UNIVERSIDAD
DE LA SERENA
CHILE

CICLO DE VIDA
“InventarisPro” v0.2

Equipo de desarrollo:
Antony Rodriguez.
Roberto Contreras.
Sebastian Morgado.

Profesor:
Guillermo Leyton.

Asignatura:
Programación Avanzada

Índice

Propósito

Este documento tiene como objetivo determinar el ciclo de vida para el software InventarisPro basado en :

- Las características del problema.
- Las características del ámbito de trabajo.
- Las preguntas del libro “Ingeniería de Software Sommerville 9na Edición”.[1]

Características del problema

A continuación se describirán las características del problema de optimización del inventario de un negocio.

- **¿El problema es lineal o no lineal?**
  - El problema es **no lineal**, ya que no todas las variables del problema tienen el mismo peso, para cada negocio es diferente el peso de estas, por lo tanto cambia al momento de optimizar un stock de un negocio a otro.

    Ejemplo:

    Para los negocios de Abarrotes A y B, los dos venden jugo y pan. En el negocio A se vende más pan que jugo, mientras que en B ocurre lo contrario. Por lo tanto la optimización del stock de los productos del negocio A sería diferente a la del negocio B, haciendo este problema no lineal.
    Este caso se repetiría en los distintos tipos de negocio que nuestro modelo abarca.[2]

- **¿El problema es estático o dinámico?**
  - El problema es **dinámico**, ya que optimizar el stock se ve afectado con el paso del tiempo.
    Ejemplo:
    En el caso de una tienda de ropa, importa la estación del año en que se encuentren, ya que dependiendo de está los clientes comprarán una determinada ropa y por ende la optimización del stock cambia.

- **¿El problema es sincrónico o asincrónico?**
  - El problema es **asíncrono**, porque el hecho de que factores como el agotamiento del stock de un producto, la insuficiencia de stock ante una alta demanda de un producto o que el dueño del establecimiento no pueda atender el negocio por razones personales, no necesariamente hace que falle el proceso de optimización. El proceso se puede retomar después de solucionar estos inconvenientes.

[1] Ingeniería de Software Sommerville 9na Edición
[2] Ver Alcance en Plan General

- **¿El problema es determinista?**
  - El optimizar el stock es determinista.
    - Siempre que se tenga un stock de productos que analizar, este podrá optimizarlo.

- **¿El problema es estable?**
  - El problema **NO** es estable, por que si existe perturbación alguna que modifique el stock del inventario (robo o merma), la optimización se ve afectada negativamente dependiendo de la importancia del producto.

- **¿El problema es isomórfico?**
  - El problema de optimizar el stock NO es isomórfico, porque para serlo, este debe ser lineal y biyectivo, cosa con la que no cumple nuestro sistema, basta con leer que no es lineal.

- **¿El problema es uni-sistémico?**
  - No, ya que participan tres sistemas, el vendedor, la base de datos y el software. El vendedor realiza las ventas siempre y cuando hayan existencias registradas en la base de datos, de esta manera el software puede recomendar como optimizar el stock.

- **¿El problema es azaroso?**
  - Es azaroso, ya que las variables internas, factores y las cualidades propias de la persona que emprende un negocio están expuestas a contratiempos y dificultades.

Características del ámbito de trabajo

- Contamos con 36 días (sin contar el 12/12) desde hoy 06/11 hasta el día 12/12 para el desarrollo del proyecto.
- Grupo de desarrollo de tres integrantes, con un compromiso alto y una flexibilidad y disposición media. Sin experiencia práctica en el tipo de desarrollo.
- Dado el tiempo de desarrollo y la situación actual del proyecto se requiere o se debe tener en cuenta tiempos de desarrollo rápidos para el desarrollo de las tareas.
- Grupo SQA de dos integrantes con un compromiso, flexibilidad y disposición desconocida.
- Consideramos como cliente en este proyecto tanto al profesor como al equipo SQA, ya que son ellos de cierta forma los que permiten la continuidad del proyecto.
- El cliente no participa de forma constante durante todo el proceso de desarrollo, más bien en algunas instancias específicas de revisión.
- Se consideran reuniones y participación activa entre los miembros del equipo.
- Contemplamos 2 iteraciones principales para este proyecto:
  - Una segunda iteración contempla la entrega y aceptación de una primera fase del prototipo.

➢ Una tercera iteración contempla la entrega y aceptación de la segunda fase del prototipo integrada con el prototipo anterior.

Preguntas del libro “Ingeniería de Software Sommerville 9na Edición”

Para decidir sobre el equilibrio entre un enfoque basado en un plan y uno ágil, se responderán algunas preguntas técnicas, humanas y organizacionales:

Marcadores distintivos de metodologías:

- **Metodología clásica**
- **Metodología Ágil**

1. ¿Es importante tener una especificación y un diseño muy detallados antes de dirigirse a la implementación? Siendo así, probablemente usted tenga que usar un enfoque basado en un plan.

- Si bien las especificaciones y requerimientos han evolucionado a través del desarrollo del proyecto, es prioritario generar un buen diseño del modelo para luego implementarlo en un prototipo de software. -> **Metodología clásica**

2. ¿Es práctica una estrategia de entrega incremental, donde se dé el software a los clientes y se obtenga así una rápida retroalimentación de ellos? De ser el caso, considere el uso de métodos ágiles.

- Si, sería muy útil para obtener opiniones iterativas del cliente respecto al software y mejorar la calidad de este en el plazo de entrega restante. **->** **Metodología Ágil**

3. ¿Qué tan grande es el sistema que se desarrollará? Los métodos ágiles son más efectivos cuando el sistema logra diseñarse con un pequeño equipo asignado que se comunique de manera informal. Esto sería imposible para los grandes sistemas que precisan equipos de desarrollo más amplios, de manera que tal vez se utilice un enfoque basado en un plan.

- Ya que somos un grupo de 3 personas donde se precisa la comunicación informal, además de que se desarrollará un software relativamente pequeño con respecto a una gran empresa de desarrollo de software. **->** **Metodología Ágil**

4. ¿Qué tipo de sistema se desarrollará? Los sistemas que demandan mucho análisis antes de la implementación (por ejemplo, sistema en tiempo real con requerimientos de temporización compleja), por lo general, necesitan un diseño bastante detallado para realizar este análisis. En tales circunstancias, quizá sea mejor un enfoque basado en un plan.

- Se desarrollará un sistema de inventario que optimice el stock. Para esto hay que cumplir con los procesos de ing. de software (hay que desarrollar muchos documentos) entonces, si, es mejor un enfoque basado en plan. **->** **Metodología clásica**

5. ¿Cuál es el tiempo de vida que se espera del sistema? Los sistemas con lapsos de vida prolongados podrían requerir más documentación de diseño, para comunicar al equipo de apoyo los propósitos originales de los desarrolladores del sistema. Sin embargo, los defensores de los métodos ágiles argumentan acertadamente que con frecuencia la documentación no se conserva actualizada, ni se usa mucho para el mantenimiento del sistema a largo plazo.

- Ágil, ya que se espera un tiempo de vida corto, y su escalabilidad no está garantizada. **->** **Metodología ágil**

6. ¿Qué tecnologías se hallan disponibles para apoyar el desarrollo del sistema? Los métodos ágiles se auxilian a menudo de buenas herramientas para seguir la pista de un diseño en evolución. Si se desarrolla un sistema con un IDE sin contar con buenas herramientas para visualización y análisis de programas, entonces posiblemente se requiera más documentación de diseño.

Hoy en la actualidad se tiene acceso a muchas herramientas y de buena calidad, como pueden ser:

- **Visual Studio:** Visual Studio es un IDE el cual pone a disposición una serie de herramientas que permiten optimizar el tiempo de desarrollo e implementación.
- **Trello:** Trello es un software para organizar proyectos a través de la asignación de tareas aisladas, estás tareas se pueden categorizar en (Por hacer, En proceso, Listas), permitiendo así un desarrollo más organizado y expedito.
- **Github:** Sistema de control de versiones
- **Azure DevOps:** DevOps permite que los roles que antes estaban aislados (desarrollo, operaciones de TI, ingeniería de la calidad y seguridad) se coordinen y colaboren para producir productos mejores y más confiables. Los equipos adquieren la capacidad de responder mejor a las necesidades de los clientes, aumentar la confianza en las aplicaciones que crean y alcanzar los objetivos en menos tiempo.
- **Discord:** Para una mejor comunicación entre los integrantes del grupo de desarrollo.
- **Google Docs:** Para el desarrollo y redacción de documentos en tiempo real en grupo.

Es por estas herramientas, las cuales nos permiten agilizar el desarrollo, que se opta por usar la **Metodología ágil.**

7. ¿Cómo está organizado el equipo de desarrollo? Si el equipo de desarrollo está distribuido, o si parte del desarrollo se subcontrata, entonces tal vez se requiera elaborar documentos de diseño para comunicarse a través de los equipos de desarrollo. Quizá se necesite planear por adelantado cuáles son.

- El equipo de desarrollo se encuentra actualmente trabajando en conjunto pero en tareas distintas, cada uno abordando distintos aspectos de la problemática. No se encuentra distribuido como grupos por separado, en el sentido de no necesitar documento de diseño para comunicarse. **->** **Metodología ágil.**

8. ¿Existen problemas culturales que afecten el desarrollo del sistema? Las organizaciones de ingeniería tradicionales presentan una cultura de desarrollo basada en un plan, pues es una norma en ingeniería. Esto requiere comúnmente una amplia documentación de diseño, en vez del conocimiento informal que se utiliza en los procesos ágiles.

- Es necesario seguir los pasos de ingeniería de software, pasos en los cuales está incorporado varios documentos de planificación, que además tienen prioridad de trabajo por sobre el desarrollo de software. **->** **Metodología clásica.**

9. ¿Qué tan buenos son los diseñadores y programadores en el equipo de desarrollo?
Se argumenta en ocasiones que los métodos ágiles requieren niveles de habilidad superiores a los enfoques basados en un plan, en que los programadores simplemente traducen un diseño detallado en un código. Si usted tiene un equipo con niveles de habilidad relativamente bajos, es probable que necesite del mejor personal para desarrollar el diseño, siendo otros los responsables de la programación.

- Los programadores y desarrolladores de nuestro equipo no cuentan con la experiencia suficiente con respecto a la documentación correspondiente a la ingeniería de software. -> **Metodología clásica**

10. ¿El sistema está sujeto a regulación externa? Si un regulador externo tiene que aprobar el sistema (por ejemplo, la Agencia de Aviación Federal [FAA] estadounidense aprueba el software que es crítico para la operación de una aeronave), entonces, tal vez se le requerirá documentación detallada como parte del sistema de seguridad.

- El software es evaluado por organismos externos, lo evalúa el profesor de la asignatura y el grupo SQA, los cuales en este contexto se consideran organismo externos al equipo de desarrollo. -> **Metodología clásica**

Conteo final

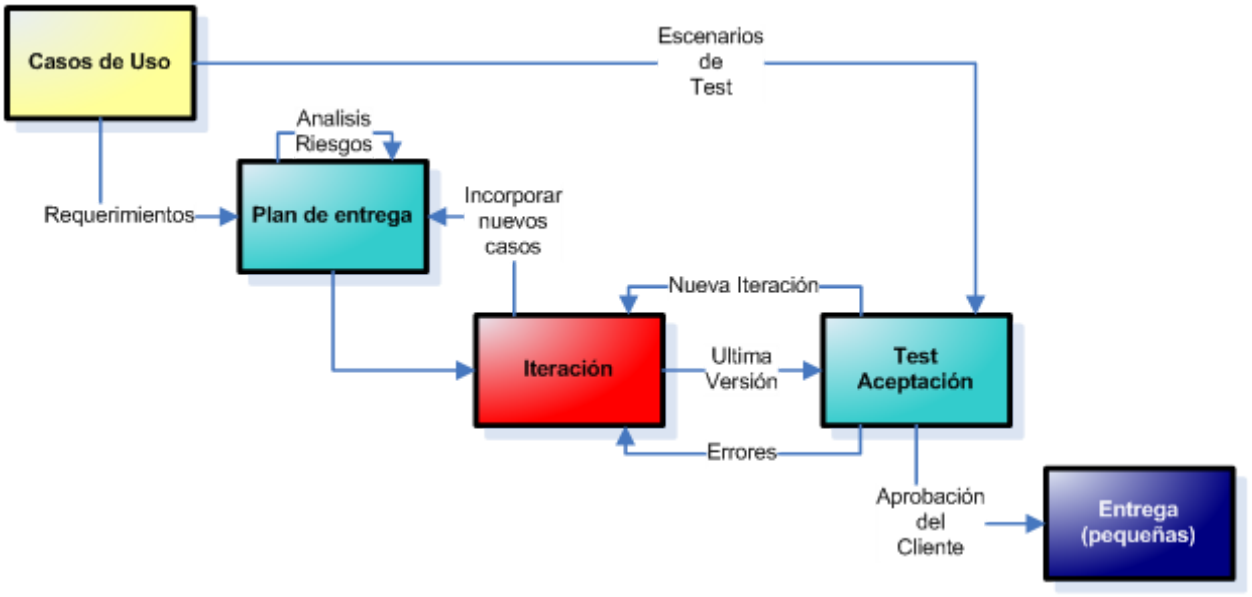
Metodología	Conteo
Metodología basado en Plan	5
Metodología ágil	5

En base al conteo final, se puede rescatar que el proyecto es descrito como un 50% perteneciente a la metodología Ágil y un 50% perteneciente a la metodología de Plan. Debido a lo mencionado anteriormente y por la cantidad de tiempo restante que queda para la entrega de este proyecto, además de la necesidad de completar las tareas o hitos en plazo de tiempos cortos, se considerarán **Metodologías Ágiles** por sobre las **Metodologías Clásicas**.

Ciclos de vida Candidatos:

Programacion Extrema:

Extreme Programming (o XP) es una metodología de desarrollo que pertenece a las denominadas metodologías ágiles, cuyo objetivo es el desarrollo y gestión de proyectos con eficiencia, flexibilidad y control.
Esta metodología consiste en un conjunto de prácticas, basadas en valores que los participantes en el proyecto deben mantener, que a través del trabajo en grupo, tiene como objetivo crear como producto final, un software con un muy alto grado de calidad.

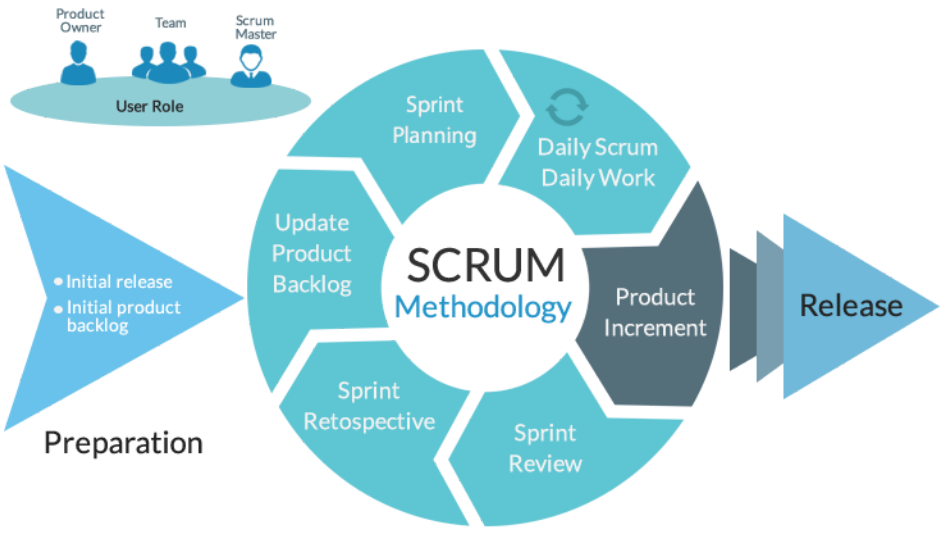


Ventajas	Desventajas
Relación estrecha con el cliente/ grupo validador.	Es recomendable emplear XP solo en proyectos a corto plazo y simples.
Ausencia de trabajos de programación innecesarios.	Requiere de un rígido ajuste a los principios de XP.
Menos errores gracias a la programación en pareja.	Puede no siempre ser más fácil que el desarrollo tradicional.
Aplicación rápida de cambios.	Requiere control de versiones dada la falta de documentación sobre cada cambio.
Permite ahorrar tiempo.	Mayor esfuerzo de trabajo.

Se hacen pruebas continuas durante el proyecto.	Requiere de un uso continuo del tiempo de trabajo para cumplir con el ciclo.

Scrum

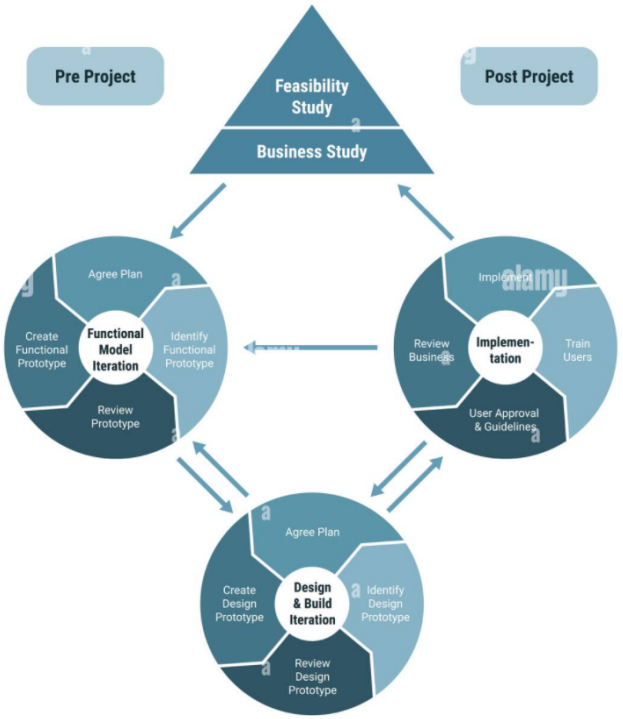
Scrum es un proceso en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente, en equipo, y obtener el mejor resultado posible de un proyecto. Estas prácticas se apoyan unas a otras y su selección tiene origen en un estudio de la manera de trabajar de equipos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



Ventajas	Desventajas
Puede ayudar a un equipo completar entregas de proyectos rápidamente y eficientemente	Scrum a menudo conduce a un deslizamiento del alcance, debido a la falta de una fecha de finalización definida
Asegura un uso efectivo del tiempo	Las posibilidades de que el proyecto fracase son altas si las personas no están muy comprometidas o no cooperan
Los desarrollos se codifican y prueban durante la revisión del sprint	El marco sólo puede tener éxito con miembros del equipo experimentados
Funciona bien para proyectos de desarrollo de rápido movimiento	Las reuniones diarias a veces frustran a los miembros del equipo
El equipo obtiene una visibilidad clara a través de las reuniones de scrum	Si algún miembro del equipo se va en medio de un proyecto, puede tener un gran impacto negativo en el proyecto
Los sprints cortos permiten cambios basados en comentarios con mucha más facilidad	La calidad es difícil de implementar hasta que el equipo pasa por un proceso de prueba agresivo
El esfuerzo individual de cada miembro del equipo es visible durante las reuniones diarias de scrum	Los equipos de Scrum a veces tienen dificultades para adoptar la metodología Scrum porque es un gran cambio con respecto a sus métodos de trabajo tradicionales.

Dynamic Systems Development Method (DSDM)

DSDM es una metodología ágil de desarrollo de software. Es un enfoque iterativo e incremental que se basa en gran medida en la metodología de Desarrollo Rápido de Aplicaciones (RAD).
DSDM considera la necesidad de reaccionar rápidamente a lo largo del desarrollo del producto al tiempo que incorpora las limitaciones que a menudo imponen las culturas y procesos corporativos.
La gestión de proyectos mediante DSDM ayuda a resolver muchos de los problemas a los que nos enfrentamos cuando ejecutamos proyectos de forma más tradicional.
Los clientes son humanos. A medida que su conocimiento del producto aumenta a lo largo de la fase de desarrollo, tienden a cambiar de opinión. Por lo tanto, el enfoque de DSDM para comprender los requisitos del cliente es involucrar a un cliente en el desarrollo de la solución.
La participación temprana del cliente ayuda a evitar problemas, evitar que se construyan funciones no deseadas y garantizar que la empresa obtenga lo que necesita. También ayuda a evitar el exceso de ingeniería o el recubrimiento de oro de la producción.



Ventajas	Desventajas
La calidad del producto es mejorada a través de la participación de los usuarios a lo largo del ciclo de vida del proyecto.	Se necesita una alta participación de los usuarios en el desarrollo, para evitar que los desarrolladores asuman criterios que no son ciertos.
Asegura desarrollos rápidos.	No es una metodología de desarrollo común. El proceso es un tanto difícil de comprender
Permite realizar cambios de forma fácil.	
Permite la reutilización de aplicación a través de los módulos existentes.	
Reduce los costos de proyectos a través de las ventajas mencionadas anteriormente	

Tabla de importancia

Para determinar el ciclo de vida a seguir, se elaboraron las siguientes tablas. La primera es la tabla de importancia que contiene el nivel de importancia para cada característica y el peso asociado a dicha característica. La segunda tabla contiene características que se tendrán en cuenta para evaluar la compatibilidad de cada ciclo de vida. Esta compatibilidad será medida de la siguiente manera:

Nivel de importancia	Peso
Característica esencial	3
Característica importante, pero no esencial	2
Deseable, pero puede prescindir de ésta sin tener un alto impacto en la decisión.	1

Característica	Scrum	XP	DSDM
Se adapta a una documentación y diseño específico	3	2	1
Se adapta para un sistema pequeño	2	3	2
Se adapta para un corto tiempo de vida (fecha límite para la entrega del proyecto)	3	3	3
Se adapta para un grupo pequeño de desarrollo	2	3	2
Se adapta para un corto tiempo de desarrollo (iteraciones)	2	2	3
Se adapta para el desarrollo de un prototipo (versión no final del sistema)	2	3	2
Se adapta a las iteraciones del grupo SQA	2	3	2
Se de adapta con la poca experiencia por parte del grupo de desarrollo	1	1	1
Total	17	20	16

Conclusión

A partir de los 3 ciclos de vida descritos anteriormente y sus respectivas evaluaciones, como equipo, se ha elegido XP dada la naturaleza del proyecto y lo bien que encaja con las características del estado actual del proyecto. Como de desarrollo contamos con un poco más de un mes para hacer una entrega formal del proyecto, XP nos asegura un uso eficiente del tiempo, a través de la división de tareas para el desarrollo y su potencial al aplicarlo en proyectos a corto plazo.