**Cuestionario Nº 1**

1. **Metodologías más usadas actualmente en la industria del software.**

**Metodologías tradicionales.**

Inicialmente, el desarrollo de software era totalmente manual, existía una fuerte necesidad de mejorar los procesos y lograr las metas deseadas, el diseño y las bases de las metodologías existentes en otras áreas debían ser importadas y adaptadas para el desarrollo de software. Esta nueva fase de adaptación consistió en un desarrollo dividido en fases sucesivas, que en cierta medida mejoraron la demanda subyacente en el sector del software.

Según Pérez (2011) imponen una planificación rígida y meticulosa del proyecto, soportada por herramientas y una carga de trabajo pesada en planificación, diseño y documentación, en un afán por hacer al desarrollo predecible dentro de un marco de temporalidad y costo; recordando que derivan de solventar los problemas de la crisis del software vivida décadas atrás.

Sin embargo una desventaja importante de este enfoque es el alto costo de implementar el cambio y no proporciona una buena solución para proyectos con entornos volátiles.

***Cuadro 1. Metodologías Tradicionales.***

|  |  |
| --- | --- |
| **Metodologías tradicionales** | |
| Rapid Application Development (RAD) | Su traducción al español, desarrollo rápido de aplicaciones, presentado por James Martin en 1980. La metodología se concentra en una lista de tareas y una estructura basada en el desglose de las tareas orientada en la rapidez. Su propósito fue la de cubrir la necesidad de agilizar las entregas de aplicaciones.  Comprende el desarrollo interactivo, la construcción de prototipos y el uso de utilidades CASE (Computer Aided Software Engineering). Tradicionalmente, el desarrollo rápido de aplicaciones tiende a englobar también la usabilidad, utilidad y la rapidez de ejecución.  Hoy en día se suele utilizar para referirnos al desarrollo rápido de interfaces gráficas de usuario tales como Glade, o entornos de desarrollo integrado completos, algunas de las plataformas más conocidas son Visual Studio, Lazarus, etc. |
| Racional Unified Process (RUP) | Propuesta en 1998 por Ivar Jacobson, Grady Booch y James Rumbaugh.  Metodología basada en los modelos en Cascada y por Componentes. Esta metodología cuenta con algunas características que la hacen especial, como “casos de uso”, la cual es la descripción del servicio que el usuario requiere del sistema y la secuencia de iteraciones usuario-sistema, además se centra en la arquitectura, dicta pautas específicas para la constitución del equipo y las escalas de tiempo, es iterativa e incremental.  Es una de las más usadas para el análisis, desarrollo y documentación de sistemas orientados a objetos, muy aplicada en proyectos de gran complejidad y magnitud con apoyo de equipos expertos. |
| Microsoft Solution Framework (MSF) | Introducida por primera vez en 1994 como un conjunto de las mejores prácticas (principios, modelos, disciplinas, conceptos y directrices) en los desarrollos de Software de Microsoft y Microsoft Consulting Service. Es flexible, permite aplicar de manera individual e independiente cada uno de sus componentes, es escalable según la magnitud del proyecto, además está fundamentada en los modelos espiral y cascada.  Profesa la aplicación de principios fundamentales tanto para una organización eficiente de trabajo, esquemas o modelos para la organización de los equipos y disciplinas de gestión. |

**Metodologías ágiles.**

Al implementar metodologías clásicas o tradicionales en proyectos de mediano tamaño con mayores requerimientos de tiempo de respuesta y requerimientos imprecisos y variables, los resultados obtenidos son ineficaces, por lo que se invierte más tiempo en pensar en el diseño, controlar y afrontar cualquier cambio de especificaciones haciendo así un proceso totalmente ineficiente e improductivo.

Siendo este el problema, nacen las metodologías ágiles, enfocadas prioritariamente en el software y no en la documentación o arquitectura, dando así espacio a entregas funcionales y tempranas del software sin verse afectados por la condición de requerimientos cambiantes.

Aplicando este tipo de metodologías se consigue que ante los cambios que puedan presentarse los costos sean reducibles, además se prioriza la colaboración del cliente con el equipo incluso por encima de un contrato.

***Cuadro 2. Metodologías ágiles***

|  |  |
| --- | --- |
| **Metodologías ágiles** | |
| Extreme Programming (XP) | Busca guiar a equipos de desarrollo de software pequeños, entre dos y diez desarrolladores, en ambientes de requerimientos imprecisos o cambiantes. Consta con 5 valores que fundamentan a sus principios: Comunicación, Retroalimentación, Simplicidad, Respeto y Coraje. Gracias estos valores y fundamentos es que se derivan las distintas “actividades XP” como son, el diseño más simple posible, aceptar cambios que mejoren la estructura del sistema, etc. |
| Scrum | Utiliza un enfoque incremental que se basa en la teoría del control empírico. Los equipos que trabajan con este tipo de metodología son catalogados como auto gestionados, multifuncionales y trabajan en iteraciones |
| Metodología Espiral | Proporciona un énfasis en un área clave con análisis de riesgo iterativo deliberativo. Además se adapta a sistemas complejos a gran escala, que presenta prototipos que ayudan a medir dicho riesgo al que se van a enfrentar, además de aprovechar los conceptos de arriba hacia abajo y de abajo hacia arriba. La espiral se elige generalmente sobre el enfoque de Cascada para proyectos grandes, costosos y complicados.  El proyecto en espiral comienza con algo pequeño, explora los riesgos, desarrolla un plan para gestionarlos y luego decide si dar el siguiente paso del proyecto. |

***Metodologías tradicionales vs Metodologías ágiles***

A continuación se presentara un cuadro donde se compara ambas metodologías.

***Cuadro 3. Metodologías tradicionales vs Metodologías ágiles***

|  |  |
| --- | --- |
| ***Metodologías tradicionales*** | ***Metodologías ágiles*** |
| Se aferran a las normas provenientes de los estándares planteados en el entorno de desarrollo | Se basan en el empirismo obtenido por la práctica de la producción del código |
| Se resiste a los cambios | Es flexible para cambios durante el desarrollo |
| Los costos son altos si presentan alguna inconsistencia | Se pueden reducir costos por cambios inesperados |
| Son pocos los roles | Gran cantidad de roles |
| Documentación exhaustiva | La documentación no es la prioridad |
| Se entrega el software al terminar el desarrollo | Entregas constantes de los prototipos del software a desarrollar |
| El cliente no forma parte del proyecto si no hay una reunión de por medio. | Existe comunicación constante e interactiva con el cliente. |
| Es aplicable para proyectos de cualquier tamaño sin embargo donde mayor uso se los da son en proyectos grandes. | Suele ser implementado para trabajos pequeños que se desarrollen en el mismo lugar. |
| La estructura del software es prioridad y se define tempranamente en el proyecto. | La énfasis en la estructura del software es menor, sin embargo se va definiendo y mejorando durante el desarrollo del proyecto. |

**Bibliografia y Webgrafía:**

Información extraída de: <https://invidgroup.com/es/las-mejores-metodologias-de-desarrollo-de-software-en-2019/>

Gamboa, J. Z. (2018). Evolución de las Metodologías y Modelos utilizados en el Desarrollo de Software. *INNOVA Research Journal*, *3*(10), 20-33.

Figueroa, R. G., Solís, C. J., & Cabrera, A. A. (2008). Metodologías tradicionales vs. Metodologías ágiles. *Universidad Técnica Particular de Loja, Escuela de Ciencias de la Computación*, *9*, 1-10.

Maida, E. G., & Pacienzia, J. (2015). Metodologías de desarrollo de software.

Montero, B. M., Cevallos, H. V., & Cuesta, J. D. (2018). Metodologías ágiles frente a las tradicionales en el proceso de desarrollo de software. *Espirales revista multidisciplinaria de investigación*, *2*(17).