# Selección metodología

## Cuadro comparativo de las metodologías

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Modelo de ciclo de vida cascada** | **Modelo de ciclo de vida en V** | **Modelo de ciclo de vida iterativo** | **Modelo de ciclo de vida incremental** | **Modelo de ciclo de vida espiral** |
| **Definición** | Es el enfoque metodológico que ordena las etapas del proceso para el desarrollo de software, de tal forma que al finalizar la primera etapa se inicia la segunda etapa y así consecutivamente. | Es un proceso de pasos en el desarrollo del ciclo de vida de un proyecto de software, en esta se deben producirse las actividades y resultados que deben ocurrir durante el proceso del desarrollo de este. | Es un proceso que permite al desarrollador sacar ventaja de lo que se ha aprendido a lo largo del desarrollo anterior.  Incrementando y actualizando cada vez que se avanza en el proyecto. | Es un proceso de crecimiento progresivo de la funcionalidad, o sea que el producto va evolucionando con cada una de las entregas previstas. | Es un proceso de ciclos que se repiten. cada ciclo tiene las mismas fases y cuando termina da un producto ampliado con respecto al ciclo anterior. |
| **Características** | -Consiste en la ejecución secuencial de una serie de fases que se suceden, lo que da nombre al modelo.  -Cada fase genera documentación la cual deberá estar aprobada para continuar con la siguiente fase.  -Una fase no comienza hasta que la anterior ha terminado. | -Mejora la garantía de calidad  -Reduce los gastos totales durante todo el proyecto. | -Se puede determinar la calidad de los procesos y del producto a medida que se vaya actualizando.  -Permite el entendimiento de futuros problemas que puedan surgir para aprender y mejorar el proceso. | -Se evitan proyectos largos y se entrega a los usuarios con cierta frecuencia.  -El usuario se involucra más en el proyecto.  -Requiere gestores experimentados. | -Tiene una etapa en donde analizan los riesgos.  -Este es el enfoque más realista actualmente. |
| **Etapas**  **o**  **Fases** | -Requisitos  -Diseño  -Implementación  -Pruebas  -Mantenimiento | -Análisis  -Especificación  -Diseño  -Programación  -Prueba  -Documentación  -Mantenimiento  -Reingeniería | -Comunicación  -Planeación  -Modelado (análisis y diseño)  -Construcción  -Despliegue  (entrega, retroalimentación) | -Requerimientos  -Definición de las tareas y las iteraciones  -Diseño de los incrementos.  -Desarrollo del incremento.  -Integración de incrementos.  -Entrega del producto. | -Determinar objetivos.  -Análisis del riesgo.  -Desarrollar y probar.  -Planificación. |
| **Esquema** |  |  |  |  |  |
| **Aplicación o implementación** | Es utilizado en situaciones en las que el proyecto a realizar es simple, y en las que el desarrollo de estas es bien conocido. | Es utilizado en pruebas unitarias o pruebas de integración |  | Es utilizado mediante un calendario con plazos de entrega. | Para el desarrollo de software con diferentes versiones. |
| **Objetivos** | El poder organizar y documentar de forma ordenada y secuencial. |  | El poder realizar un Feedback por cada aplicación de requerimientos, obtener un aprendizaje y un conocimiento más avanzado. | El contacto directo con el cliente, se va evolucionando el software. | Es parecido al modelo incremental, aquí se toma en cuenta el concepto de riesgo. |
| **modo de identificación de problemas** | Mediante la comunicación | Modelado de los requerimientos | Mediante la comunicación y por los requerimientos. | Mediante la comunicación | Mediante la comunicación |
| **Determinación de requerimientos** | Mediante la comunicación | Mediante la comunicación y el modelado de requerimientos | Mediante la comunicación | Mediante la comunicación | Mediante la comunicación |

|  |  |
| --- | --- |
|  | **Modelo de ciclo de vida prototipos** |
| **Descripción** | Este modelo tipo prototipado debe ser construido en poco tiempo. Se identifican los requisitos, se planea con rapidez una iteración de prototipos y se presenta el modelado. |
| **Características** | * Este modelo es una aplicación funcional * Los prototipos se crean con rapidez. * Los prototipos van evolucionando a través de un proceso. |
| **Etapas**  **o**  **Fases** | * Modelado, diseño rápido * Construcción del prototipo * Desarrollo, entrega y retroalimentación * Comunicación |
| **Esquema** |  |
| **Aplicación o implementación** |  |
| **Objetivos** | la finalidad de este modelo tipo prototipo es probar varias veces el sistema mediante analistas y usuarios. |
| **Modo de identificación de problemas** | Mediante la comunicación |
| **Determinación de requerimientos** | Mediante la planeación |

## Ventajas y desventajas de los modelos tradicionales

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Modelo Ciclo de vida**  **Casada** | **Modelo ciclo de vida en**  **V** | **Modelo ciclo de vida Iterativo** | **Modelo ciclo de vida Incremental** | **Modelo ciclo de vida**  **Espiral** | **Modelo ciclo de vida Prototipos** |
| Los conflictos, bugs y errores de programación a veces conducen a un aumento de los costes y a una cantidad considerable de tiempo. Lo mismo se aplica si los clientes no están satisfechos. | El modelo en cuatro niveles puede ser demasiado simple para mapear todo el proceso de desarrollo desde el punto de vista de los desarrolladores | Infunde responsabilidad en el equipo de desarrollo al trabajar directamente con el cliente, requiriendo de profesionales sobre el promedio. | Requiere de metas claras para conocer el estado del proyecto. | Requiere experiencia en la identificación de riesgos | Surgen cambios imprevistos que retrasan el progreso del prototipo. |
| Las especificaciones que se hacen inicialmente son a menudo difíciles de entender para los clientes porque son más abstractas de lo que se supone que el software debe hacer. Especialmente en proyectos subcontratados, esto puede ser una desventaja decisiva, ya que la fecha de lanzamiento debe posponerse y el mercado puede haber cambiado durante este tiempo. | Está sobre todo centrado en la gestión de proyectos. | Requiere de un cliente involucrado durante todo el curso del proyecto. Hay clientes que simplemente no estarán dispuestos a invertir el tiempo necesario. | El modelo Incremental no es recomendable para casos de sistemas de tiempo real, de alto nivel de seguridad, de procesamiento distribuido, y/o de alto índice de riesgos. | Genera mucho tiempo en el desarrollo del sistema. | El desarrollador y el cliente tienen poca comunicación al inicio del proceso. |
| La entrega del software lleva más tiempo porque los departamentos no trabajan simultáneamente y cada fase sólo puede comenzar cuando se ha completado la fase anterior. | Su estructura relativamente rígida permite una respuesta poco flexible a los cambios durante el desarrollo, y, por lo tanto, promueve un curso lineal del proyecto. | El trato con el cliente debe basarse en principios éticos y colaboración mutua, más que trabajar cada parte independientemente, defendiendo sólo su propio beneficio. | Requiere de mucha planeación, tanto administrativa como técnica. | Debido a su elevada complejidad no se aconseja utilizarlo en pequeños sistemas. | Adoptarlo como el sistema final: Los usuarios y profesionales de sistemas pueden considerar al prototipo como el sistema final cuando aún es incompleto e inadecuado. |
| En ocasiones solo se puede detectar un fallo una vez terminado el proyecto, lo cual genera retrasos de tiempo considerable. | Las pruebas pueden ser caras y, a veces, no lo suficientemente efectivas. | La entrega de un programa que es parcial pero funcional puede hacer vulnerable al programa debido a la falta de robustez en su sistema, provocando que agentes ajenos puedan interferir con el correcto funcionamiento del programa en sí. | Demora cuando el cliente quiere entrega rápida o parciales. | Resulta difícil convencer a grandes clientes de que el enfoque evolutivo es controlable. | Administración difícil: Dicha dificultad radica en manejar el prototipo como un proyecto dentro del Ciclo de Desarrollo de Sistema sin perder de vista cuál era su propósito. |
| Normalmente, al realizar un proyecto de gran envergadura o de varios niveles, estos no permiten la división en fases del proyecto con claridad y fácil comprensión para los involucrados. | El cliente debe tener paciencia pues obtendrá el producto al final del ciclo de vida. | Los incrementos no deben constar de muchas líneas de código ya que la idea de los incrementos es agregar accesorios al programa principal (o funcional), para que este tenga una y mil formas de desenvolverse en su tarea; llenar los incrementos de muchas líneas de código provocaría que se perdiera la objetividad o base de lo que se trata el desarrollo incremental. | Orientado a ciertos tipos de clientes-usuario | Modelo costoso | El desarrollador puede caer en la tentación de ampliar el prototipo para construir el sistema final sin tener en cuenta los compromisos de calidad y mantenimiento que tiene con el cliente. |
| Debido a la estructura lógica del modelo, a menudo se pueden evitar errores conceptuales. | Optimización de la comunicación entre las partes involucradas a través de términos y responsabilidades claramente definidos. | Los productos desarrollados con este modelo tienen una menor probabilidad de fallar. | Con un paradigma incremental se reduce el tiempo de desarrollo inicial, ya que se implementa la funcionalidad parcial. | El modelo en espiral puede adaptarse y aplicarse a lo largo de la vida del software de computadora. | No modifica el flujo del ciclo de vida |
| El modelo conduce a una extensa documentación técnica, que es un alivio para los nuevos programadores y desarrolladores y también es útil en la fase de prueba. | La relación entre las etapas de desarrollo y los distintos tipos de pruebas facilitan la localización de fallos. | Permite separar la complejidad del proyecto, gracias a su desarrollo por parte de cada iteración o bloque. | Si se detecta un error grave, solo desechamos la última iteración. | Como el software evoluciona a medida que progresa el proceso, el desarrollador y el cliente comprenden y reaccionan mejor ante riesgos en cada uno de los niveles evolutivos. | Este modelo es útil cuando el cliente conoce los objetivos generales para el software, pero no identifica los requisitos detallados de entrada, procesamiento o salida. |
| El progreso del proyecto puede ser monitoreado usando metas. | Es un modelo sencillo y de fácil aprendizaje | El producto es consistente y puntual en el desarrollo. | No es necesario disponer de los requerimientos de todas las funcionalidades en el comienzo del proyecto y además facilita la labor del desarrollo con la frase de divide y conquistaras. | El modelo en espiral permite a quien lo desarrolla aplicar el enfoque de construcción de prototipos en cualquier etapa de evolución del producto. | Reduce costo y aumenta la probabilidad de éxito |
| El coste total puede estimarse con relativa precisión si no hay conflictos. | Involucra al usuario en las pruebas. | Se obtiene un aprendizaje en cada iteración que es aplicado en el desarrollo del producto y aumenta las experiencias para próximos proyectos. | También provee un impacto ventajoso frente al cliente, que es la entrega temprana de partes operativas del software | La utilización de grandes sistemas ha doblado la productividad. | Reduce el riesgo de construir productos que no satisfagan las necesidades de los usuarios |
| Permite comenzar a desarrollar el Software con rapidez. | Es un modelo sencillo y de fácil aprendizaje. | En el desarrollo de este modelo se da la retroalimentación muy temprano a los usuarios. | Permite entregar al cliente un producto más rápido en comparación del modelo de cascada. | El modelo en espiral demanda una consideración directa de los riesgos técnicos en todas las etapas del proyecto y si se aplica adecuadamente debe reducir los riesgos antes de que se conviertan en problemas. | Modificación del Sistema en etapas tempranas de su desarrollo: El éxito del uso del prototipo depende de qué tan pronto y con qué frecuencia se reciba la retroalimentación del usuario para hacer cambios y adecuarlos a las necesidades actuales. |

## Metodologías agiles y tradicionales

|  |  |
| --- | --- |
| **Metodología Ágiles** | **Metodología Tradicionales** |
| Orientada a proyectos pequeños. | Orientada a proyectos de cualquier envergadura. |
| Problemas de escalabilidad en proyectos grandes. | Problemas de adaptación a proyectos pequeños. |
| Se esperan cambios importantes durante el proyecto. | No se esperan cambios importantes durante el proyecto. |
| Enfocada a equipos pequeños. | Enfocada a equipos grandes y dispersos. |
| Multifuncional. | Especialidad en una cosa. |
| Se desarrolla en base a las necesidades detectadas. | Desfase entre la situación actual del producto y la situación real del mercado. |
| El cambio de forma es parte del proceso. | Su naturaleza se resiste al cambio. |
| Énfasis en el individuo y el trabajo individual. | Énfasis en la documentación del proceso: Roles, actividades, etc. |
| Poco control de cambios. | Control de cambios estricto y aprobado. |
| Enfocada a proyectos que posean poca documentación. | Proyectos con mucha documentación. |
| Mayor flexibilidad en los contratos. | Contratos prefijados. |
| El cliente es parte del equipo. | El cliente es informado mediante reuniones con la dirección. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Rational Unified Process** | **Extreme Programming** | **Scrum** | **Cascada** |
| RUP | XP | Scrum | Cascada o Secuencial |
| Kruchten 1996 | Beck 1999 | Sutherland 1994  Schwaher 1995 | Winston W. Royce 1970 |
| Grandes empresas orientadas al desarrollo Web y móvil | Desarrollo de proyectos pequeños debido a evitar grandes costos. | Todo tipo de empresas orientadas a pequeños, medianos y grandes proyectos, | Grandes empresas orientados hacia grandes proyectos empresariales y Web |
| fases:  -Inicio  -Elaboración  -Construcción  -Transición | fases:  -Definición de roles  -Estimación del esfuerzo  -Elección de construcción  -Programación  -Repetición | fases:  -Reunión de planificación de Sprint  -El Scrum Diario  -Desarrollo durante el Sprint  -Revisión del Sprint  -Retrospectiva del Sprint | fases:  -Pre-análisis  -Análisis  -Diseño  -Desarrollo  -Pruebas  -Implementación y mantención |
| Estrecha relación con el cliente | La programación extrema gira en torno al cliente, al punto que se le considera como parte del equipo. | Constante relación con el cliente durante todo el ciclo de vida del proyecto | Estrecha relación con el cliente al momento de la toma de información |
| En cada fase de la metodología se realizan distintas iteraciones perfeccionando los objetivos y si no se termina una fase se sigue con la siguiente. | Se debe integrar como mínimo una vez al día y se van realizando constantemente las pruebas sobre la totalidad del proceso. | Se necesita revisión diaria describiendo:   1. Trabajo de dia anterior 2. Trabajo previo a realizar 3. Trabajos que se pueden realizar | Los resultados de cada una de las fases del proyecto se deben comparar y verificar inmediatamente con los documentos elaborados previamente. |
| Forma disciplinada de asignar tareas y responsabilidades (quién hace qué, cuándo y cómo). | Los miembros del equipo programan en parejas. | Cada miembro del equipo trabaja de forma individual. | Todo el equipo de trabajo se orienta al desarrollo de una tarea o parte del proyecto y lo desarrollan en conjunto hasta finalizar con la etapa |
| Tiene un desarrollo en general iterativo que las semanas de entregas pueden variar según la envergadura del proyecto. | Las iteraciones de entrega son de 1 a 3 semanas | Las iteraciones de entrega son de 1 a 4 semanas. | Tras finalizar cada una de las etapas se realiza un chequeo y retroalimentación de lo trabajado. |
| El alcance es definido antes del comienzo del proyecto y plasmado en el Documento de Alcance. | El cliente define el alcance al inicio del proyecto, brindándoles una orientación acerca de sus necesidades y lo que desea que el sistema realice. | El alcance es definido en la Lista de Objetivos (Project Backlog) que es reevaluado al final de cada iteración. | Desarrolladores y clientes se ponen de acuerdo en una fase muy temprana del proyecto. |

## Metodología seleccionada

“La metodología que se ha decidido usar en nuestro proyecto es la llamada “SCRUM”, esto después de haber comparado las ventajas y desventajas que poseen las metodologías tradicionales y ágiles además de ver una a una la eficacia que tendrían en nuestro proyecto esto es debido a la reducción en los costes y tiempos de entrega, además de las ventajas que nos ofrecería al mejorar la comunicación entre desarrolladores y cliente puesto que existirían reuniones para analizar los avances día a día en cada “Daily Scrum”, además, que esto disminuiría considerablemente la existencia de posibles fallas o problemas que podrían existir al usar otra metodología. Y esta metodología fomenta la innovación, lo que es útil para programadores con experiencia laboral en el campo y permitiría añadir funciones las cuales podrían ser útiles en nuestro servicio a diseñar. Y para finalizar, lo que más nos atrajo respecto a otras metodologías es la flexibilidad de esta.”