INSTITUTO TECNOLÓGICO DE MINATITLÁN

INGENIERÍA EN SISTEMAS COMPUTACIONALES

“MANUAL DE PRÁCTICAS “

MATERIA

GESTION DE PROYECTOS EN SOFTWARE

MINATITLÁN, VER. JUNIO DEL 2023

3.2 ÍNDICE DEL MANUAL DE PRÁCTICAS

3.1 INTRODUCCIÓN

El presente manual dará a conocer las prácticas relacionadas con los cinco temas de la materia de Gestión de Proyectos en Software.

3.2 JUSTIFICACIÓN

Un Manual de prácticas puede definirse como un compendio de documentos que contemplan una serie de aportes a la práctica científica y social de los alumnos que se encuentren realizando dicha práctica, las cuales también incluyen las normas y procedimientos que orientarán el desempeño del alumno y facilitarán la integración de la teoría con la práctica, en un contexto real de aprendizaje.

Este manual de prácticas está basado según el contenido de “el libro Guía para la elaboración y registro de textos o trabajos académicos”, con el que cuenta el Tecnológico Nacional de México.

El manual de prácticas servirá como apoyo de aprendizaje para los alumnos de la materia de Gestión de proyectos en software y también como apoyo didáctico para los maestros de dicha materia, ya que se presentarán consejos y sugerencias para dicha realización de las prácticas también se dará materia de apoyo para estas mismas.

3.3 OBJETIVO GENERAL DEL MANUAL DE PRÁCTICAS

El objetivo que se pretende lograr con este manual de prácticas es aplicar los elementos de la investigación documental para elaborar escritos académicos de su entorno profesional.

3.4 DESARROLLO

3.4.1 Práctica 1 Investigar y elaborar un cuadro comparativo de las etapas que integran las 5 metodologías tradicionales como son: Modelo en espiral, Cascada, Modelo prototipo, desarrollo por etapas, desarrollo interactivo y depurativo.

3.4.1.1 Objetivo

La elaboración de un cuadro comparativo de las etapas de las 5 metodologías tradicionales (Modelo en espiral, Cascada, Modelo prototipo, desarrollo por etapas, desarrollo interactivo y depurativo) es una práctica útil para comprender las diferencias y similitudes entre estas metodologías en el proceso de desarrollo de software. El objetivo principal de realizar esta práctica es ayudar a los profesionales de TI y a las organizaciones a seleccionar la metodología más adecuada para un proyecto específico en función de sus necesidades y requisitos.

El objetivo de realizar este cuadro comparativo es ayudar a los equipos de desarrollo de software y a las organizaciones a comprender las diferencias en la secuencia de actividades, el enfoque en la planificación y la gestión de riesgos, y el grado de flexibilidad que ofrecen estas metodologías. Al comprender estas diferencias, se puede tomar una decisión más informada sobre qué enfoque adoptar para un proyecto específico, teniendo en cuenta sus requisitos y objetivos únicos.

3.4.1.2 Introducción

Las metodologías tradicionales imponen una disciplina rigurosa de trabajo sobre el proceso de desarrollo del software, con el fin de conseguir un software más eficiente. Para ello, se hace énfasis en la planificación total de todo el trabajo a realizar y una vez que está todo detallado, comienza el ciclo de desarrollo del producto software. Se centran especialmente en el control del proceso, mediante una rigurosa definición de roles, actividades, artefactos, herramientas y notaciones para el modelado y documentación detallada. Además, las metodologías tradicionales no se adaptan adecuadamente a los cambios, por lo que no son

métodos adecuados cuando se trabaja en un entorno, donde los requisitos no pueden predecirse o bien pueden variar.

3.4.1.3 Correlación Los Temas Y Subtemas Del Programa De Estudio Vigente.

Esta actividad corresponde al subtema 1.1. Conceptos básicos para la gestión de proyectos

3.4.1.4 Material Y Equipo Necesario

1. Hojas blancas tamaño carta.
2. Lápiz de grafito.
3. Goma de Borrar
4. Sacapuntas
5. Equipo de computo
6. Internet
7. Microsoft power point

3.4.1.5 Metodología

Paso 1: Definir el Alcance de la Práctica

Establece claramente los objetivos y el alcance de la práctica. ¿Qué deseas lograr con esta comparación de metodologías? ¿Cuáles son los criterios clave de evaluación?

Paso 2: Selección de Metodologías

Elige las cinco metodologías tradicionales que deseas comparar: Modelo en Espiral, Cascada, Modelo Prototipo, Desarrollo por Etapas, Desarrollo Interactivo y Depurativo.

Paso 3: Investigación

Investiga cada una de las metodologías seleccionadas en profundidad. Recopila información sobre sus características, principios, ventajas y desventajas, y las etapas clave que las componen.

Paso 4: Creación de un Cuadro Comparativo

Crea una tabla o cuadro comparativo donde se detallen las etapas de cada una de las metodologías. Incluye las siguientes columnas: Metodología, Etapas, Características Clave.

Paso 5: Comparación Detallada

Llena el cuadro comparativo con la información recopilada en el paso anterior. Asegúrate de describir cada etapa de manera clara y concisa, destacando las características clave de cada una.

Paso 6: Análisis de Similitudes y Diferencias

Analiza el cuadro comparativo para identificar similitudes y diferencias entre las etapas de las metodologías. Pregúntate cuáles son las ventajas y desventajas de cada enfoque en función de las etapas específicas.

Paso 7: Evaluación y Conclusiones

Evalúa las metodologías en función de los objetivos definidos en el paso 1. ¿Cuál de las metodologías es más adecuada para un proyecto específico? ¿Cuáles son las consideraciones clave que deben tenerse en cuenta al seleccionar una metodología?

Paso 8: Documentación y Presentación

Documenta los hallazgos, conclusiones y recomendaciones en un informe o presentación. Asegúrate de respaldar tus recomendaciones con evidencia sólida y argumentos sólidos.

Paso 9: Revisión y Validación

Revisa y valida tu trabajo para garantizar que esté completo y preciso. Considera buscar retroalimentación de colegas o profesores si es posible.

Paso 10: Presentación Final

Presenta tus hallazgos y recomendaciones de manera clara y concisa según sea necesario. Puedes utilizar gráficos o visualizaciones para resaltar las diferencias entre las metodologías si es útil.

3.4.1.6 Sugerencias Didácticas

- Ninguna

3.4.1.7 Reporte Del Alumno

El alumno debe de realizar la actividad detallando paso a paso la elaboración de esta, incluyendo capturas, mediante el formato de un reporte de prácticas dando detalle de los resultados obtenidos, así como su conclusión y aprendizajes obtenidos.

3.4.1.8 Bibliografías

Braude Eric J. (2003), E. Ingeniería de Software una perspectiva orientada a objetos. México:Alfaomega.

Piattini M.G. (2007), Calidad de Sistemas Informáticos. México: Alfaomega. Sommerville, I. (2011). Ingeniería de Software. España: Pearson Addison Wesley

Presuman, R. S. (2010), Ingeniería del Software un enfoque práctico. México: MC Graw-Hill.

3.4.2 Práctica 2 Investigar y elaborar un cuadro comparativo de las etapas que integran las 5 metodologías agiles como son:: Programación Extrema, Metodología Scrum, Metodología Crystal, Metodología MDSD, Metodología Kanban.

3.4.2.1 Objetivo

El objetivo de esta práctica es que los participantes adquieran un profundo entendimiento de las metodologías ágiles más relevantes en el desarrollo de software, identificando y comparando las etapas clave de cada una. Al finalizar la práctica, los estudiantes serán capaces de seleccionar y justificar la elección de la metodología ágil más adecuada para proyectos específicos, considerando sus características, ventajas y desventajas.

3.4.2.2 Introducción

Cuando hablamos de gestión ágil de proyecto (o “Agile”, como se usa en inglés), no hablamos de una metodología concreta. Más bien es una serie de metodologías ágiles en la gestión de proyectos donde la idea principal es que el producto final se divide en varias etapas o tareas más pequeñas que se pueden entregar más rápidos y más fáciles. Las partes interesadas evalúan estas tareas regularmente. Y de este modo, el equipo puede responder a los cambios más rápidamente mediante la construcción iterativa del producto final.

3.4.2.3 Correlación Con Los Temas Y Subtemas Del Programa De Estudio Vigente.

Esta actividad corresponde al subtema 1.1. Conceptos básicos para la gestión de proyectos

3.4.2.4 Material Y Equipo Necesario

1. Hojas blancas tamaño carta.
2. Lápiz de grafito.

3. Goma de Borrar
4. Sacapuntas
5. Equipo de computo
6. Internet
7. Microsoft power point

3.4.2.5 Metodología

Paso 1: Selección de Metodologías

Elije las cinco metodologías ágiles que deseas comparar: Programación Extrema (XP), Scrum, Crystal, MDSD y Kanban.

Paso 2: Investigación Individual

Asigna a cada estudiante o equipo una metodología específica para investigar en profundidad. Proporciona recursos, libros, documentos y ejemplos de proyectos relacionados con cada metodología.

Paso 3: Análisis de Etapas

Pide a los estudiantes que identifiquen y describan las etapas clave de la metodología asignada. Deben enfocarse en las fases del ciclo de vida del desarrollo de software que son relevantes para esa metodología en particular.

Paso 4: Creación del Cuadro Comparativo

Organiza la información recopilada en un cuadro comparativo. Crea una tabla que incluya las siguientes columnas: Metodología, Etapas, Características Clave.

Paso 5: Comparación y Contraste

Anima a los estudiantes a analizar y comparar las etapas de las distintas metodologías. ¿Qué similitudes y diferencias encuentran? ¿Cuáles son las ventajas y desventajas de cada enfoque?

Paso 6: Discusión Grupal

Organiza sesiones de discusión grupal donde los estudiantes compartan sus hallazgos y discutan las diferencias y similitudes entre las metodologías. Fomenta el intercambio de ideas y preguntas.

Paso 7: Conclusiones y Recomendaciones

Pide a los estudiantes que elaboren conclusiones basadas en la comparación de las metodologías. ¿Cuándo sería apropiado utilizar una metodología sobre otra? ¿Qué factores deben considerarse al tomar una decisión?

Paso 8: Presentación

Invita a los estudiantes a presentar sus hallazgos y recomendaciones ante el grupo. Esto promoverá la comunicación efectiva y la capacidad de presentación.

Paso 9: Evaluación

Evalúa el trabajo de los estudiantes teniendo en cuenta la calidad de sus investigaciones, el cuadro comparativo, la participación en la discusión y las presentaciones.

Paso 10: Reflexión y Retroalimentación

Pide a los estudiantes que reflexionen sobre lo que han aprendido y cómo esta comprensión puede influir en su futura práctica en el desarrollo de software. También recopila retroalimentación sobre la efectividad de la práctica.

3.4.2.6 Sugerencias Didácticas

- Introducción y Contextualización: Comienza la clase con una breve introducción a las metodologías ágiles y su importancia en el desarrollo de software. Explica el propósito de la práctica.
- División en Equipos: Divide a los estudiantes en equipos, asignando a cada equipo una metodología ágil específica para investigar.

- Investigación Individual: Cada miembro del equipo investiga una fase o etapa clave de la metodología asignada y recopila información relevante.

3.4.2.7 Reporte Del Alumno

El alumno debe de realizar la actividad detallando paso a paso la elaboración de esta, incluyendo capturas, mediante el formato de un reporte de prácticas dando detalle de los resultados obtenidos, así como su conclusión y aprendizajes obtenidos.

3.4.2.8 Bibliografías

- Braude Eric J. (2003), E. Ingeniería de Software una perspectiva orientada a objetos. México:Alfaomega.
- Piattini M.G. (2007), Calidad de Sistemas Informáticos. México: Alfaomega. Sommerville, I. (2011). Ingeniería de Software. España: Pearson Addison Wesley
- Presuman, R. S. (2010), Ingeniería del Software un enfoque práctico. México: MC Graw-Hill.

3.4.3 Práctica 3 Realizar un investigación sobre ¿Qué es el PMBOK, y como se encuentra integrada su guía de referencia?

3.4.3.1 Objetivo

El objetivo de esta práctica es investigar y comprender el significado y contenido del PMBOK (Project Management Body of Knowledge) y su integración en la guía de referencia. La práctica busca proporcionar una visión integral de la estructura, conceptos y enfoques de gestión de proyectos presentes en el PMBOK, así como identificar cómo esta guía de referencia se relaciona y se utiliza en el ámbito de la gestión de proyectos a nivel global.

3.4.3.2 Introducción

La Guía PMBOK identifica el subconjunto de fundamentos de gestión de proyectos que es "generalmente reconocido" como una "buena práctica". Con "generalmente reconocido" se trata de referir a los conocimientos y prácticas aplicables a la mayoría de los proyectos, la mayor parte del tiempo; en la que hay un consenso sobre su utilidad e importancia; mientras que "buena práctica" implica que hay un acuerdo general para la aplicación de los conocimientos, habilidades, herramientas y técnicas que pueden aumentar las posibilidades de éxito a lo largo de muchos proyectos.

Sin embargo, esto no significa que las tendencias de gestión de proyectos estén especificadas o incluidas en la guía. (Por ejemplo, el parámetro de arrastre de camino crítico, una metodología aplicable para la gestión de un proyecto, no está definida en sí en la guía PMBOK).

La Guía PMBOK también es usada para la preparación de las certificaciones ofrecidas por el PMI.

3.4.3.3 Correlación Con Los Temas Y Subtemas Del Programa De Estudio Vigente.

Esta actividad corresponde al subtema 1.3 Fundamentos de Project Management Institute.

3.4.3.4 Material Y Equipo Necesario

1. Hojas blancas tamaño carta.
2. Lápiz de grafito.
3. Goma de Borrar
4. Sacapuntas
5. Equipo de computo
6. Internet

7. Microsoft Word

3.4.3.5 Metodología

Paso 1: Introducción al PMBOK

Comienza la práctica con una introducción al PMBOK (Project Management Body of Knowledge) y su importancia en la gestión de proyectos. Explica por qué es una guía de referencia relevante y ampliamente reconocida.

Paso 2: Selección de Temas Relevantes

Identifica los temas y áreas de interés relacionados con el PMBOK que se abordarán en la práctica. Esto podría incluir los procesos de gestión de proyectos, las áreas de conocimiento, los grupos de procesos, entre otros.

Paso 3: Investigación Individual

Asigna a los estudiantes o equipos específicos temas relacionados con el PMBOK para investigar en profundidad. Proporciona recursos, como el propio PMBOK, libros, sitios web y documentos técnicos.

Paso 4: Comprensión de la Estructura

Pide a los estudiantes que analicen la estructura del PMBOK, incluyendo la organización de procesos, áreas de conocimiento y grupos de procesos. Comprender la estructura es fundamental para contextualizar la información.

Paso 5: Creación de un Cuadro o Diagrama

Anima a los estudiantes a crear un cuadro o diagrama que visualice la relación entre los procesos, áreas de conocimiento y grupos de procesos del PMBOK. Esto ayudará a clarificar la información.

Paso 6: Integración con la Guía de Referencia

Explora cómo el PMBOK se integra en la guía de referencia utilizada en la gestión de proyectos, como el estándar PMI (Project Management Institute) o cualquier otro estándar adoptado por la organización.

Paso 7: Discusión en Grupo

Organiza sesiones de discusión en grupos pequeños, donde los estudiantes compartan sus hallazgos y debatan sobre la aplicación práctica del PMBOK en proyectos reales.

Paso 8: Presentación de Resultados

Pide a los equipos que presenten sus hallazgos y conclusiones ante el grupo completo. Fomenta el diálogo y la retroalimentación.

Paso 9: Estudio de Casos Prácticos

Proporciona casos prácticos o ejemplos de proyectos que aplican los principios y conceptos del PMBOK. Analiza cómo se utilizan en situaciones reales.

Paso 10: Reflexión Individual

Pide a los estudiantes que reflexionen individualmente sobre la utilidad del PMBOK y cómo pueden aplicarlo en su futura carrera profesional en gestión de proyectos.

Paso 11: Evaluación y Retroalimentación

Evalúa el trabajo de los estudiantes teniendo en cuenta la calidad de sus investigaciones, presentaciones y participación en discusiones. Recopila retroalimentación sobre la efectividad de la práctica.

Paso 12: Conclusiones Finales

Finaliza la práctica resumiendo las conclusiones generales sobre el PMBOK y su relevancia en la gestión de proyectos.

3.4.3.6 Sugerencias Didácticas

- Introducción Interactiva: Comienza la práctica con una actividad interactiva que destaque la importancia de la gestión de proyectos y cómo el PMBOK

se relaciona con ella. Esto puede incluir ejemplos de proyectos exitosos o desafiantes.

- Grupos de Investigación: Divide a los estudiantes en grupos pequeños y asigna a cada grupo un tema específico relacionado con el PMBOK para investigar. Esto fomenta la colaboración y la distribución de tareas.
- Recursos Diversificados: Proporciona una variedad de recursos, como el PMBOK en sí, libros, videos, y casos de estudio, para que los estudiantes puedan obtener información de múltiples fuentes.

3.4.3.7 Reporte Del Alumno

El alumno debe de realizar la actividad detallando paso a paso la elaboración de esta, incluyendo capturas, mediante el formato de un reporte de prácticas dando detalle de los resultados obtenidos, así como su conclusión y aprendizajes obtenidos.

3.4.3.8 Bibliografías

- Wisocki, R. K. (2009). Effective Project Management. Indianapolis, Indiana, USA: Wiley Publishing, Inc.
- Chemuturi, M. & Caglet, T. M. (2010). Mastering Software Project Management: Best Practices, Tools and Techniques. USA: J. Ross Publishing
- Konrad Mike, Shrum Sandy, CMMI (2ª ed.): Guía para la integración de procesos y la mejora d e productos. Madrid: Addison Wesley. (ISBN 9788478290963)

3.4.4 Práctica 4 Elaborar el acta constitución del proyecto de software a gestionar y administrar para su desarrollo.

3.4.4.1 Objetivo

El objetivo de esta práctica es elaborar un Acta de Constitución del Proyecto de Software, que sirva como documento fundamental para definir y establecer las bases, alcance, objetivos, recursos, y responsabilidades del proyecto de desarrollo de software a gestionar y administrar. Esta práctica busca proporcionar una sólida estructura inicial para el proyecto, garantizando la comprensión y el compromiso de todas las partes involucradas en su ejecución y éxito.

3.4.4.2 Introducción

El acta de constitución del proyecto (Project Charter) es el documento en el cual se documenta ese punto de partida, la relación entre estrategia organizacional y el alcance del proyecto, así como la relación de colaboración que existirá entre la organización solicitante del proyecto y la organización ejecutora.

Es un componente de suma importancia para la gerencia de proyectos, en especial en las fases de inicio y planificación. Es un documento al cual se hará referencia en toda la vida del proyecto.

Como lo establece la metodología PMI en la guía del PMBOK, el acta de constitución del proyecto es un documento emitido por el iniciador o patrocinador (Sponsor) que autoriza formalmente la existencia de un proyecto, confiriendo al gerente del proyecto (El Project Manager) la autoridad para asignar recursos de la organización a sus actividades.

3.4.4.3 Especificar La Correlación Con El O Los Temas Y Subtemas Del Programa De Estudio Vigente.

Esta actividad corresponde al subtema 2.1 Plan de calidad del software.

3.4.4.4 Material Y Equipo Necesario

1. Hojas blancas tamaño carta.

2. Lápiz de grafito.
3. Goma de Borrar
4. Sacapuntas
5. Equipo de computo
6. Internet
7. Microsoft Word

3.4.4.5 Metodología

Paso 1: Definir el Propósito y Alcance del Proyecto

- Reúne a los interesados clave, incluyendo patrocinadores, clientes y miembros del equipo de proyecto, para definir claramente el propósito y el alcance del proyecto de software. Esto incluye identificar los objetivos, requerimientos y resultados esperados.

Paso 2: Identificar las Partes Interesadas

- Enumera y describe a todas las partes interesadas del proyecto, incluyendo sus roles y responsabilidades. Esto garantizará una comprensión clara de quiénes están involucrados y qué esperan del proyecto.

Paso 3: Designar un Gerente de Proyecto

- Selecciona o designa a un gerente de proyecto responsable de liderar y administrar el proyecto de software. Define sus responsabilidades y autoridad en el acta.

Paso 4: Definir los Objetivos y Metas

- Especifica los objetivos y metas del proyecto, asegurándote de que sean medibles y alcanzables. Esto proporcionará un punto de referencia para evaluar el éxito del proyecto.

Paso 5: Establecer el Cronograma y los Plazos

- Define un cronograma preliminar que incluya hitos clave y fechas de entrega. Esto ayudará a establecer expectativas de tiempo y gestionar la planificación del proyecto.

Paso 6: Estimar los Recursos

- Identifica los recursos necesarios para llevar a cabo el proyecto, como personal, tecnología, hardware y software. Esto ayudará en la asignación y gestión de recursos.

Paso 7: Analizar y Gestionar los Riesgos

- Identifica posibles riesgos y amenazas que puedan afectar al proyecto y describe estrategias para mitigarlos o abordarlos.

Paso 8: Definir el Presupuesto

- Establece un presupuesto preliminar para el proyecto, incluyendo costos estimados para recursos, adquisiciones y otros gastos relacionados con el desarrollo del software.

Paso 9: Documentar las Aprobaciones

- Especifica los procesos de revisión y aprobación que serán necesarios a lo largo del proyecto. Esto garantiza que las decisiones importantes se tomen de manera informada y con la aprobación adecuada.

Paso 10: Crear el Acta de Constitución del Proyecto

- Utiliza la información recopilada en los pasos anteriores para redactar el Acta de Constitución del Proyecto de Software. Asegúrate de que el documento sea claro, conciso y completo.

Paso 11: Revisión y Validación

- Somete el acta a una revisión y validación por parte de todas las partes interesadas, incluyendo el patrocinador del proyecto, el equipo de desarrollo y otros interesados clave.

Paso 12: Aprobación Final y Distribución

- Una vez que el acta esté validada, obtén la aprobación final de todas las partes interesadas y distribuye el documento a todos los involucrados en el proyecto.

3.4.4.6 Sugerencias Didácticas

- Introducción al Proyecto de Software: Comienza la práctica con una introducción sobre el proyecto de software que se abordará en el ejercicio. Explica su importancia y contexto.
- Formación de Equipos: Divide a los estudiantes en equipos pequeños, asignando a cada equipo un proyecto de software ficticio para el cual elaborarán el Acta de Constitución.
- Definición de Roles: Asigna roles dentro de cada equipo, como el Gerente de Proyecto, el Patrocinador, los Miembros del Equipo, el Cliente, etc.
- Investigación Preliminar: Pide a los equipos que realicen una investigación preliminar sobre el tipo de software que se desarrollará, sus objetivos, el entorno operativo, y las necesidades del cliente.

3.4.4.7 Reporte Del Alumno

El alumno debe de realizar la actividad detallando paso a paso la elaboración de esta, incluyendo capturas, mediante el formato de un reporte de prácticas dando detalle de los resultados obtenidos, así como su conclusión y aprendizajes obtenidos.

3.4.4.8 Bibliografías

- Konrad Mike, Shrum Sandy, CMMI (2ª ed.): Guía para la integración de procesos y la mejora de productos. Madrid: Addison Wesley. (ISBN 9788478290963)

3.4.5 Práctica 5 Elaborar el backlog de la lista de tareas de proyecto del software a gestionar

3.4.5.1 Objetivo

El objetivo de esta práctica es elaborar un Backlog de la Lista de Tareas de Proyecto para el desarrollo de software a gestionar. La práctica tiene como propósito principal definir y priorizar todas las tareas y requisitos del proyecto de manera que se establezca una hoja de ruta clara para el equipo de desarrollo, se gestionen las expectativas de los interesados y se asegure que el proyecto avance de manera efectiva hacia su entrega exitosa.

3.4.5.2 Introducción

El backlog de un producto es una lista de trabajo ordenado por prioridades para el equipo de desarrollo que se obtiene de la hoja de ruta y sus requisitos. Los elementos más importantes se muestran al principio del backlog del producto para que el equipo sepa qué hay que entregar primero. El equipo de desarrollo no trabaja con el backlog al ritmo que dicta el propietario del producto, y este no presiona al equipo de desarrollo para que saque el trabajo adelante. En su lugar, el equipo de desarrollo saca trabajo del backlog del producto en la medida de sus capacidades.

3.4.5.3 Especificar La Correlación Con El O Los Temas Y Subtemas Del Programa De Estudio Vigente.

Esta actividad corresponde al subtema 2.2 La gestión de proyectos usando un marco de calidad.

3.4.5.4 Material Y Equipo Necesario

1. Hojas blancas tamaño carta.
2. Lápiz de grafito.
3. Goma de Borrar
4. Sacapuntas

5. Equipo de computo
6. Internet
7. Microsoft Excel

3.4.5.5 Metodología

Paso 1: Definición del Equipo de Proyecto

Reúne al equipo de proyecto, que incluye al Gerente de Proyecto, representantes del cliente, analistas de negocio y miembros del equipo de desarrollo. Asegúrate de que todos los interesados clave estén presentes.

Paso 2: Identificación de Requisitos Iniciales

Realiza una sesión de lluvia de ideas para identificar los requisitos y características iniciales del software. Registra todas las ideas y requerimientos en una lista inicial.

Paso 3: Priorización de Requisitos

Utiliza técnicas de priorización, como el método MoSCoW (Must have, Should have, Could have, Won't have), para categorizar los requisitos en función de su importancia y necesidad.

Paso 4: Descomposición en Tareas

Divide los requisitos prioritarios en tareas más pequeñas y manejables. Cada tarea debe ser lo suficientemente específica como para ser implementada por el equipo de desarrollo.

Paso 5: Estimación de Tareas

Estima el esfuerzo y el tiempo necesario para completar cada tarea. Puedes utilizar técnicas como la estimación por puntos de historia o la estimación por horas.

Paso 6: Creación del Backlog

Organiza las tareas estimadas en un Backlog de la Lista de Tareas del Proyecto de Software. Este backlog debe incluir una descripción clara de cada tarea, su prioridad, la estimación de esfuerzo y cualquier dependencia entre tareas.

Paso 7: Validación con el Cliente

Presenta el Backlog al cliente o patrocinador del proyecto para su revisión y validación. Asegúrate de que los requisitos y prioridades sean consistentes con las expectativas del cliente.

Paso 8: Refinamiento Continuo

Realiza reuniones de refinamiento del backlog de forma regular para agregar, eliminar o modificar tareas según sea necesario a lo largo del proyecto. Esto garantiza que el backlog esté siempre actualizado.

Paso 9: Asignación de Tareas y Sprints

En el contexto de metodologías ágiles como Scrum, asigna tareas del backlog a sprints específicos, definiendo así el alcance de cada iteración de desarrollo.

Paso 10: Seguimiento y Gestión

A medida que el proyecto avance, realiza un seguimiento constante del backlog para asegurarte de que las tareas se completen según lo planeado. Gestiona cualquier cambio o desviación.

Paso 11: Comunicación y Transparencia

Mantén una comunicación abierta y transparente con todos los interesados, manteniéndolos informados sobre el estado y los cambios en el backlog.

Paso 12: Evaluación Final y Lecciones Aprendidas

Al final del proyecto, evalúa el éxito del backlog en términos de cumplimiento de objetivos y requisitos. Documenta las lecciones aprendidas para mejorar futuros proyectos.

3.4.5.6 Sugerencias Didácticas

- **Introducción al Proyecto:** Comienza la práctica con una introducción al proyecto de software, proporcionando información de contexto sobre su propósito, objetivos y partes interesadas.
- **Explicación del Backlog:** Presenta una explicación clara del concepto de Backlog de la Lista de Tareas del Proyecto y su importancia en la gestión ágil de proyectos de software.
- **Formación de Equipos:** Divide a los estudiantes en equipos pequeños, asignando a cada equipo un proyecto de software ficticio o real para el cual elaborarán el backlog.
- **Análisis de Acta de Constitución del Proyecto:** Proporciona a cada equipo el Acta de Constitución del Proyecto correspondiente al proyecto asignado. Los equipos deben analizarla detenidamente para comprender los objetivos y requisitos.
- **Identificación de Requisitos y Tareas:** Guía a los equipos en la identificación de requisitos y su descomposición en tareas específicas. Pueden utilizar técnicas de lluvia de ideas y mapas mentales.

3.4.5.7 Reporte Del Alumno

El alumno debe de realizar la actividad detallando paso a paso la elaboración de esta, incluyendo capturas, mediante el formato de un reporte de prácticas dando detalle de los resultados obtenidos, así como su conclusión y aprendizajes obtenidos.

3.4.5.8 Bibliografías

- Konrad Mike, Shrum Sandy, CMMI (2ª ed.): Guía para la integración de procesos y la mejora de productos. Madrid: Addison Wesley. (ISBN 9788478290963)

3.4.6 Práctica 6 Elaborar las épicas que formaran parte de proyecto de software a gestionar

3.4.6.1 Objetivo

El objetivo de esta práctica es elaborar las Épicas que formarán parte del proyecto de software a gestionar. Esta actividad tiene como propósito principal identificar y definir las metas de alto nivel, objetivos y funcionalidades clave del proyecto, proporcionando una visión global y clara de lo que se busca lograr. Las Épicas servirán como elementos fundamentales para la planificación, seguimiento y comunicación del proyecto de software.

3.4.6.2 Introducción

Las épicas son definiciones generales de la necesidades y deseos del cliente del producto o servicio que desean obtener en la medida que el proyecto avanza por lo cual, el origen de las épicas se presenta en la visión del producto y los requerimientos del cliente los cuales vienen siendo la expresión de forma general de una necesidad que desean suplir y que se encuentran dentro del alcance de la visión del producto por consiguiente, durante la definición de las épicas es posible identificar algún riesgos que puede presentar el proyecto durante su ejecución. Sin embargo, no son suficientes para determinar cuánto tiempo y esfuerzo se requiere para su Implementación ya que son expresiones de las necesidades de forma general por lo cual, cada épica se conforma de un conjunto de Historias de usuario las cuales pueden ser estimadas para entregar en un tiempo determinado.

.

3.4.6.3 Especificar La Correlación Con El O Los Temas Y Subtemas Del Programa De Estudio Vigente.

Esta actividad corresponde al subtema 2.3 Estándares y Métricas de calidad en la ingeniería de software.

3.4.6.4 Material Y Equipo Necesario

1. Hojas blancas tamaño carta.
2. Lápiz de grafito.
3. Goma de Borrar
4. Sacapuntas
5. Equipo de computo
6. Internet
7. Microsoft Excel

3.4.6.5 Metodología

Introducción al Proyecto: Comienza la práctica con una introducción al proyecto de software. Explica su propósito, importancia y contexto.

Formación de Equipos: Divide a los estudiantes en equipos pequeños, asignando a cada equipo un proyecto de software ficticio o real para el cual elaborarán las Épicas.

Explicación de Épicas: Presenta una explicación detallada de lo que son las Épicas en el contexto de la gestión de proyectos de software. Destaca su importancia como objetivos de alto nivel.

Identificación de Requisitos y Funcionalidades: Instruye a los equipos para que realicen sesiones de lluvia de ideas o análisis de requisitos a fin de identificar las funcionalidades clave y los objetivos del proyecto. Los equipos pueden utilizar entrevistas ficticias con el cliente o patrocinador del proyecto.

Priorización y Jerarquización: Ayuda a los equipos a priorizar y jerarquizar las funcionalidades identificadas, identificando aquellas que son esenciales y estratégicas para el éxito del proyecto.

Documentación de Épicas: Cada equipo debe documentar las Épicas en detalle. Cada Épica debe incluir una descripción, los objetivos estratégicos que aborda y cualquier requisito de alto nivel.

Validación con el Cliente o Patrocinador Ficticio: Organiza una sesión de validación ficticia donde los equipos presenten sus Épicas al cliente o patrocinador del proyecto. Esto añadirá un componente realista a la práctica.

Refinamiento y Retroalimentación: Tras la validación, permite que los equipos refinen sus Épicas según la retroalimentación del "cliente". Pueden hacer ajustes en función de las prioridades y los requisitos adicionales.

Presentación de Épicas: Cada equipo debe presentar sus Épicas ante la clase, explicando la razón detrás de sus decisiones de priorización y cómo cada Épica contribuye al éxito del proyecto.

Evaluación y Retroalimentación: Evalúa el trabajo de los estudiantes considerando la calidad de las Épicas y la claridad de su presentación. Recopila retroalimentación de los estudiantes sobre la práctica.

Lecciones Aprendidas: Concluye la práctica con una discusión sobre las lecciones aprendidas en cuanto a la elaboración de Épicas y cómo este proceso es crucial para la planificación y gestión efectiva de proyectos de software.

3.4.6.6 Sugerencias Didácticas

- **Colaboración en Equipos:** Fomenta la colaboración y el trabajo en equipo al asignar proyectos a grupos pequeños de estudiantes. Esto les ayudará a aprender a comunicarse y colaborar eficazmente en la definición de Épicas.
- **Validación y Retroalimentación Ficticia:** Simula una validación y retroalimentación por parte de un "cliente" o "patrocinador" ficticio para que

los estudiantes experimenten cómo se presentan y defienden las Épicas ante los interesados clave.

- **Uso de Herramientas de Gestión:** Introduce a los estudiantes en herramientas de gestión de proyectos o software de colaboración que les permitan documentar y organizar las Épicas de manera efectiva.
- **Discusión y Reflexión:** Promueve una discusión en clase después de las presentaciones de Épicas para que los estudiantes compartan sus experiencias y reflexionen sobre la importancia de las Épicas en la planificación de proyectos de software.

3.4.6.7 Reporte Del Alumno

El alumno debe de realizar la actividad detallando paso a paso la elaboración de esta, incluyendo capturas, mediante el formato de un reporte de prácticas dando detalle de los resultados obtenidos, así como su conclusión y aprendizajes obtenidos.

3.4.6.8 Bibliografías

- Konrad Mike, Shrum Sandy, CMMI (2ª ed.): Guía para la integración de procesos y la mejora de productos. Madrid: Addison Wesley. (ISBN 9788478290963)

3.4.7 Práctica 7 Elaborar las historias de usuarios que formaran parte de proyecto de software a gestionar

3.4.7.1 Objetivo

El objetivo de esta práctica es elaborar las Historias de Usuarios que formarán parte del proyecto de software a gestionar. La meta principal es identificar, definir y documentar de manera detallada los requisitos y funcionalidades específicas que el software debe cumplir, desde la perspectiva del usuario. Esto permitirá una comprensión más profunda de las necesidades de los

usuarios y servirá como base para la planificación y desarrollo del proyecto de software.

3.4.7.2 Introducción

Una historia de usuario es una explicación general e informal de una función de software escrita desde la perspectiva del usuario final o cliente. El propósito de una historia de usuario es articular cómo un elemento de trabajo entregará un valor particular al cliente. Ten en cuenta que los "clientes" no tienen por qué ser usuarios finales externos en el sentido tradicional, también pueden ser clientes internos o colegas dentro de tu organización que dependen de tu equipo. Las historias de usuario son unas pocas frases en lenguaje sencillo que describen el resultado deseado. No entran en detalles, ya que los requisitos se añaden más tarde, una vez acordados por el equipo.

3.4.7.3 Especificar La Correlación Con El O Los Temas Y Subtemas Del Programa De Estudio Vigente.

Esta actividad corresponde al subtema 2.4 Impacto de la calidad en tiempo, costo y alcance del proyecto.

3.4.7.4 Material Y Equipo Necesario

1. Hojas blancas tamaño carta.
2. Lápiz de grafito.
3. Goma de Borrar
4. Sacapuntas
5. Equipo de computo
6. Internet
7. Microsoft Excel

3.4.7.5 Metodología

Definición del Alcance del Proyecto: Comienza por definir claramente el alcance del proyecto de software. Esto incluye identificar a los interesados clave, comprender los objetivos del proyecto y establecer los límites del mismo.

Identificación de Interesados y Usuarios Finales: Identifica a todas las partes interesadas del proyecto, incluyendo a los usuarios finales que interactuarán con el software. Comprende sus necesidades, expectativas y requerimientos.

Creación de Equipos de Trabajo: Forma equipos de trabajo que incluyan a miembros del equipo de desarrollo, representantes de usuarios y otros interesados relevantes. Cada equipo se centrará en un área funcional o característica del software.

Sesiones de Brainstorming: Realiza sesiones de brainstorming en equipos para identificar funcionalidades y características específicas del software. Anima a los equipos a pensar en términos de lo que los usuarios desean lograr con el software.

Definición de Historias de Usuarios: Con base en las sesiones de brainstorming, cada equipo debe definir Historias de Usuarios. Cada historia debe ser breve, centrarse en una sola funcionalidad o necesidad del usuario y seguir el formato "Como [tipo de usuario], quiero [realizar una acción] para [beneficio o objetivo]".

Priorización y Evaluación: Los equipos deben priorizar las Historias de Usuarios en función de su importancia y valor para los usuarios finales. Pueden utilizar métodos como la técnica MoSCoW o la puntuación de valor.

Detalle de Historias de Usuarios: Las Historias de Usuarios deben ser detalladas, incluyendo criterios de aceptación que describan cómo se verificará que la funcionalidad se ha implementado correctamente.

Revisión y Validación: Realiza una revisión y validación de las Historias de Usuarios con los interesados y usuarios finales para asegurarte de que reflejen con precisión sus necesidades y expectativas.

Documentación y Organización: Documenta las Historias de Usuarios en un formato accesible, como tarjetas físicas o entradas en una herramienta de gestión de proyectos. Organiza las historias en un backlog priorizado.

Planificación de Iteraciones: Si se utiliza una metodología ágil, planifica las iteraciones (sprints) en las que se desarrollarán las Historias de Usuarios seleccionadas para la implementación.

Desarrollo y Seguimiento: Lleva a cabo el desarrollo de las Historias de Usuarios en las iteraciones planificadas, realizando un seguimiento constante para garantizar que se cumplan los criterios de aceptación.

Comunicación Continua: Mantén una comunicación continua con los interesados y usuarios finales para garantizar que las Historias de Usuarios sigan siendo relevantes a medida que avanza el proyecto.

3.4.7.6 Sugerencias Didácticas

Ejemplos de Historias de Usuarios: Comienza la práctica proporcionando ejemplos claros de Historias de Usuarios y cómo se estructuran. Esto ayudará a los estudiantes a comprender el formato y el propósito de las historias.

Trabajo en Equipos Multidisciplinarios: Fomenta la formación de equipos que incluyan a estudiantes con diferentes habilidades, como desarrolladores, diseñadores y usuarios finales ficticios. Esto simula la colaboración multidisciplinaria en proyectos reales.

Validación Ficticia de Usuarios: Organiza una sesión de validación ficticia donde los equipos presenten sus Historias de Usuarios a otros equipos que representan a los usuarios finales. Esto les permitirá experimentar la retroalimentación y el proceso de validación.

Herramientas de Gestión de Proyectos: Introduce a los estudiantes en herramientas de gestión de proyectos o software específico para crear y gestionar Historias de Usuarios. Esto les preparará para el uso de herramientas en entornos profesionales.

3.4.7.7 Reporte Del Alumno

El alumno debe de realizar la actividad detallando paso a paso la elaboración de esta, incluyendo capturas, mediante el formato de un reporte de prácticas dando detalle de los resultados obtenidos, así como su conclusión y aprendizajes obtenidos.

3.4.7.8 Bibliografías

- Konrad Mike, Shrum Sandy, CMMI (2ª ed.): Guía para la integración de procesos y la mejora de productos. Madrid: Addison Wesley. (ISBN 9788478290963)

3.4.8 Práctica 8 Elaborar el registro de interesados del proyecto de software a gestionar

3.4.8.1 Objetivo

El objetivo de esta práctica es elaborar el Registro de Interesados del proyecto de software a gestionar. La meta principal es identificar, documentar y comprender a todas las partes interesadas, tanto internas como externas, que pueden influir o verse afectadas por el proyecto de software. El Registro de Interesados servirá como una herramienta fundamental para la gestión de las expectativas, la comunicación efectiva y la toma de decisiones informadas durante todo el ciclo de vida del proyecto.

3.4.8.2 Introducción

La Gestión de los interesados del proyecto (Stakeholders), es una función que cada vez adquiere mayor importancia en la Gerencia de proyectos, pues ha quedado demostrado que lograr la participación eficaz de los interesados en la ejecución y toma de decisiones es fundamental para el éxito.

En la gestión de proyectos un elemento fundamental es el registro de interesados es una herramienta importante para la gestión de proyectos. Ayuda a

identificar a todas las partes interesadas en el proyecto y a proporcionar información relevante sobre ellas. Con un registro de interesados, los gerentes de proyectos pueden asegurarse de que todos los interesados estén informados, involucrados y comprometidos con el proyecto.

Además, un registro de interesados puede ayudar a gestionar las expectativas de los interesados. Cuando los interesados están bien informados, es menos probable que se sientan frustrados o decepcionados con el resultado final del proyecto. También es más probable que los interesados brinden su apoyo y recursos necesarios para el éxito del proyecto.

3.4.8.3 Especificar La Correlación Con El O Los Temas Y Subtemas Del Programa De Estudio Vigente.

Esta actividad corresponde al subtema 3.4 Estimación de personal requerido.

3.4.8.4 Material Y Equipo Necesario

1. Hojas blancas tamaño carta.
2. Lápiz de grafito.
3. Goma de Borrar
4. Sacapuntas
5. Equipo de computo
6. Internet
7. Microsoft Excel

3.4.8.5 Metodología

Definición del Alcance del Proyecto: Comienza por definir claramente el alcance del proyecto de software. Esto te ayudará a identificar quiénes pueden ser los posibles interesados y cómo podrían estar involucrados.

Identificación de Interesados Potenciales: Realiza una investigación exhaustiva para identificar a todos los interesados potenciales del proyecto. Esto

incluye a los patrocinadores, usuarios finales, equipos de desarrollo, equipos de prueba, gerentes, reguladores, entre otros.

Categorización de Interesados: Clasifica a los interesados en categorías según su nivel de influencia en el proyecto y su impacto en él. Puedes utilizar una matriz de poder/interés para esto.

Recopilación de Información: Reúne información detallada sobre cada interesado, incluyendo su nombre, cargo, organización, roles, necesidades, expectativas, intereses, influencia y prioridades.

Definición de Estrategias de Compromiso: Para cada categoría de interesados, define estrategias de compromiso que describan cómo planeas interactuar y comunicarte con ellos a lo largo del proyecto. Esto puede incluir reuniones regulares, informes, actualizaciones de estado, etc.

Documentación del Registro de Interesados: Documenta toda la información recopilada en un Registro de Interesados estructurado. Puedes utilizar una hoja de cálculo, una herramienta de gestión de proyectos o una plantilla específica para esto.

Validación del Registro: Valida el Registro de Interesados con el equipo de proyecto y otros interesados clave para asegurarte de que esté completo y preciso.

Mantenimiento Continuo: A lo largo del proyecto, actualiza el Registro de Interesados a medida que cambien las circunstancias, se identifiquen nuevos interesados o evolucionen sus necesidades y expectativas.

Comunicación y Gestión de Interesados: Utiliza el Registro de Interesados como una herramienta activa para gestionar la comunicación y las relaciones con los interesados. Mantén un canal abierto y efectivo de comunicación con ellos.

Monitoreo y Control: Monitorea y controla la participación y el compromiso de los interesados a lo largo del proyecto. Ajusta las estrategias de compromiso según sea necesario para mantener su apoyo.

Lecciones Aprendidas: Al final del proyecto, reflexiona sobre cómo se gestionaron los interesados y si las estrategias de compromiso fueron efectivas. Documenta las lecciones aprendidas para proyectos futuros.

3.4.8.6 Sugerencias Didácticas

Estudio de Caso Práctico: Proporciona a los estudiantes un estudio de caso de un proyecto de software ficticio o real e invítalos a identificar y documentar a los interesados potenciales. Esto les dará una experiencia práctica y contextualizada.

Simulación de Comunicación con Interesados: Simula situaciones de comunicación con interesados a través de representaciones de roles. Los estudiantes pueden practicar cómo interactuar con diferentes tipos de interesados en situaciones realistas.

Utilización de Herramientas: Introduce a los estudiantes en el uso de herramientas de gestión de proyectos que faciliten la creación y gestión del Registro de Interesados, como hojas de cálculo o software de gestión de proyectos.

Ejercicios de Actualización Continua: A medida que avanza la práctica, plantea ejercicios adicionales para actualizar el Registro de Interesados en función de cambios en el proyecto, nuevas identificaciones de interesados o modificaciones en sus necesidades y expectativas.

3.4.8.7 Reporte Del Alumno

El alumno debe de realizar la actividad detallando paso a paso la elaboración de esta, incluyendo capturas, mediante el formato de un reporte de prácticas dando detalle de los resultados obtenidos, así como su conclusión y aprendizajes obtenidos.

3.4.8.8 Bibliografías

- Konrad Mike, Shrum Sandy, CMMI (2ª ed.): Guía para la integración de procesos y la mejora de productos. Madrid: Addison Wesley. (ISBN 9788478290963)
- Chemuturi, M. & Caglet, T. M. (2010). Mastering Software Project Management: Best Practices, Tools and Techniques. USA: J. Ross Publishing

3.4.9 Práctica 9 Elaborar la estimación de tareas que integran al proyecto de software a gestionar

3.4.9.1 Objetivo

El objetivo de esta práctica es elaborar la estimación de tareas que forman parte del proyecto de software a gestionar. El propósito principal es calcular y asignar de manera precisa los recursos, tiempo y esfuerzo requeridos para llevar a cabo cada tarea del proyecto. La estimación de tareas es esencial para la planificación, programación y gestión eficiente del proyecto de software, lo que contribuye al cumplimiento exitoso de los objetivos del proyecto.

3.4.9.2 Introducción

Una tarea tiene asignada una persona para su realización, y es recomendable que el esfuerzo estimado parara llevarla a cabo sea como máximo el equivalente al realizable en una jornada de trabajo. Los equipos de trabajo ágiles descomponen las historias de usuario se descomponen en tareas para gestionar y seguir el avance de su ejecución.

Estimar es la actividad de predecir lo que una pieza de trabajo requerirá en términos de tiempo, recursos y costo. Esto puede tener un rango desde un estimado de alto nivel de un proyecto o programa hasta estimar en detalle las actividades individuales en un paquete de trabajo.

Toda tarea debe tener una estimación de tareas puede ser un «arte», pero piense en estos conceptos: los riesgos, la repetitividad, el esfuerzo y la prueba de las tareas; nos ayudará a tener una visión más general del desempeño de nuestro trabajo y su calidad.

3.4.9.3 Especificar La Correlación Con El O Los Temas Y Subtemas Del Programa De Estudio Vigente.

Esta actividad corresponde al subtema 3.2 Estimaciones de tiempo.

3.4.9.4 Material Y Equipo Necesario

1. Hojas blancas tamaño carta.
2. Lápiz de grafito.
3. Goma de Borrar
4. Sacapuntas
5. Equipo de computo
6. Internet
7. Microsoft Excel

3.4.9.5 Metodología

Desglose de Tareas: Divide el proyecto en tareas más pequeñas y manejables. Estas tareas deben ser lo suficientemente específicas como para estimar su duración y recursos necesarios de manera precisa.

Identificación de Dependencias: Identifica las dependencias entre las tareas. Algunas tareas pueden ser secuenciales, mientras que otras pueden realizarse en paralelo. Esto afectará la planificación y las estimaciones.

Asignación de Recursos: Determina qué recursos se necesitan para cada tarea, ya sean personas, equipos, hardware o software. Considera la disponibilidad de recursos y asigna responsabilidades.

Estimación de Duración: Estima el tiempo que llevará completar cada tarea. Utiliza técnicas como la estimación por expertos, la analogía histórica o la estimación paramétrica, según corresponda.

Estimación de Esfuerzo: Calcula el esfuerzo necesario para realizar cada tarea, que puede estar relacionado con el trabajo humano y los recursos técnicos. Asegúrate de considerar todos los aspectos involucrados.

Contingencias y Reservas: Incluye un margen de contingencia en las estimaciones para tener en cuenta imprevistos y riesgos. También considera reservas de tiempo para tareas críticas.

Revisión y Validación: Revisa y valida las estimaciones con el equipo de proyecto y otros expertos relevantes. Asegúrate de que las estimaciones sean realistas y respaldadas por datos o experiencia.

Documentación de Estimaciones: Documenta todas las estimaciones en un formato claro y accesible. Esto incluye la duración, el esfuerzo, los recursos asignados y cualquier supuesto clave.

Seguimiento y Actualización: A lo largo del proyecto, realiza un seguimiento constante de las tareas y sus estimaciones. Actualiza las estimaciones si surgen cambios en el alcance, los riesgos o las condiciones del proyecto.

Comunicación y Transparencia: Comunica las estimaciones a todas las partes interesadas, incluyendo al equipo de proyecto, la dirección y los clientes. La transparencia en las estimaciones es crucial para la gestión efectiva del proyecto.

Aprendizaje Continuo: Al final del proyecto, revisa las estimaciones realizadas en comparación con los resultados reales. Esto proporcionará lecciones aprendidas que pueden aplicarse en proyectos futuros para mejorar las estimaciones.

3.4.9.6 Sugerencias Didácticas

Estudios de Casos Prácticos: Proporciona a los estudiantes ejemplos de proyectos de software reales o ficticios y pídeles que desglosen las tareas y realicen estimaciones. Esto les permitirá aplicar conceptos de estimación en situaciones concretas.

Juegos de Simulación: Diseña juegos de simulación donde los estudiantes trabajen en equipos para estimar tareas bajo condiciones de tiempo limitado y recursos finitos. Esto fomentará la toma de decisiones rápida y la colaboración.

Uso de Herramientas de Estimación: Introduce a los estudiantes en el uso de herramientas de software de estimación, como herramientas de gestión de proyectos o software de estimación específico. Esto les proporcionará experiencia práctica con herramientas que se utilizan en la industria.

Evaluación y Retroalimentación Continua: Proporciona ejercicios de estimación periódicos y retroalimentación constante sobre la calidad de las estimaciones. Los estudiantes pueden aprender de sus errores y mejorar sus habilidades de estimación con el tiempo.

3.4.9.7 Reporte Del Alumno

El alumno debe de realizar la actividad detallando paso a paso la elaboración de esta, incluyendo capturas, mediante el formato de un reporte de prácticas dando detalle de los resultados obtenidos, así como su conclusión y aprendizajes obtenidos.

3.4.9.8 Bibliografías

- Konrad Mike, Shrum Sandy, CMMI (2ª ed.): Guía para la integración de procesos y la mejora de productos. Madrid: Addison Wesley. (ISBN 9788478290963)
- Chemuturi, M. & Caglet, T. M. (2010). Mastering Software Project Management: Best Practices, Tools and Techniques. USA: J. Ross Publishing

3.4.10 Práctica 6 Elaborar los Sprint que formaran parte del proyecto de software a gestionar

3.4.10.1 Objetivo

El objetivo de esta práctica es elaborar los Sprint que formarán parte del proyecto de software a gestionar. La meta principal es dividir el proyecto en iteraciones más pequeñas y manejables, conocidas como Sprints, que permitan un desarrollo ágil y una entrega incremental del software. Al final de esta práctica, los equipos deberán haber planificado y estructurado los Sprints para llevar a cabo la implementación efectiva del proyecto de software.

3.4.10.2 Introducción

Un sprint es un período breve de tiempo fijo en el que un equipo de scrum trabaja para completar una cantidad de trabajo establecida. Los sprints se encuentran en el corazón de las metodologías scrum y ágil, y hacer bien los sprints ayudarán a tu equipo ágil a lanzar mejor software con menos quebraderos de cabeza.

En cada Sprint o cada ciclo de trabajo lo que vamos a conseguir es lo que se denomina un entregable o incremento del producto, que aporte valor al cliente.

Por lo general, el Sprint no es inferior a una semana, ni superior a un mes. A su vez, dentro de cada Sprint se fija un objetivo, que será desarrollado en ese periodo con un método y un plan determinado.

3.4.10.3 Especificar La Correlación Con El O Los Temas Y Subtemas Del Programa De Estudio Vigente.

Esta actividad corresponde al subtema 3.2 Estimaciones de tiempo.

3.4.10.4 Material Y Equipo Necesario

1. Hojas blancas tamaño carta.
2. Lápiz de grafito.
3. Goma de Borrar

4. Sacapuntas
5. Equipo de computo
6. Internet
7. Microsoft Excel

3.4.10.5 Metodología

Definición del Objetivo del Sprint: Comienza cada Sprint con una clara definición de los objetivos y metas que se deben alcanzar durante esa iteración. Esto asegurará que el equipo tenga una dirección clara.

Selección de Historias de Usuario: Identifica y selecciona las Historias de Usuario y las funcionalidades que se abordarán durante el Sprint. Estas historias deben ser priorizadas y estar listas para ser desarrolladas.

Estimación de Tareas: Divide las Historias de Usuario seleccionadas en tareas más pequeñas y estimables. Estima el tiempo y los recursos necesarios para cada tarea.

Planificación de Iteración: Organiza las tareas estimadas en un plan de iteración que detalle cómo se llevarán a cabo a lo largo del Sprint. Asigna responsabilidades y fechas de entrega.

Definición de Criterios de Aceptación: Para cada Historia de Usuario, establece criterios de aceptación claros que definan cuándo se considerará que una tarea o historia está completada.

Desarrollo y Pruebas Continuas: Durante el Sprint, el equipo de desarrollo llevará a cabo la implementación de las tareas planificadas. Las pruebas se realizan de manera continua para garantizar la calidad.

Reuniones Diarias de Scrum: Realiza reuniones diarias de Scrum para mantener al equipo informado sobre el progreso, identificar obstáculos y ajustar el plan si es necesario.

Demostración del Producto: Al final del Sprint, organiza una demostración del producto o de las funcionalidades desarrolladas para el cliente o el equipo de negocio.

Revisión y Retroalimentación: Tras la demostración, recopila la retroalimentación del cliente o del equipo de negocio. Esto ayudará a refinar y ajustar las prioridades para futuros Sprints.

Retrospectiva del Sprint: Realiza una retrospectiva del Sprint con el equipo de desarrollo para analizar lo que funcionó bien y lo que se puede mejorar en términos de procesos y colaboración.

Documentación y Lecciones Aprendidas: Documenta los resultados del Sprint, incluyendo cualquier ajuste en las Historias de Usuario o el plan de iteración. Registra las lecciones aprendidas para futuros Sprints.

Inicio de un Nuevo Sprint: Con base en la retroalimentación y la revisión del Sprint anterior, comienza un nuevo Sprint con nuevos objetivos y prioridades.

3.4.10.6 Sugerencias Didácticas

Simulación de Desarrollo Ágil: Organiza una simulación de desarrollo ágil en la que los estudiantes trabajen en equipos para planificar y ejecutar Sprints. Esto les dará una experiencia práctica de cómo funciona el desarrollo ágil en la vida real.

Estudio de Casos Prácticos: Proporciona casos de estudio de proyectos de software reales o ficticios y pide a los estudiantes que planifiquen Sprints basados en las necesidades y los objetivos del proyecto.

Herramientas de Gestión de Proyectos: Introduce a los estudiantes en el uso de herramientas de gestión de proyectos ágiles, como tableros Kanban o software de gestión ágil. Esto les familiarizará con las herramientas utilizadas en la industria.

Revisión de Resultados Anteriores: Anima a los estudiantes a revisar y comparar los resultados de Sprints anteriores con los objetivos y las estimaciones

iniciales. Esto les ayudará a aprender de la experiencia y mejorar en futuros Sprints.

3.4.10.7 Reporte Del Alumno

El alumno debe de realizar la actividad detallando paso a paso la elaboración de esta, incluyendo capturas, mediante el formato de un reporte de prácticas dando detalle de los resultados obtenidos, así como su conclusión y aprendizajes obtenidos.

3.4.10.8 Bibliografías

- Konrad Mike, Shrum Sandy, CMMI (2ª ed.): Guía para la integración de procesos y la mejora de productos. Madrid: Addison Wesley. (ISBN 9788478290963)
- Chemuturi, M. & Caglet, T. M. (2010). Mastering Software Project Management: Best Practices, Tools and Techniques. USA: J. Ross Publishing

3.4.11 Práctica 11 Elaborar el cronograma de actividades del proyecto de software a gestionar

3.4.11.1 Objetivo

El propósito de esta práctica es elaborar el cronograma de actividades del proyecto de software a gestionar. El cronograma tiene como objetivo principal planificar y organizar de manera eficiente el trabajo a lo largo del proyecto, estableciendo fechas de inicio y finalización para cada tarea y actividad. Esto permite una gestión efectiva del tiempo, la asignación de recursos y el seguimiento del progreso, contribuyendo al cumplimiento exitoso de los objetivos del proyecto de software.

3.4.11.2 Introducción

El cronograma de actividades de un proyecto es una herramienta de gestión de proyectos que muestra el listado de tareas necesarias para realizar un proyecto en orden cronológico. Con un cronograma de actividades todos los integrantes de un proyecto pueden visualizar de forma rápida la hoja de ruta del proyecto, los hitos, asignación de tareas individuales e interdependencia entre tareas.

Los cronogramas de trabajo son especialmente útiles en Project management para organizar el tiempo de trabajo del equipo. Al utilizar un cronograma de trabajo podrás conocer mejor la disponibilidad de tiempo de los miembros del equipo. En función del tipo de horario que realice el equipo, puede haber cronogramas de trabajo de horario completo y cronogramas de actividades de trabajo de horario flexible o incluso rotatorio.

3.4.11.3 Especificar La Correlación Con El O Los Temas Y Subtemas Del Programa De Estudio Vigente.

Esta actividad corresponde a los subtemas 3.2 Estimaciones de tiempo, 3.3 Estimaciones de costos, 3.4 Estimación de personal requerido y 3.5 Análisis de riesgos.

3.4.11.4 Material Y Equipo Necesario

1. Hojas blancas tamaño carta.
2. Lápiz de grafito.
3. Goma de Borrar
4. Sacapuntas
5. Equipo de computo
6. Internet
7. Microsoft Project

3.4.11.5 Metodología

Definición del Alcance del Proyecto: Comienza por comprender claramente el alcance del proyecto de software, incluyendo los objetivos, entregables y requisitos.

Desglose de Trabajo: Divide el proyecto en tareas más pequeñas y manejables. Cada tarea debe ser lo suficientemente específica como para ser estimada y monitoreada.

Secuenciación de Tareas: Establece la secuencia lógica de las tareas. Algunas tareas deben realizarse en secuencia, mientras que otras pueden ejecutarse en paralelo.

Estimación de Duración: Estima la duración de cada tarea. Utiliza técnicas de estimación, como la experiencia pasada, la opinión de expertos o la estimación paramétrica.

Asignación de Recursos: Asigna los recursos necesarios para cada tarea, incluyendo personal, hardware y software. Asegúrate de que los recursos estén disponibles en las fechas programadas.

Definición de Dependencias: Identifica las dependencias entre las tareas. Algunas tareas pueden depender de la finalización de otras antes de poder comenzar.

Construcción del Diagrama de Gantt: Utiliza un software de gestión de proyectos o una herramienta de diagrama de Gantt para construir el cronograma. Representa las tareas, sus duraciones y dependencias en el diagrama.

Inclusión de Hitos: Agrega hitos importantes en el cronograma para marcar fechas clave, como la finalización de fases importantes o entregables.

Revisión y Ajuste: Revisa el cronograma con el equipo del proyecto para asegurarte de que sea realista y alcanzable. Realiza ajustes según sea necesario.

Comunicación y Seguimiento: Comunica el cronograma a todas las partes interesadas y establece un sistema de seguimiento para monitorear el progreso y realizar ajustes a medida que avanza el proyecto.

Reservas y Contingencias: Incluye márgenes de contingencia en el cronograma para imprevistos o retrasos potenciales. También considera la disponibilidad de recursos y días no laborables.

Actualización Continua: A lo largo del proyecto, actualiza el cronograma según sea necesario para reflejar cambios en el alcance, las prioridades o las condiciones del proyecto.

Cierre del Proyecto: Al final del proyecto, revisa el cronograma para evaluar el cumplimiento de las fechas planificadas y documenta las lecciones aprendidas para proyectos futuros.

3.4.11.6 Sugerencias Didácticas

Estudio de Casos Prácticos: Proporciona a los estudiantes ejemplos de proyectos de software reales o ficticios y pídeles que desarrollen un cronograma de actividades basado en los requisitos y objetivos del proyecto.

Uso de Herramientas de Gestión de Proyectos: Introduce a los estudiantes en el uso de herramientas de software de gestión de proyectos que les permitan crear y gestionar cronogramas de manera eficiente.

Proyecto de Grupo: Divide a los estudiantes en grupos y asigna a cada grupo un proyecto de software ficticio. Cada grupo deberá elaborar su propio cronograma de actividades, lo que fomentará la colaboración y el trabajo en equipo.

Ejercicio de Ajuste de Cronograma: Proporciona situaciones hipotéticas en las que los estudiantes deban ajustar un cronograma debido a cambios en el alcance o retrasos. Esto les ayudará a desarrollar habilidades de gestión de cambios.

3.4.11.7 Reporte Del Alumno

El alumno debe de realizar la actividad detallando paso a paso la elaboración de esta, incluyendo capturas, mediante el formato de un reporte de prácticas dando detalle de los resultados obtenidos, así como su conclusión y aprendizajes obtenidos.

3.4.11.8 Bibliografías

- Konrad Mike, Shrum Sandy, CMMI (2ª ed.): Guía para la integración de procesos y la mejora de productos. Madrid: Addison Wesley. (ISBN 9788478290963)

3.4.12 Práctica 12 Elaborar los casos de prueba del proyecto de software a gestionar

3.4.12.1 Objetivo

El objetivo de esta práctica es elaborar los casos de prueba del proyecto de software a gestionar. La meta principal es identificar, diseñar y documentar de manera precisa los casos de prueba que permitirán verificar y validar la funcionalidad del software desarrollado. La creación de casos de prueba contribuye a garantizar la calidad, fiabilidad y rendimiento del software, asegurando que cumpla con los requisitos y expectativas de los usuarios.

3.4.12.2 Introducción

Los casos de prueba son los escenarios que se utilizan para medir la funcionalidad de la aplicación a través de un conjunto de ciertas acciones o condiciones para verificar los resultados esperados. En otras palabras, un caso de prueba es un conjunto de acciones ejecutadas para autenticar la funcionalidad de su aplicación de software.

Un caso de prueba consta de varias cosas, como pasos de prueba, datos de prueba y condiciones previas y posteriores desarrolladas para un escenario de prueba particular. Los casos de prueba se pueden aplicar a cualquier aplicación de software. Se puede hacer a través de pruebas manuales y automatizadas o cualquier herramienta de gestión de pruebas.

3.4.12.3 Especificar La Correlación Con El O Los Temas Y Subtemas Del Programa De Estudio Vigente.

Esta actividad corresponde a los subtemas 4.1. Propuesta y 4.2. Lineamientos de comunicación y seguimiento.

3.4.12.4 Material Y Equipo Necesario

1. Hojas blancas tamaño carta.
2. Lápiz de grafito.
3. Goma de Borrar
4. Sacapuntas
5. Equipo de computo
6. Internet
7. Microsoft Word

3.4.12.5 Metodología

Revisión de Requisitos: Comienza por revisar y comprender a fondo los requisitos del software. Esto te ayudará a identificar las funcionalidades clave que deben ser probadas.

Identificación de Escenarios de Prueba: Identifica los diferentes escenarios de prueba, que son situaciones o condiciones bajo las cuales se probará el software. Estos escenarios pueden incluir casos normales de uso, casos de borde y casos de error.

Diseño de Casos de Prueba: Para cada escenario de prueba identificado, diseña casos de prueba específicos que describan los pasos a seguir, los datos de entrada, las acciones a realizar y los resultados esperados.

Especificación de Datos de Prueba: Define los datos de prueba que se utilizarán en cada caso de prueba. Esto incluye datos de entrada que simulan situaciones del mundo real y valores límite que exploran los extremos.

Cobertura de Requisitos: Asegúrate de que tus casos de prueba cubran todos los requisitos y funcionalidades del software. Puedes utilizar una matriz de trazabilidad para hacer un seguimiento de la cobertura.

Priorización de Casos de Prueba: Prioriza los casos de prueba en función de su importancia y riesgo. Esto te ayudará a enfocar tus esfuerzos de prueba en las áreas críticas del software.

Creación de Datos de Prueba: Prepara los datos de prueba necesarios para ejecutar los casos de prueba. Esto puede incluir la creación de conjuntos de datos específicos o la configuración de entornos de prueba.

Ejecución de Pruebas: Ejecuta los casos de prueba en el entorno de prueba y registra los resultados. Asegúrate de seguir los pasos de cada caso de prueba tal como se especificaron.

Documentación de Defectos: Si se encuentran defectos durante la ejecución de las pruebas, documenta de manera detallada cada defecto, incluyendo su descripción, pasos para reproducirlo y su gravedad.

Revisión y Validación: Revisa y valida los resultados de las pruebas con el equipo de desarrollo para confirmar que se han corregido los defectos y que el software funciona como se espera.

Automatización de Pruebas: Si es aplicable, considera la automatización de casos de prueba repetitivos o críticos para acelerar el proceso de prueba.

Informe de Pruebas: Genera un informe de pruebas que incluya una descripción de las pruebas realizadas, los resultados, los defectos encontrados y cualquier hallazgo relevante.

Cierre de Pruebas: Una vez que se ha completado el proceso de prueba y los defectos han sido corregidos y validados, se considera que las pruebas están completas.

Gestión de Cambios: A medida que se realizan cambios en el software, asegúrate de actualizar y mantener los casos de prueba correspondientes.

3.4.12.6 Sugerencias Didácticas

Ejercicios Prácticos: Proporciona a los estudiantes ejercicios prácticos que incluyan requisitos de software ficticios y pídeles que diseñen casos de prueba para cubrir esos requisitos. Esto les ayudará a aplicar los conceptos de diseño de pruebas en situaciones reales.

Uso de Herramientas de Pruebas: Introduce a los estudiantes en el uso de herramientas de automatización de pruebas o herramientas de gestión de pruebas. Esto les permitirá practicar cómo crear y ejecutar casos de prueba en un entorno profesional.

Trabajo en Equipo: Fomenta la colaboración entre estudiantes, permitiéndoles trabajar en equipos para diseñar casos de prueba. Esto refleja la colaboración en equipos de pruebas de software en la industria.

Revisión y Retroalimentación: Organiza sesiones de revisión y retroalimentación, donde los estudiantes pueden presentar y discutir sus casos de prueba con el profesor o con otros compañeros. Esto fomentará la mejora continua y el aprendizaje colaborativo.

3.4.12.7 Reporte Del Alumno

El alumno debe de realizar la actividad detallando paso a paso la elaboración de esta, incluyendo capturas, mediante el formato de un reporte de prácticas dando detalle de los resultados obtenidos, así como su conclusión y aprendizajes obtenidos.

3.4.12.8 Bibliografías

- Pressman, R. S. (2010), Ingeniería del Software un enfoque práctico. México: MC Graw-Hill.
- Chemuturi, M. & Caglet, T. M. (2010). Mastering Software Project Management: Best Practices, Tools and Techniques. USA: J. Ross Publishing

3.4.13 Práctica 13 Elaborar las pruebas de aceptación del proyecto de software a gestionar

3.4.13.1 Objetivo

El objetivo de esta práctica es elaborar las pruebas de aceptación del proyecto de software a gestionar. El propósito principal es diseñar y documentar pruebas que evalúen si el software cumple con los criterios de aceptación definidos por el cliente o las partes interesadas. Las pruebas de aceptación son fundamentales para garantizar que el software sea funcional y cumpla con las expectativas del usuario final antes de su implementación y puesta en producción.

3.4.13.2 Introducción

Las pruebas de aceptación son definidas por el usuario del sistema y preparadas por el equipo de desarrollo, aunque la ejecución y aprobación final corresponden al usuario.

Estas pruebas van dirigidas a comprobar que el sistema cumple los requisitos de funcionamiento esperado, recogidos en el catalogo de requisitos y en los criterios de aceptación del sistema de información, y conseguir así la aceptación final del sistema por parte del usuario.

El responsable de usuarios debe revisar los criterios de aceptación que se especificaron previamente en el plan de pruebas del sistema y, posteriormente, dirigir las pruebas de aceptación final.

3.4.13.3 Especificar La Correlación Con El O Los Temas Y Subtemas Del Programa De Estudio Vigente.

Esta actividad corresponde a los subtemas 4.1. Propuesta y 4.2. Lineamientos de comunicación y seguimiento.

3.4.13.4 Material Y Equipo Necesario

1. Hojas blancas tamaño carta.
2. Lápiz de grafito.
3. Goma de Borrar
4. Sacapuntas
5. Equipo de computo
6. Internet
7. Microsoft Word

3.4.13.5 Metodología

Revisión de Requisitos de Aceptación: Comienza por revisar y comprender los requisitos de aceptación definidos por el cliente o las partes interesadas. Estos requisitos establecen los criterios que el software debe cumplir para ser aceptado.

Identificación de Escenarios de Prueba: Identifica los diferentes escenarios de prueba que deben cubrirse para evaluar los requisitos de aceptación. Cada escenario representa una situación realista de uso del software.

Diseño de Casos de Prueba de Aceptación: Para cada escenario de prueba, diseña casos de prueba específicos que describan los pasos a seguir, los datos de entrada y los resultados esperados para verificar el cumplimiento de los requisitos de aceptación.

Especificación de Datos de Prueba: Define los datos de prueba necesarios para ejecutar los casos de prueba de aceptación. Estos datos deben ser representativos de las condiciones reales de uso del software.

Configuración del Entorno de Prueba: Prepara el entorno de prueba con la configuración necesaria para simular el uso real del software. Esto puede incluir la configuración de bases de datos, servidores, redes, etc.

Ejecución de Pruebas de Aceptación: Ejecuta los casos de prueba de aceptación en el entorno de prueba siguiendo los pasos especificados. Registra los resultados de cada prueba, indicando si se cumplen o no los requisitos de aceptación.

Documentación de Defectos: Si se encuentran defectos durante la ejecución de las pruebas, documenta de manera detallada cada defecto, incluyendo su descripción, pasos para reproducirlo y su gravedad.

Validación con el Cliente o Usuario Final: Una vez completadas las pruebas de aceptación, presenta los resultados al cliente o usuario final para su validación y aprobación. El cliente debe confirmar que el software cumple con sus requisitos.

Aprobación y Entrega del Software: Si el cliente aprueba las pruebas de aceptación, el software está listo para ser entregado y puesto en producción. Si se identifican problemas, se deben abordar y volver a probar.

Documentación de Resultados: Documenta todos los resultados de las pruebas de aceptación, incluyendo los casos de prueba ejecutados, los resultados, los defectos encontrados y cualquier validación o aprobación por parte del cliente.

Seguimiento y Cierre: Realiza un seguimiento para asegurarte de que todos los problemas identificados se han resuelto y que el software cumple con los requisitos de aceptación antes de la entrega final.

3.4.13.6 Sugerencias Didácticas

Estudio de Casos Reales: Proporciona a los estudiantes casos reales de proyectos de software con requisitos de aceptación y pídeles que diseñen pruebas de aceptación basadas en esos requisitos. Esto les dará experiencia práctica con situaciones reales de la industria.

Simulación de Validación con el Cliente: Organiza una simulación en la que los estudiantes representen el papel del cliente o usuario final y validen las

pruebas de aceptación diseñadas por otros equipos. Esto fomentará la comunicación y comprensión entre roles clave.

Uso de Herramientas de Pruebas: Introduce a los estudiantes en el uso de herramientas de gestión de pruebas y automatización de pruebas para crear, ejecutar y gestionar pruebas de aceptación de manera eficiente.

Proyecto de Grupo: Fomenta la colaboración al permitir que los estudiantes trabajen en grupos para diseñar pruebas de aceptación para un proyecto de software ficticio. Esto reflejará la colaboración en equipos de pruebas en la industria.

Revisión y Retroalimentación: Organiza sesiones de revisión y retroalimentación, donde los estudiantes presenten y discutan sus pruebas de aceptación con el profesor o con otros compañeros. Esto fomentará la mejora continua y el aprendizaje colaborativo.

3.4.13.7 Reporte Del Alumno

El alumno debe de realizar la actividad detallando paso a paso la elaboración de esta, incluyendo capturas, mediante el formato de un reporte de prácticas dando detalle de los resultados obtenidos, así como su conclusión y aprendizajes obtenidos.

3.4.13.8 Bibliografías

- Pressman, R. S. (2010), Ingeniería del Software un enfoque práctico. México: MC Graw-Hill.
- Chemuturi, M. & Caglet, T. M. (2010). Mastering Software Project Management: Best Practices, Tools and Techniques. USA: J. Ross Publishing

3.4.14 Práctica 14 Elaborar los criterios de aceptación del proyecto de software a gestionar

3.4.14.1 Objetivo

El objetivo de esta práctica es elaborar los criterios de aceptación del proyecto de software a gestionar. La meta principal es definir de manera clara y detallada los estándares y condiciones que el software debe cumplir para considerarse aceptado por el cliente o las partes interesadas. Establecer criterios de aceptación sólidos es esencial para garantizar que el software cumpla con las expectativas y requisitos del usuario final antes de su implementación y puesta en producción.

3.4.14.2 Introducción

Los criterios de aceptación definen los requisitos de una historia de usuario para la aceptación de los interesados y los clientes. Se pueden definir los criterios de aceptación que utilizarán los equipos para determinar si se ha finalizado la historia de usuario.

A partir del Criterio de Aceptación se suele definir de manera más detallada la documentación del comportamiento esperado de la funcionalidad y de sus pruebas funcionales y técnicas. Se pueden utilizar varios enfoques para plasmar estas pruebas, como el Behaviour-Driven Development o las «Pruebas (funcionales) de Aceptación».

Podemos definir los Criterios de Aceptación como las características que un producto debe cumplir en orden de corroborar que fue desarrollado según las expectativas de los interesados, estas son determinadas en el alcance del proyecto.

3.4.14.3 Especificar La Correlación Con El O Los Temas Y Subtemas Del Programa De Estudio Vigente.

Esta actividad corresponde a los subtemas 4.1. Propuesta y 4.2. Lineamientos de comunicación y seguimiento.

3.4.14.4 Material Y Equipo Necesario

1. Hojas blancas tamaño carta.
2. Lápiz de grafito.
3. Goma de Borrar
4. Sacapuntas
5. Equipo de computo
6. Internet
7. Microsoft Word

3.4.14.5 Metodología

Revisión de Requisitos del Proyecto: Comienza por revisar y comprender los requisitos del proyecto de software, incluyendo los requisitos funcionales y no funcionales.

Identificación de Características Clave: Identifica las características y funcionalidades clave que deben ser evaluadas en términos de su aceptación por parte del cliente o las partes interesadas.

Definición de Criterios de Aceptación: Para cada característica o funcionalidad identificada, define criterios de aceptación claros y específicos que indiquen cuándo se considerará que la característica cumple con las expectativas.

Criterios Cuantitativos y Cualitativos: Los criterios de aceptación pueden incluir elementos cuantitativos (por ejemplo, rendimiento, velocidad de respuesta) y cualitativos (por ejemplo, usabilidad, apariencia visual).

Validación con el Cliente o Usuario Final: Presenta los criterios de aceptación propuestos al cliente o usuario final para su validación y aprobación. Asegúrate de que estén alineados con sus expectativas.

Refinamiento y Ajuste: Si es necesario, ajusta los criterios de aceptación en función de la retroalimentación del cliente o las partes interesadas. Deben reflejar con precisión sus requisitos y expectativas.

Documentación de Criterios de Aceptación: Documenta de manera clara y accesible los criterios de aceptación para cada característica o funcionalidad del software.

Integración en Plan de Pruebas: Integra los criterios de aceptación en el plan de pruebas del proyecto, de modo que sean utilizados como base para las pruebas de aceptación.

Seguimiento y Verificación: A medida que se desarrolla y prueba el software, realiza un seguimiento para asegurarte de que se cumplan los criterios de aceptación definidos.

Validación y Aprobación Final: Una vez que se completa el desarrollo y las pruebas, presenta el software al cliente o usuario final para la validación final de los criterios de aceptación.

Gestión de Cambios: A medida que se realicen cambios en el software o se identifiquen nuevas características, asegúrate de actualizar y ajustar los criterios de aceptación según sea necesario.

Documentación de Resultados: Documenta todos los resultados de las pruebas de aceptación, incluyendo si se cumplen o no los criterios de aceptación y cualquier validación o aprobación por parte del cliente.

3.4.14.6 Sugerencias Didácticas

Estudio de Casos Prácticos: Proporciona a los estudiantes casos de proyectos de software reales o ficticios y pídeles que definan criterios de aceptación para las características o funcionalidades específicas de esos proyectos. Esto les dará experiencia práctica en la elaboración de criterios de aceptación.

Validación con Roles Simulados: Organiza una actividad en la que los estudiantes representen los roles de cliente, usuario final y desarrollador. Pídeles que definan y validen los criterios de aceptación desde las perspectivas de estos diferentes roles.

Revisión de Criterios de Aceptación Existentes: Proporciona ejemplos de criterios de aceptación existentes y pídeles a los estudiantes que los revisen, ajusten y mejoren según sea necesario. Esto les permitirá practicar la mejora continua de los criterios.

Colaboración en Equipos: Fomenta la colaboración entre los estudiantes al permitir que trabajen en equipos para definir criterios de aceptación para un proyecto de software simulado. Esto refleja la colaboración en equipos de desarrollo de software en la vida real.

Retroalimentación y Validación: Organiza sesiones de revisión y validación en las que los estudiantes presenten sus criterios de aceptación a otros estudiantes para obtener retroalimentación y validación de sus enfoques.

3.4.14.7 Reporte Del Alumno

El alumno debe de realizar la actividad detallando paso a paso la elaboración de esta, incluyendo capturas, mediante el formato de un reporte de prácticas dando detalle de los resultados obtenidos, así como su conclusión y aprendizajes obtenidos.

3.4.14.8 Bibliografías

- Pressman, R. S. (2010), Ingeniería del Software un enfoque práctico. México: MC Graw-Hill.
- Chemuturi, M. & Caglet, T. M. (2010). Mastering Software Project Management: Best Practices, Tools and Techniques. USA: J. Ross Publishing

3.4.15 Práctica 15 Elaborar el scrumboard de los entregables del proyecto de software a gestionar

3.4.15.1 Objetivo

El objetivo de esta práctica es elaborar el Scrumboard (tablero Scrum) de los entregables del proyecto de software a gestionar. El propósito principal es visualizar y organizar de manera efectiva las tareas, historias de usuario o elementos de trabajo en un entorno ágil de desarrollo de software. El Scrumboard facilita la planificación, seguimiento y colaboración del equipo, lo que contribuye a la entrega exitosa y a tiempo de los entregables del proyecto.

3.4.15.2 Introducción

Un Scrum Board o tablero Scrum es una representación visual del trabajo a realizar por un equipo Scrum durante una iteración o sprint. Es una herramienta esencial del marco de trabajo de Scrum, que permite seguir la consecución de los users stories (o Product Backlog Items) y las actividades durante un sprint.

Los tableros Scrum o Scrum Boards están destinados a los equipos que planifican su trabajo en sprints: períodos de tiempo para crear un incremento de trabajo que se pueda enviar.

Para el Scrum Board no existen normas formales, cada equipo confecciona su tablero de la manera en que sea más cómoda su gestión, aunque es común que todos sigan un patrón común.

3.4.15.3 Especificar La Correlación Con El O Los Temas Y Subtemas Del Programa De Estudio Vigente.

Esta actividad corresponde al subtema 5.1. Administración de actividades.

3.4.15.4 Material Y Equipo Necesario

1. Hojas blancas tamaño carta.
2. Lápiz de grafito.

3. Goma de Borrar
4. Sacapuntas
5. Equipo de computo
6. Internet
7. Microsoft Word

3.4.15.5 Metodología

Definición de Elementos de Trabajo: Comienza por definir y listar todos los elementos de trabajo necesarios para el proyecto, como historias de usuario, tareas, características o funcionalidades.

Priorización: Prioriza los elementos de trabajo en función de su importancia y valor para el cliente o usuario final. Esto te ayudará a determinar el orden en que se abordarán.

Creación de Tarjetas: Para cada elemento de trabajo, crea tarjetas físicas o virtuales que representen visualmente la tarea o historia de usuario. Cada tarjeta debe incluir información relevante, como una descripción y una estimación de esfuerzo.

Columnas del Scrumboard: Crea columnas en el Scrumboard que representen los estados por los que pasará cada elemento de trabajo. Las columnas típicas incluyen "Por Hacer", "En Progreso" y "Hecho". Puedes personalizar las columnas según las necesidades del proyecto.

Asignación de Responsabilidades: Asigna responsables a cada elemento de trabajo, asegurándote de que haya claridad sobre quién es el encargado de completar cada tarea.

Visualización en el Scrumboard: Coloca las tarjetas en las columnas correspondientes del Scrumboard para indicar el estado actual de cada elemento de trabajo. Por ejemplo, las tarjetas se mueven de "Por Hacer" a "En Progreso" a medida que se trabajan y finalmente a "Hecho" una vez completadas.

Reuniones de Seguimiento: Programa reuniones regulares de seguimiento del proyecto, como las reuniones diarias de Scrum, para revisar el estado del Scrumboard, identificar bloqueos y ajustar la asignación de trabajo si es necesario.

Actualización Continua: Mantén el Scrumboard actualizado de manera continua a medida que el equipo avanza en el proyecto. Esto proporciona una vista en tiempo real del progreso y ayuda a identificar problemas rápidamente.

Gestión de Cambios: Si se requieren cambios en el alcance o los entregables del proyecto, ajusta el Scrumboard en consecuencia y comunica estos cambios al equipo.

Cierre del Proyecto: Una vez que todos los elementos de trabajo se hayan movido a la columna "Hecho" y el proyecto esté completo, cierra el proyecto y realiza una revisión retrospectiva para aprender de la experiencia.

3.4.15.6 Sugerencias Didácticas

Ejercicio de Simulación: Proporciona a los estudiantes una lista de tareas o historias de usuario ficticias y pídeles que creen un Scrumboard con columnas "Por Hacer", "En Progreso" y "Hecho". Esto les permitirá practicar la visualización y la gestión de tareas en un entorno ágil.

Scrum en Equipo: Divide a los estudiantes en equipos pequeños y asigna a cada equipo un proyecto de software ficticio. Pídeles que colaboren para crear un Scrumboard y gestionar el trabajo como un equipo Scrum real.

Herramientas de Scrum Virtuales: Introduce a los estudiantes en herramientas de Scrum virtuales, como Jira o Trello, y pídeles que utilicen estas herramientas para crear y gestionar su Scrumboard. Esto les dará experiencia con herramientas ampliamente utilizadas en la industria.

Reuniones de Seguimiento Simuladas: Organiza reuniones de seguimiento simuladas en las que los estudiantes presenten su Scrumboard y

discutan el progreso y los bloqueos. Esto les ayudará a practicar la comunicación y colaboración en un entorno ágil.

Gestión de Cambios: Introduce cambios ficticios en los proyectos y pídeles a los estudiantes que ajusten su Scrumboard para reflejar estos cambios. Esto les ayudará a comprender cómo gestionar cambios en proyectos ágiles.

3.4.15.7 Reporte Del Alumno

El alumno debe de realizar la actividad detallando paso a paso la elaboración de esta, incluyendo capturas, mediante el formato de un reporte de prácticas dando detalle de los resultados obtenidos, así como su conclusión y aprendizajes obtenidos.

3.4.15.8 Bibliografías

- Pressman, R. S. (2010), Ingeniería del Software un enfoque práctico. México: MC Graw-Hill.
- Chemuturi, M. & Caglet, T. M. (2010). Mastering Software Project Management: Best Practices, Tools and Techniques. USA: J. Ross Publishing

3.4.16 Práctica 16 Elaborar el sprint backlog del proyecto de software a gestionar

3.4.16.1 Objetivo

El objetivo de esta práctica es elaborar el Sprint Backlog del proyecto de software a gestionar. La meta principal es identificar y planificar las tareas y elementos de trabajo específicos que serán abordados durante un sprint dentro de la metodología Scrum. El Sprint Backlog es esencial para la ejecución efectiva del sprint, asegurando que el equipo tenga claridad sobre las actividades a realizar y los objetivos a alcanzar durante ese período de tiempo definido.

3.4.16.2 Introducción

El Sprint Backlog una planificación táctica del trabajo a realizar en la iteración actual. Esta lista permite ver las tareas donde el equipo está teniendo problemas y no avanza, con lo que le permite tomar decisiones al respecto.

Para cada uno de los objetivos/requisitos se muestran sus tareas, el esfuerzo pendiente para finalizarlas y la autoasignación inicial que han hecho los miembros del equipo.

Subconjunto de objetivos/requisitos del Product Backlog seleccionado para la iteración actual y su plan de tareas de desarrollo. El equipo lo elabora en la reunión de planificación de la iteración (Sprint planning) seleccionando lo que prevé que podrá completar y demostrar al cliente al finalizar la iteración, en forma de incremento de producto preparado para ser entregado.

3.4.16.3 Especificar La Correlación Con El O Los Temas Y Subtemas Del Programa De Estudio Vigente.

Esta actividad corresponde al subtema 5.2. Administración del tiempo.

3.4.16.4 Material Y Equipo Necesario

1. Hojas blancas tamaño carta.
2. Lápiz de grafito.
3. Goma de Borrar
4. Sacapuntas
5. Equipo de computo
6. Internet
7. Microsoft Word

3.4.16.5 Metodología

Selección de Elementos de Trabajo: Comienza por seleccionar las historias de usuario o elementos de trabajo más prioritarios y valiosos del Product Backlog para incluir en el próximo sprint.

Revisión de Historias de Usuario: Junto con el equipo Scrum (Scrum Master, Product Owner y miembros del equipo de desarrollo), revisa las historias de usuario seleccionadas para asegurarse de que estén bien definidas y sean comprensibles.

Descomposición de Historias de Usuario: Si alguna historia de usuario es demasiado grande o compleja, descompónla en tareas más pequeñas y manejables que puedan completarse durante el sprint.

Estimación de Tareas: Estima el esfuerzo necesario para completar cada tarea o historia de usuario. Puedes utilizar técnicas como la estimación en puntos de historia o la estimación en horas.

Priorización de Tareas: Prioriza las tareas y elementos de trabajo en función de su importancia y dependencias. Asegúrate de que el equipo tenga claro el orden de trabajo.

Asignación de Tareas: Asigna tareas a los miembros del equipo de desarrollo en función de sus habilidades y capacidades. Cada miembro debe tener responsabilidades claras.

Registro en el Sprint Backlog: Registra todas las tareas y elementos de trabajo seleccionados, descompuestos, estimados y asignados en el Sprint Backlog. Esto puede hacerse en una herramienta de gestión de proyectos o en un tablero físico.

Definición de Objetivos: Define los objetivos específicos que se deben lograr durante el sprint. Estos objetivos deben estar alineados con las historias de usuario y tareas del Sprint Backlog.

Reunión de Planificación del Sprint: Realiza una reunión de planificación de sprint en la que el equipo revisa y confirma el Sprint Backlog y se compromete a completar las tareas durante el sprint.

Ejecución del Sprint: Durante el sprint, el equipo trabaja en las tareas del Sprint Backlog, actualizando el estado de las tareas a medida que avanzan.

Reuniones Diarias: Realiza reuniones diarias de Scrum para revisar el progreso del sprint, identificar bloqueos y ajustar el Sprint Backlog si es necesario.

Revisión del Sprint: Al final del sprint, lleva a cabo una reunión de revisión del sprint en la que el equipo demuestra el trabajo completado y recopila retroalimentación.

Retrospectiva del Sprint: Realiza una retrospectiva del sprint para identificar mejoras y lecciones aprendidas en la gestión del Sprint Backlog.

Cierre del Sprint: Concluye el sprint y ajusta el Sprint Backlog para el próximo sprint en función de las prioridades y cambios.

3.4.16.6 Sugerencias Didácticas

Ejercicio de Planificación de Sprint: Proporciona a los estudiantes una lista de historias de usuario ficticias y pídeles que trabajen en equipo para seleccionar, descomponer, estimar y asignar tareas en un Sprint Backlog simulado. Esto les dará experiencia práctica en la planificación de sprints.

Simulación de Reuniones de Scrum: Organiza simulaciones de reuniones de Scrum, como la reunión de planificación del sprint y las reuniones diarias. Los estudiantes pueden representar los roles de Scrum Master, Product Owner y miembros del equipo de desarrollo para practicar la colaboración y la comunicación.

Uso de Herramientas Scrum: Introduce a los estudiantes en herramientas Scrum, como tableros digitales (por ejemplo, Jira o Trello) y pídeles que utilicen estas herramientas para crear y gestionar su Sprint Backlog. Esto les dará experiencia con herramientas ampliamente utilizadas en la industria.

Retroalimentación Continua: Fomenta la revisión y retroalimentación regular del Sprint Backlog durante el sprint. Los estudiantes pueden identificar y abordar bloqueos o ajustar las tareas según sea necesario.

Revisión y Retrospectiva Simuladas: Al final del sprint, organiza una revisión y una retrospectiva simuladas donde los estudiantes presenten el trabajo

completado y discutan las mejoras que podrían implementar en la próxima iteración.

3.4.16.7 Reporte Del Alumno

El alumno debe de realizar la actividad detallando paso a paso la elaboración de esta, incluyendo capturas, mediante el formato de un reporte de prácticas dando detalle de los resultados obtenidos, así como su conclusión y aprendizajes obtenidos.

3.4.16.8 Bibliografías

- Pressman, R. S. (2010), Ingeniería del Software un enfoque práctico. México: MC Graw-Hill.
- Chemuturi, M. & Caglet, T. M. (2010). Mastering Software Project Management: Best Practices, Tools and Techniques. USA: J. Ross Publishing

3.4.17 Práctica 17 Elaborar el burndown chart del proyecto de software a gestionar

3.4.17.1 Objetivo

El objetivo de esta práctica es elaborar el Burndown Chart del proyecto de software a gestionar. El propósito principal es proporcionar una herramienta visual que permita al equipo de desarrollo y a las partes interesadas seguir el progreso del proyecto de manera efectiva. El Burndown Chart muestra la cantidad de trabajo pendiente frente al tiempo transcurrido y ayuda a identificar posibles desviaciones en la planificación, lo que facilita la toma de decisiones informadas para garantizar la entrega exitosa del proyecto en el tiempo previsto.

3.4.17.2 Introducción

Un Burndown Chart, diagrama de quemado o gráfica de trabajo pendiente es una gráfica en la que podemos ver el estado del progreso de un Sprint. Es una herramienta utilizada en desarrollo ágil de software, sobre todo en Scrum, que relaciona el trabajo pendiente con un periodo de tiempo determinado.

Una gráfica de burndown puede ofrecernos mucha más información que el avance de un Sprint si sabemos interpretarla correctamente y podremos ayudar mucho más a los equipos gracias a esto. Puede aportar tanto valor que merece la pena dedicarle un post entero.

Asimismo, esta herramienta permite dar presentar de forma dinámica entre el equipo del proyecto y las partes interesadas. Además, puede proporcionar información sobre las distintas actividades, a partir de la cual pueden surgir sugerencias de mejora en diferentes aspectos

3.4.17.3 Especificar La Correlación Con El O Los Temas Y Subtemas Del Programa De Estudio Vigente.

Esta actividad corresponde al subtema 5.3. Evaluación y ajustes del proyecto.

3.4.17.4 Material Y Equipo Necesario

1. Hojas blancas tamaño carta.
2. Lápiz de grafito.
3. Goma de Borrar
4. Sacapuntas
5. Equipo de computo
6. Internet
7. Microsoft Excel

3.4.17.5 Metodología

Definición de Tareas o Historias de Usuario: Comienza por definir todas las tareas o historias de usuario que formarán parte del proyecto de software. Estas tareas deben estar bien definidas y ser estimables.

Estimación de Tareas: Estima el esfuerzo necesario para completar cada tarea o historia de usuario. Puedes utilizar técnicas de estimación como la estimación en puntos de historia, la estimación en horas o cualquier otro método que funcione para tu equipo.

Creación del Sprint Backlog: Organiza las tareas estimadas en un Sprint Backlog que indique qué tareas se abordarán en cada sprint. Asegúrate de que el equipo esté comprometido con la cantidad de trabajo planificada para el sprint.

Establecimiento del Eje X del Burndown Chart: En el eje X del Burndown Chart, representa el tiempo en unidades de tiempo (días, semanas, etc.). El eje X representa la duración total del proyecto.

Establecimiento del Eje Y del Burndown Chart: En el eje Y del Burndown Chart, representa la cantidad de trabajo pendiente (por ejemplo, puntos de historia o esfuerzo estimado). El eje Y representa la cantidad total de trabajo del proyecto.

Registros Diarios: A diario, registra la cantidad de trabajo pendiente en el proyecto. Esto se hace sumando el esfuerzo estimado de las tareas que aún no se han completado.

Actualización del Burndown Chart: Actualiza el Burndown Chart a diario con los datos recopilados. Representa la cantidad de trabajo pendiente en función del tiempo transcurrido.

Seguimiento y Análisis: Utiliza el Burndown Chart para realizar un seguimiento del progreso del proyecto. Identifica tendencias y desviaciones con respecto al plan original.

Comunicación del Progreso: Comparte el Burndown Chart con el equipo de desarrollo y las partes interesadas para mantenerlos informados sobre el progreso del proyecto.

Ajustes y Decisiones: Si el Burndown Chart muestra desviaciones significativas en el progreso, utiliza esta información para tomar decisiones informadas, como la reasignación de tareas o la gestión de cambios en el proyecto.

Cierre del Proyecto: Una vez que todo el trabajo planificado se haya completado, el Burndown Chart debe llegar a cero, lo que indica la finalización exitosa del proyecto.

3.4.17.6 Sugerencias Didácticas

Simulación de Proyecto: Proporciona a los estudiantes un proyecto de software ficticio con tareas y estimaciones. Pídeles que utilicen estas tareas para crear un Burndown Chart simulado.

Seguimiento Diario Simulado: Realiza una actividad en la que los estudiantes registren diariamente el progreso de las tareas en el proyecto ficticio y actualicen el Burndown Chart en consecuencia. Esto les dará experiencia práctica en la actualización constante del gráfico.

Análisis de Tendencias: Desafía a los estudiantes a analizar las tendencias en el Burndown Chart simulado y a identificar patrones, como la velocidad del equipo o los problemas que afectan el progreso.

Revisión de Desviaciones: Introduce desviaciones ficticias en el proyecto y pídeles a los estudiantes que ajusten el Burndown Chart de acuerdo con estos cambios. Esto les ayudará a comprender cómo el gráfico refleja cambios en el proyecto.

Comparación con Planificación: Pide a los estudiantes que comparen el progreso real con la planificación inicial y que identifiquen posibles desviaciones. Esto fomentará la toma de decisiones basada en datos.

Presentación y Comunicación: Anima a los estudiantes a presentar y comunicar los resultados del Burndown Chart a otros miembros del equipo o partes interesadas, practicando la comunicación efectiva.

Retroalimentación y Mejora: Organiza sesiones de retroalimentación en las que los estudiantes revisen y discutan el uso del Burndown Chart y propongan mejoras en su aplicación.

3.4.17.7 Reporte Del Alumno

El alumno debe de realizar la actividad detallando paso a paso la elaboración de esta, incluyendo capturas, mediante el formato de un reporte de prácticas dando detalle de los resultados obtenidos, así como su conclusión y aprendizajes obtenidos.

3.4.17.8 Bibliografías

- Pressman, R. S. (2010), Ingeniería del Software un enfoque práctico. México: MC Graw-Hill.
- Chemuturi, M. & Caglet, T. M. (2010). Mastering Software Project Management: Best Practices, Tools and Techniques. USA: J. Ross Publishing

FUENTES DE INFORMACIÓN

ANEXO 1 DEL MANUAL DE PRÁCTICAS

Evaluaciones

ANEXO 2

ANEXO 3

Respuestas De Las Evaluaciones