispositivos. El Consorcio DSDM es el propietario y administrador del enfoque DSDM y sólo sus participantes pueden utilizarlo. Para DSDM, el desarrollo es visto como repetitivo, así como paso a paso, en el que el cliente debe existir constantemente, y en el que hay que planificar las modificaciones, siendo la única metodología que cubre todo el proceso de vida del trabajo, incorporando técnicas de gestión de proyectos, y otras técnicas para asegurar la rentabilidad de la tarea, según la solución prevista.

Las principales características de DSDM son:

- Banda ancha de crecimiento, adaptabilidad, así como entrega de artículos sin disminuir la tecnología utilizada.
- Está pensada como un RAD (Fast Application Advancement) siendo perfecta para proyectos cuyos márgenes de tiempo, así como de gasto son marginales.
- Esta técnica se divide en 5 fases, siendo las 3 últimas repetitivas: estudio de conveniencia, investigación de la empresa, modelado funcional, estilo y construcción, y también finalmente ejecución.



Método de Desarrollo de Sistemas Dinámicos (DSDM)

13. MÉTODO SIMPLIFICADO EN EL DESARROLLO DE SOFTWARE (MSDS)

Después de revisar las metodologías existentes, se propone una nueva metodología a la que llamaremos: MÉTODO SIMPLIFICADO EN EL DESARROLLO DE SOFTWARE (MSDS), la cual consta de una serie de actividades generalizadas.

Este método es adaptable y fiable la principal finalidad es mejorar el flujo de tareas en el desarrollo del proyecto, visualizar, optimizar y la administrar los roles es la función de las fases que se desarrollaran a continuación.

13.1 Fases de la metodología

Visión

En esta fase el grupo y también el cliente especifican las demandas del trabajo requerido y los objetivos generales del proyecto. La etapa culmina con el hito autorizado de Visión y Alcance. El "imaginario" del trabajo, es donde todo el grupo tendrá ciertamente un concepto claro, aunque general, de los objetivos, la infraestructura, innovación, la arquitectura, deberes y también los riesgos, todo ello necesario para el correcto desarrollo del proyecto.

Actividades claves:

- Definir el proyecto especificando los objetivos, el alcance, las limitaciones y las suposiciones
- Elaborar una interpretación de las demandas definiendo lo que deben hacer los nuevos servicios web.
- Establecer un diseño teórico de los servicios.
- Evaluar los riesgos a alto nivel del proyecto.
- Definir la estructura del equipo de trabajo.

Preparación

Durante la fase de preparación, el equipo elabora un proyecto de plan maestro de tareas, así como un calendario de tareas y también una especificación funcional de las mismas. Esta fase culmina con el hito de la estrategia de tareas aprobadas. Se plantean las necesidades específicas del cliente, teniendo en cuenta que el método simplificado no está cerrado, es decir, permite realizar ajustes dentro del trabajo, también en la fase de avance.
Actividades claves:

- Recoger información sobre los servicios de Internet.
- Evaluar las fuentes necesarias para finalizar la tarea.
- Construir el plan maestro de las tareas programadas en la primera etapa.
- Redactar la rutina de la tarea.

Avance en el desarrollo

Esta fase incluye una colección de lanzamientos internos del producto, creados partes para determinar su desarrollo, así como para asegurarse de que todos sus módulos o componentes están sincronizados y también pueden integrarse. La etapa finaliza con el hito Rango Completo. Produce efectivamente el código necesario para crear un producto funcional para el cliente.

Actividades claves:

- Verificar el diseño físico replicando el ambiente del servidor web y también realizando pruebas de dispositivo, integración y aplicación.
- Construir el sistema, configurando y situando los servidores web de Internet de producción que se utilizarán en su red.

Estabilidad del proyecto

Esta fase se centra en la evaluación del producto. El proceso de pruebas pone de manifiesto el uso y el procedimiento del artículo en problemas de entorno real. La etapa culmina con el hito

autorizado de preparación para el lanzamiento.

Actividades claves:

- Reconocimiento del estilo.
- Llevar a cabo un reconocimiento piloto, así como un despliegue controlado para presentar las nuevas soluciones a un conjunto específico de usuarios a pequeña o mediana escala.

Implementación

En esta fase el grupo lleva a cabo la innovación y también los componentes utilizados por el remedio, apoya la implementación, apoya el procedimiento y el cambio del trabajo, y adquiere la aprobación final del consumidor. La fase finaliza con el hito de la Implementación total.

Actividades claves:

- Completar la formación del gestor y del cliente.
- Poner en marcha el nuevo sistema, evaluar la eficacia y remediar cualquier tipo de problema.
- Supervisar el sistema y planificar las renovaciones.

MÉTODO SIMPLIFICADO EN EL DESARROLLO DE SOFTWARE.

14. Evolución de metodologías para el desarrollo de software.

Programación	Año	Métodos	Descripción
Programación lineal	40´s	Diagramas de flujo	Todo el programa en un solo bloque, con ejecución secuencial de instrucciones. Eran los tiempos del ensamblador, las capacidades reducidas y la necesidad de optimizar al máximo.
Programación estructurada	70´s	Análisis descendente (Top-Down)	Programa dividido en procedimientos: distintos “bloques” que se van llamando según se necesiten. Se confecciona “de arriba abajo”, pensando primero en funcionalidades generales, a grandes rasgos. Para irlas refinando poco a poco, hasta llegar a detallar cada uno de los procedimientos y su interacción.
Programación orientada a objeto (OOP)	80´s	Programa dividido en clases (objetos)	Dentro de las cuales van encapsuladas las propiedades(variables) y los procedimientos (operaciones). Algunas de esas variables y operaciones son de uso restringido al propio objeto, solo pueden ser llamadas desde el interior. Y otras son de uso público. Esto nos proporciona dos grandes ventajas: evitar interferencias extrañas dentro distintas partes del programa y podemos cambiar la implementación concreta de un objeto sin afectar al resto del sistema.
Metodologías Ágiles	90´s	Formatos -metodologías ágiles	Nacidas para dar respuesta al entorno siempre cambiante y en rápida evolución en que se han de desarrollar programas informáticos. En lugar de hacer proyectos monolíticos perfectamente estructurados en fases, una detrás de otra. Se intenta trabajar en ciclos cortos como mini-proyectos que implementan una parte de las funcionalidades, pero sin perder el rumbo general del proyecto global.
VFSM (Virtual Finite State Machine)	Finales de los 90´s	Formatos -metodologías ágiles	Intenta abstraer la arquitectura real Metodología Ágiles sobre la que se implementarán los programas. Para intentar aplicar métodos matemáticos genéricos que nos permitan simular su comportamiento. Y poder así diseñar con garantías de cuál va a ser su funcionamiento. Aplica la tecnología de las Máquinas Virtuales. Y nació en el entorno de los sistemas embebidos.

15. ¿PORQUE NO HAY UNA METODOLOGÍA GENERAL PARA TODOS LOS PROYECTOS?

El hecho es que cada uno de los métodos señalados no es más que una teoría, una simplificación para aclarar lo que realmente sucede o una punta de lo que debe tener lugar. En general, son sólo estimaciones a la verdad, basadas en presunciones sobre los tipos de problemas que se encuentran con frecuencia, sobre las suposiciones de los clientes, sobre los recursos disponibles, las fechas de entrega, los dispositivos, la complejidad de las tareas, etc. Ninguna de ellas es 100% efectiva en todas las situaciones específicas. Por lo tanto, no existe un servicio mágico que asegure siempre que el crecimiento de los grandes sistemas sea fácil. No olvide que... Hay que entender al cliente. Por lo general, el cliente aclara lo que necesita de forma excesiva o inferior a la auténtica realidad.

El programador debe evitar que se le confíe una idea sencilla del proyecto, tiene que comprender toda su complejidad, aunque a veces no sea evidente. El programador y también el analista deben tener una buena comprensión de lo que se quiere decir, ambos deben entender cuál será el trabajo final y lo que espera el cliente. Los pasos a seguir deben ser convenientemente fáciles de entender y el resultado obtenido tiene que existir de forma razonable, sin producir falsas expectativas o creencias erróneas, documentar todo el procedimiento es importante. Hay que reconocer el marco en el que funcionará el programa informático, hay que definir la forma de facturar al cliente, hay que entender el soporte que se ofrecerá y hay que reconocer los requisitos reales del cliente.

16. Conclusiones

No existe una metodología global para abordar eficazmente cualquier tipo de tarea de desarrollo de aplicaciones informáticas. Cualquier tipo de metodología tiene que adaptarse al contexto del trabajo (recursos tecnológicos y también humanos, tiempo de crecimiento, tipo de sistema, etc.). Históricamente, las técnicas típicas han intentado resolver el mayor número posible de situaciones del contexto del trabajo, necesitando un esfuerzo considerable para adaptarse, especialmente en trabajos pequeños y con demandas muy transformadoras. Los métodos de destreza ofrecen una opción prácticamente a medida para muchos trabajos con estas características. Una de las cualidades más excepcionales de un método es su sencillez, tanto en su descubrimiento como en su aplicación, lo que disminuye los gastos de implementación en un grupo de crecimiento. Sin embargo, su aplicación tiene una serie de inconvenientes y limitaciones, como por ejemplo: se dirigen a grupos pequeños o medianos (Beck sugiere que el tamaño del equipo debe limitarse a entre 3 y 20 personas como máximo, otros dicen que no más de 10 personas), el entorno físico debe ser uno que permita la interacción y la asociación entre todos los miembros del equipo en todo momento, cualquier tipo de resistencia del cliente o del grupo de desarrollo a las prácticas y también a los principios puede llevar el proceso al fracaso (el lugar de trabajo, la colaboración y la conexión legal son clave), el uso de tecnologías que no tienen un ciclo de comentarios rápido o no sostienen rápidamente la adaptación, etc.

17. Referencias Bibliográficas

- Biblioteca Digital Santander. (2019). Metodologías desarrollo de software. 26 de agosto del 2022, de Becas Santander.
- Amador Duran Toro, Beatriz Bernárdez Jiménez. (2015). Metodología para el análisis de requisitos de sistemas de software versión 2.2. Monterrey, México: Fachenda.
- Universidad Católica de Argentina. (2015). Metodologías de desarrollo de software. 26 de agosto de 2022, de Universidad Católica de Argentina. Sitio web: https://uca.edu.ar/es
- Code Complete (2ª edición) por Steven McConell (2017) editorial Nuevo milenio, 26 agosto 2022
- Las metodologías en el desarrollo enfocado (2020), Universitat Carlemany, País de Andorra.
- Metodologías ágiles (2017), Biblioteca digital IEB School.
- Metodologías ágiles vs tradicionales, revista empresarial IBM (2019)
- Página de Microsoft: MSDN en español, “Métodos Heterodoxos en Desarrollo de Software”,
- Letelier P., Penadés C., “Metodologías ágiles para el desarrollo de Software: eXtreme Programming (XP)”, Universidad Politécnica de Valencia.

Sitios web

- http://www.proyectosagiles.org/
- http://es.wikipedia.org/wiki/Desarrollo_%C3%A1gil_de_software
- http://www.enterprisexp.org
- www.iso.org
- http://ingenieriadesoftware.mex.tl/61162_FDD.html
- https://www.ecured.cu/Metodologias_de_desarrollo_de_Software
- https://www.becas-santander.com/es/index.html