UNIVERSIDAD TECMILENIO®

Innovación con propósito de vida.

Guía para el Profesor

Procesos de desarrollo de software

ÍNDICE

Certificados

Para entender la importancia del curso del cual usted será **Facilitador**, es necesario ofrecer un contexto mayor sobre el programa de **Certificados** de la Universidad Tecmilenio, pues son parte medular del nuevo modelo educativo basado en el **aprender haciendo** y en **brindar una experiencia educativa a la medida de los alumnos**.

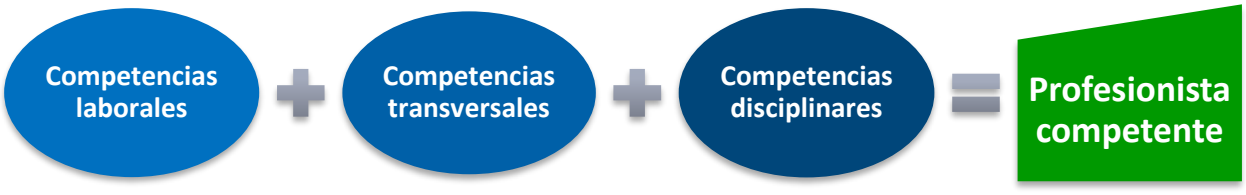
Un certificado es un **programa académico corto compuesto de varias materias,** embebido en la segunda mitad del plan de estudios de profesional, que busca desarrollar **competencias muy específicas** en el alumno y lo prepara para desempeñarse de la mejor manera en un empleo.

SABER + HACER + BIEN

Con este enfoque, buscamos en los egresados de profesional que además de **saber** (tener un conocimiento teórico), también sean **capaces de hacer** (tener la habilidad de realizar una tarea) y de **saber-hacer** (entender lo que se hace y tener la capacidad para hacerlo de la mejor forma), como se explica en este video (https://www.youtube.com/watch?v=g1maCpZXX8s):

Haz clic en la imagen

En Universidad Tecmilenio, **aprender haciendo** significa que el participante cursará **Certificados en los que desarrolla competencias disciplinares de especialidad que son valoradas por el mercado laboral**, convirtiéndose en un profesional altamente competente y elevando así su índice de empleabilidad.



La mayoría de nuestros Certificados se compone en promedio de cuatro materias, las cuales tienen un seguimiento lógico y terminan con un proyecto de gran calado y un alto nivel de complejidad (última materia). Una correcta realización del proyecto integrador demostrará el dominio de la competencia global declarada en cada certificado.

¿Certificado o certificación?

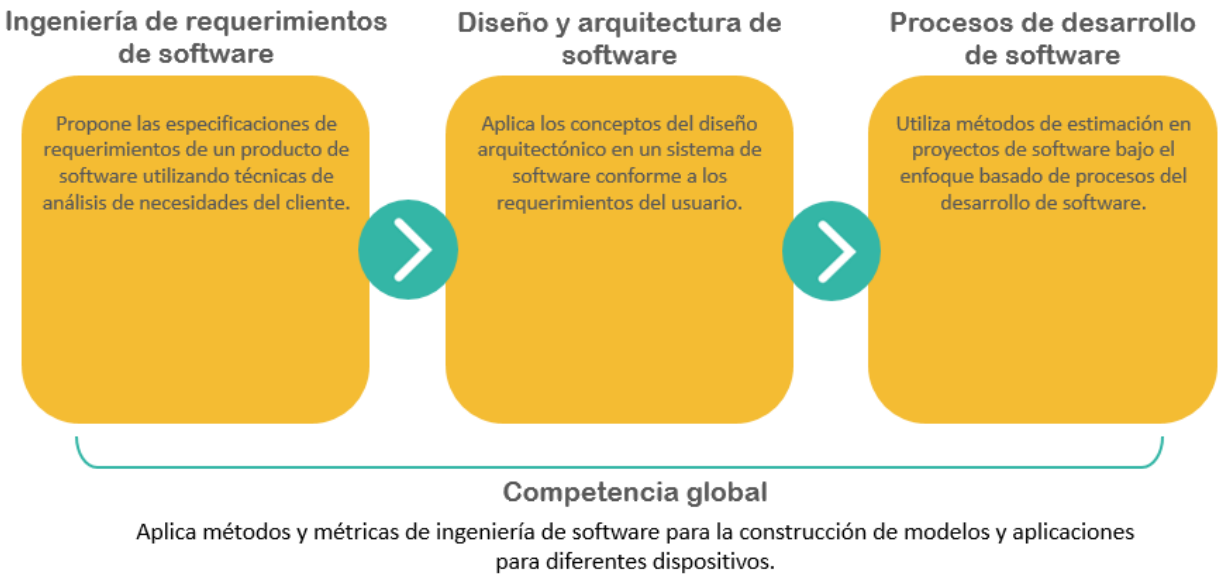
Es muy importante tener en claro que un certificado y una certificación son dos cosas distintas. Un **certificado** es un reconocimiento formal que **otorga internamente la Universidad Tecmilenio** a los estudiantes que demuestren haber aprobado las materias correspondientes, y adquirido la **competencia** global del certificado.

Por su parte, la **certificación** es también un reconocimiento, pero ésta se obtiene a través de la acreditación de un curso específico del programa académico de la Universidad y aprobando un examen de suficiencia aplicado por una **entidad acreditadora externa** (mapas mentales, idiomas, uso de software, etc.).

Su trabajo como docente facilitador de este curso es muy importante para nosotros. Gracias por aportar su conocimiento y experiencia en la impartición de este certificado. A continuación podrá revisar información detallada del curso que impartirá.

Certificado en Ingeniería de software

El certificado de Ingeniería de software se compone de 3 cursos más una materia de proyecto integrador, de acuerdo a la siguiente distribución:



Como se puede apreciar, este curso de **Procesos de desarrollo de software** es el tercer curso del certificado de Ingeniería de software. Por lo mismo, es importante que como **Facilitador verifique** que sus estudiantes hayan aprobado los cursos anteriores, pues de no haberlo hecho se podrá ver afectado el aprovechamiento académico de este curso.

Competencia del certificado

Al finalizar el **certificado de Ingeniería de software**, el participante deberá haber desarrollado y adquirido la siguiente competencia global, en toda su extensión:

> Aplica métodos y métricas de ingeniería de software para la construcción de modelos y aplicaciones para diferentes dispositivos.

Competencia del curso

La competencia específica que el participante habrá de obtener al aprobar satisfactoriamente el **curso de Procesos de desarrollo de software** es la siguiente, en toda su extensión:

> Utiliza métodos de estimación en proyectos de software bajo el enfoque basado de procesos del desarrollo de software.

Metodología del curso

En este curso de **Proceses de desarrollo de software** se revisarán 15 temas divididos en 3 módulos.

En cada tema, el participante encontrará:

- Una breve explicación del tema que ayudará al estudiante a ampliar su conocimiento.
- Una serie de lecturas y videos obligatorios para una mejor comprensión de los temas.
- Una lista de lecturas y videos recomendados para complementar el estudio del tema.
- Una práctica no evaluable que servirá para repasar los conceptos abordados en el tema.
- Una tarea o actividad de aprendizaje (evaluable) cuyo propósito es aplicar y experimentar con los conceptos estudiados.

A lo largo del curso, el participante debe trabajar en lo siguiente:

- 15 actividades
- 1 avances de evidencia
- 1 entrega final de evidencia

Actividades

Las actividades deben enviarse a través de la plataforma Blackboard en la fecha indicada.
Si las actividades se realizaron en forma física (“a mano”), deberán ser digitalizadas para enviarlas a través de dicha plataforma.

Evidencia

El proyecto final (evidencia) de este curso consiste en diseñar el software para una agencia de transporte estimando su tamaño y basándote en el análisis del sistema, además detallarás los elementos del proyecto de desarrollo e implementación necesarios. A través de ella el participante demostrará la capacidad de aplicar los conocimientos y habilidades que obtendrá a lo largo de los temas revisados en el curso. Es importante revisar la agenda del curso, pues la mayoría de las **evidencias requieren entregas de avances** que los alumnos tienen que realizar conforme avanza el periodo académico.

Los detalles de la evidencia pueden ser consultados en la última sección de este documento. Asimismo, tanto usted como los participantes podrán encontrar esta información dentro del curso, siguiendo alguna de estas 2 rutas:

Mi curso > Inicio > ¿Qué voy a aprender? > Evidencia, como se muestra enseguida:

O bien: **Mi curso > Inicio > Evidencia**, como se muestra enseguida:

> **NOTA:** Es de suma importancia que **enfatice en los participantes** guardar todos los trabajos y productos que generen durante el curso (actividades, tareas, evidencias). Esto les servirá para conformar un portafolio personal de proyectos, así como para la elaboración de su proyecto integrador (último curso del certificado). Para ello, se le solicita colocar un aviso en Blackboard (sección *Announcements*), tomando como referencia el siguiente texto:

Estimado participante, recuerda guardar siempre una copia digital de todos los trabajos, actividades y evidencias que realices en tus cursos. Contar con estos documentos te será de utilidad especialmente para dos fines:

1. Conformar un portafolio personal de proyectos, que te servirá como un medio importante para enriquecer tu proyección profesional.
2. Poder elaborar el proyecto integrador de tu certificado (última materia).

Por lo tanto, asegúrate de respaldar todos tus documentos localmente en un disco duro (computadora + USB flash drive), y de preferencia también almacenarlos en la nube (servicios como Dropbox y Google Drive).

Temario

Los temas que se abordarán en este curso de certificado son los siguientes:

Módulo 1 Proceso del software	**Tema 1.** Procesos de desarrollo de software 1.1 Modelos de desarrollo 1.2 Importancia de la administración del software 1.3 Integración de modelos de madurez de capacidades (CMMI) **Tema 2.** PSP (personal Software Process) 2.1 Conceptos básicos del PSP 2.2 Elementos del PSP 2.3 Medidas del PSP **Tema 3.** TSP (Team Software Process) 3.1 Conceptos básicos del TSP 3.2 Equipos de trabajo 3.3 Administración de equipos de trabajo **Tema 4.** Medidas del software 4.1 Tamaño del software 4.2 Uso de los datos del tamaño 4.3 Puntos de función **Tema 5.** Estimación del software 5.1 Fundamentos de la estimación del software 5.2 Método de estimación del esfuerzo 5.3 Método de estimación PROBE
Módulo 2	**Tema 6.** Planificación de proyectos 6.1 Fundamentos de la planificación del software 6.2 Planificación del software 6.3 Estimación y calendarización actividades

Planificación de Proyectos de Software	**Tema 7.** Planificación de equipo de trabajo 7.1 Perfiles del personal 7.2 Planificación de la capacidad de la organización 7.3 Estrategias del equipo de trabajo **Tema 8.** Administración del riesgo 8.1 Importancia de la administración del riesgo 8.2 Proceso de gestión de riesgos **Tema 9.** Estimación del costo de desarrollo del software 9.1 Costo del esfuerzo 9.2 Costos directos e indirectos del software 9.3 Modelos de precios **Tema 10**. Six Sigma en proyectos de software 10.1 Six Sigma 10.2 DMAIC de Six Sigma en software 10.3 DFSS de Six Sigma en software
Módulo 3 Calidad en el Software	**Tema 11**. Fundamentos de la calidad del software 11.1 Conceptos de calidad 11.2 Planeación de la calidad 11.3 Métodos de la gestión de la calidad **Tema 12**. Proceso del software confiable 12.1 Tipos de procesos 12.2 Confiabilidad del software 12.3 Procesos confiables **Tema 13**. Verificación y validación del software 13.1 Verificación del software 13.2 Técnicas de verificación 13.3 Validación del software **Tema 14.** Métricas de calidad 14.1 Métricas del software 14.2 Tipos de métricas del software 14.3 Técnicas de optimización **Tema 15.** Aseguramiento de la calidad 15.1 Gestión de la calidad de software 15.2 Gestión de defectos 15.3 Cultura de calidad

Recursos especiales

Para la impartición de este curso, se requerirá de hacer uso del laboratorio de cómputo, y disponer de Microsoft Office, Microsoft Project (2010), iMindMap y Microsoft Access o VBasic o Visual Studio.

Asimismo, el libro de texto que deberán adquirir los participantes es el siguiente:

Jensen, R. (2014). *Improving Software Development Productivity: Effective Leadership and Quantitative Methods in Software Management*. EE. UU: Prentice Hall.
ISBN: 978-0-13-356267-5
e-Book: 978-0-13-356269-9

Las explicaciones de cada tema en Blackboard no sustituyen de ninguna forma la necesidad de comprar el libro de texto que ha sido designado para este curso. Es importante hacer hincapié en esto frente a los participantes.

Evaluación

La evaluación del curso se estructura de la siguiente manera:

Unidades	Instrumento Evaluador	Puntos
15	Actividades	70
1	Entrega primer avance evidencia	15
1	Evidencia final	15
	Total	**100 puntos**

Dichos productos se entregarán de acuerdo a la siguiente agenda, definida una vez que se hayan **validado fechas y valores con la información disponible en Servicios en Línea**:

Tema	Actividad	Puntaje
Tema 1	Actividad 1	5
Tema 2	Actividad 2	4

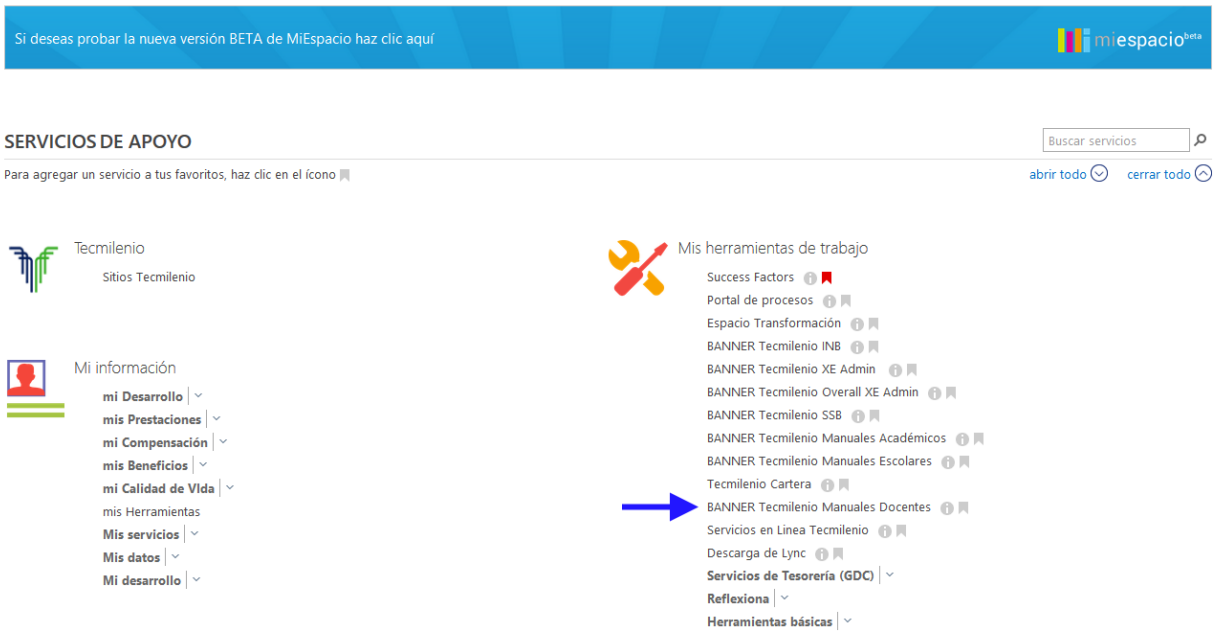
Tema 3	Actividad 3	4
Tema 4	Actividad 4	4
Tema 5	Actividad 5	4
Tema 6	Actividad 6	4
Tema 7	Actividad 7	5
	Avance 1 evidencia	15
Tema 8	Actividad 8	4
Tema 9	Actividad 9	4
Tema 10	Actividad 10	4
Tema 11	Actividad 11	5
Tema 12	Actividad 12	5
Tema 13	Actividad 13	6
Tema 14	Actividad 14	5
Tema 15	Actividad 15	7
	Evidencia final	15
	Total	**100 puntos**

IMPORTANTE:

Estimado profesor, no olvides capturar las calificaciones de tu grupo en las fechas indicadas

Puedes ver un manual para capturar calificaciones siguiendo esta ruta en Mi espacio:
Mi espacio → Servicios → De Apoyo → BANNER Tecmilenio Manuales Docentes

Puedes ver un manual para capturar inasistencias siguiendo esta ruta en Mi espacio:
Mi espacio → Servicios → De Apoyo → BANNER Tecmilenio Manuales Docentes



Notas de enseñanza por tema

Antes de impartir el curso, por favor revise de manera general los datos y conceptos proporcionados en el mismo, con el fin de detectar y, en su caso, poder actualizar y/o enriquecer previamente la información específica al tiempo en que se está impartiendo el curso.

Un aspecto de gran importancia en el desarrollo de los temas es el involucramiento del Facilitador para propiciar que la competencia del curso se cumpla, pero también ir preparando a los participantes para que vayan desarrollando propuestas de soluciones innovadoras a problemas actuales de la ingeniería de software.

Las notas de enseñanza aquí mostradas son referencia para la versión presencial y en línea, a menos que se indique lo contrario en cada tema. Puede revisarlas a continuación.

Generalidades

Para la impartición de este curso, se sugiere:

1. Revisar con tiempo la lista de entregables y la agenda en Servicios en Línea para saber en qué temas y semanas se deben realizar las actividades.
2. Revisar el manual de Blackboard para conocer las mejores formas de mantener una comunicación constante y efectiva con los estudiantes, despejar dudas y motivarlos. Puede ver un tutorial de la plataforma en esta liga: https://drive.google.com/file/d/0Bw75UcLH85hkOHVLaGo3WC1qUDA/view?usp=sharing
3. Revisar periódicamente el foro de dudas en Blackboard para resolver las preguntas e inquietudes de los alumnos acerca de las actividades y la evidencia.
4. Motivar al alumno a participar y realizar sus actividades a tiempo.
5. Proveer retroalimentación constante de las actividades que realizan los participantes.
6. Realizar un calendario y subirlo a la plataforma para que los participantes puedan visualizar de manera esquemática los temas y actividades que deberán estar revisando cada semana.
7. Recordar a los participantes que es de suma importancia que guarden tanto las actividades como la evidencia del curso en su archivo personal, pues requerirán dichos documentos para elaborar su proyecto integrador (último curso del certificado).
8. Enriquecer el curso con videos o lecturas adicionales.

Si usted imparte el **curso en modalidad online**, se recomienda también lo siguiente:

9. Realizar al menos 2 sesiones sincrónicas durante el curso con los participantes para repasar los temas revisados y resolver las diferentes dudas que puedan surgir. El Facilitador seleccionará la herramienta o plataforma que mejor le convenga: Collaborate (dentro de Blackboard), WebEx, Skype, Google Hangouts, Join.me, Zoom, etc.
   Puedes ver una **guía para organizar las sesiones sincrónicas** haciendo clic en este enlace: https://drive.google.com/file/d/0Bw75UcLH85hkdjA5bzNCNmllWW8/view?usp=sharing
10. Recordar con anuncios a los participantes acerca de las entregas de sus actividades por medio de la sección de Entrega de tareas o por correo electrónico.

Tema 1

Objetivo:
Identificar las principales metodologías de desarrollo y el modelo CMMI como parte del proceso de mejora.

Notas para la enseñanza del tema:

- Existen otras metodologías de programación como Top-down / Bottom-Up, eXtreme Process (XP), V-Model XT, Joint Application Development, Model Driven Engineering, Iterative Development Process, LEAN method (Agile), Wheel and Spoke Model, Constructionist Design Methodology. Se deja al criterio del profesor abordarlas en clase como parte de este tema.

- La última actualización del modelo CMMI fue en 2010, y sigue siendo vigente para que las empresas de desarrollo puedan autoevaluar sus procesos de desarrollo. Es un modelo que se utiliza en empresas de más de 2000 empleados, con procesos de desarrollo complejos.

- Los niveles de madurez de los procesos negativos (-1 Negligente, -2 Obstructivo, -3 Desdeñoso y -4 Socavado) fueron propuestos por Watts Humphrey, sin embargo, en el modelo actual del CMMI sólo aparecen los niveles positivos

Notas para la actividad:

SCRUM es una metodología de desarrollo de software muy popular. El objetivo de esta actividad es que el alumno pueda familiarizarse con esta metodología y pueda utilizarla en su práctica profesional.

Se omitió la descripción de esta metodología en el tema a propósito, con la idea que el alumno investigue por su propia cuenta los detalles de SCRUM.

Se sugiere elegir algunos de los videos que los alumnos encuentren para presentarlos en clase.

En el siguiente recurso podrá encontrar una guía muy completa de esta metodología escrita por sus fundadores:
Schwaner, K y Sutherland, J. (2013). *La guía de Scrum*. Recuperado de http://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-ES.pdf#zoom=100

Tema 2

Objetivo:
Utilizar el modelo del PSP como parte de una metodología para calcular la productividad de los desarrollos de software.

Notas para la enseñanza del tema:

- Las fases descritas en este tema constituyen la línea base sobre la que se fundamente el proceso PSP o lo que también se le llama PSP0.
- Existen otros niveles del PSP que servirán más adelante para realizar estimaciones del tamaño del software y en la formación de equipos de software. Por lo pronto es importante sentar las bases del proceso.
- Los formatos incluidos en este tema fueron tomados directamente del libro de apoyo del curso. Son únicamente una guía, aunque es muy conveniente que los alumnos los utilicen tal y como los propone Watts Humphrey antes de hacerles alguna modificación.

Notas para la actividad:

Si lo considera prudente puede asignar esta actividad como trabajo en equipo de 2 personas.

En internet existe poco material en video que describe el PSP como una metodología de desarrollo útil para todo programador, por lo que es una excelente oportunidad para que alumnos del Tecmilenio propongan material que puede ser visto en YouTube.

Fomente en el grupo la creatividad y exija un trabajo bien hecho. Nuestros alumnos pueden llegar a sorprendernos.

Tema 3

Objetivo:
Reconocer las características de un equipo de personas a las que se les asigna un proyecto de desarrollo de software.

Notas para la enseñanza del tema:

1. La importancia de este tema radica en hacer conciencia en el alumno de la importancia del trabajo en equipo. Se sugiere hacer un breve ejercicio sobre los problemas que los alumnos enfrentan cuando realizan algún trabajo en equipo y qué sugerencias proponen para obtener mejores resultados.

2. TSP es una metodología que, si es adecuadamente aplicada, no sólo servirá para el trabajo en equipo en proyectos de software sino para cualquier otro proyecto que involucren un gran número de personas.

Notas para la actividad:

3. Puede mencionarles que la representación será video grabada para que sus alumnos tomen en serio la actividad.
4. Es importante que si decide video-grabarlos tenga el consentimiento de ellos para tomarlo como material didáctico en otros cursos o compartirlos con el resto de la clase. Decidan en grupo si será material que puede ser público.
5. Puede ajustar el tiempo que tienen disponible por equipo para realizar la presentación. Considere unos 3 a 5 minutos entre las presentaciones de los equipos.
6. Es importante que le presenten un borrador antes de realizar la representación para evitar la improvisación o adecuar el lenguaje utilizado.

En línea

- Puede mencionarles que el grupo votará por el mejor video animado. El ganador puede ganarse algunos puntos extras en la siguiente actividad.
- Se requiere que genera un foro de discusiones específico para compartir las ligas de los videos animados.
- Se sugiere el uso de una encuesta en blackboard que sirva para votar la animación que más les gustó.
- Se sugiere el uso de la herramienta de animación POWTOON ya que es sencilla y contiene elementos de animación necesarios para esta actividad. Sin embargo, si el alumno se siente más cómodo con otra herramienta, es libre de hacerlo.

Tema 4 Medidas del software

Objetivo:
Utilizar las líneas de código y los puntos de función como técnicas para calcular el tamaño del software.

Notas para la enseñanza del tema:
Existen otros métodos para calcular el tamaño del software, entre ellos se encuentran Puntos de Casos de Uso, Análisis de Puntos de función Mark II, Puntos de características (Feature Points), Monte Carlo, Story Point, Puntos de Casos de Prueba.

Watts Humphrey considera que no existe una medición que pueda adaptarse a cualquier escenario, La mejor estrategia será reunir la mayor información, estimar y después comparar lo estimado con lo real para hacer los ajustes necesarios.

Existen algunos autores que desdeñan el uso de la técnica LOC para estimar el tamaño del software, sin embargo, es importante que el alumno la reconozca como parte de su formación. Es importante que se forme un criterio sobre las desventajas de su uso.

	Líneas de código	**Puntos de función**

Lenguaje y tecnología	LOC es dependiente del lenguaje de programación	PF es independiente del lenguaje de programación
Comunicación con los clientes	Muchos clientes no están familiarizados con LOC	La funcionalidad es relativamente más fácil de comunicarla al cliente
Herramientas	Existen muchas herramientas que pueden automatizar el conteo de LOC	No existen ninguna herramienta que pueda automatizar los PF.
Uso	LOC ofrece buenos resultados cuando el proyecto se encuentra en la etapa de Diseño	Se puede utilizar PF desde la etapa de requerimientos

Notas para la actividad:

Puede utilizar la siguiente página para mostrarles un ejemplo de una aplicación que contabiliza las líneas de código:

http://cloc.sourceforge.net/

En lugar del artículo propuesto en esta actividad puede sustituirlo por la explicación de un ejercicio completo en el que utilizaron puntos de función.
Santana, A. (2005). *Caso práctico sobre análisis de puntos de función*. Recuperado de:
Parte 1: http://sg.com.mx/content/view/216
Parte 2: http://sg.com.mx/content/view/202
Parte 3: http://sg.com.mx/content/view/305

Es posible convertir los puntos de función a líneas de código.

En la sección 4.4.4 del siguiente artículo sus autoras ofrecen una tabla de conversión según el lenguaje de programación:

Gómez, A., López, M., Migani, S. y Otazú, A. (s.f.). *Un modelo de estimación de proyectos de software*. Recuperado de
http://www.academia.edu/4853590/UN_MODELO_DE_ESTIMACION_DE_PROYECTOS_DE_SOFTWARE

Tema 5

Objetivo:
Aplicar métodos de estimación del tamaño de software a proyectos de desarrollo en los que participes.

Notas para la enseñanza del tema:
En el tema 5.3 se requerirá que el alumno se encuentre familiarizado con conceptos estadísticos de correlación, varianza y regresión lineal. Se sugiere solicitarles un breve repaso a estos conceptos antes de revisar el método PROBE.

El coeficiente de correlación permite identificar si dos variables están relacionadas entre sí.
El valor de r se encuentra entre -1.0 y +1.0. Entre más cercano esté de +1.0 se considera que existe una correlación fuerte entre ambas variables. Un valor de mayor a 0.7 se dice que existe una relación suficiente para poder ser utilizada en estimaciones.

La fórmula estadística del índice de correlación es la siguiente:

$$r(x,y) = \frac{n\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{\sqrt{\left[n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2\right]\left[n\sum_{i=1}^{n} y_i^2 - \left(\sum_{i=1}^{n} y_i\right)^2\right]}}$$

Se recomienda el uso de paquetes computacionales que calculen el coeficiente como Excel o SPSS.
Notas para la actividad:

Los cuatro métodos de PROBE se encuentran explicados en el siguiente material, que es parte de las lecturas obligatorias del curso:
SEL. (2006). *Estimación con PROBE*. Recuperado de
http://ocw.uc3m.es/ingenieria-informatica/principios-de-ingenieria-informatica/probe-ii

Tema 6

Objetivo:
Reconocer la importancia de llevar el control de un proyecto de software, basándose en la estimación y registro de avances del programador.

Notas para la enseñanza del tema:
En este tema se abordan la metodología de administración de proyectos que propone el PSP. Dejamos a su consideración incluir alguna otra metodología como otra opción de seguimiento de proyectos.

Los participantes deben estar familiarizados con aspectos relacionados con la administración de proyectos, por lo que las herramientas —como el cronograma, diagramas GANTT, diagrama de Pareto, estimación lineal y regresión— ya las deben conocer. Se sugiere que si, al mencionar este tipo de herramientas, nota que el grupo las desconoce, haga un breve repaso o ejercicio.

Notas para la actividad:

1. En esta actividad se espera que el participante practique el proceso de calendarización de tareas propuesto por Watts Humphrey.
2. Dado que no existen datos históricos sobre los que se pueda basar, se le pide al participante realizar un estimado del tiempo de cada una de las tareas basándose en su propia experiencia.
3. Se sugiere realizar una breve comparación entre los proyectos de varios participantes durante clase, con el objetivo de ejemplificar lo dispares que pueden ser planes de desarrollo para un mismo proyecto cuando la base es subjetiva.

Tema 7

Objetivo:
Calcular la capacidad de la organización para realizar un proyecto de software y reconocer estrategias del trabajo en equipo.

Notas para la enseñanza del tema:

La metodología de trabajo del TSP es abordada en diferentes temas de este curso, así la planeación se vio en el tema 6, los registros y métricas en el tema 2. Los temas de calidad serán abordados en el módulo 3 del curso.

El valor TURN utilizado en este tema es mencionado por el autor como parte importante para calcular la capacidad de la organización en el sentido en el que es más sencillo interactuar con compañeros de trabajo utilizando documentación impresa. Es posible omitir este valor si es que los programadores tienen asignado un equipo laptop que les da movilidad. En opinión de Jensen (2014) no tener la capacidad de imprimir algún documento podría afectar la productividad de la organización.

Notas para la actividad:

Para obtener el cálculo OCR, el participante deberá utilizar la ecuación propuesta por Jensen y ubicar los valores necesarios en las tablas incluidas en el tema.

Las acciones para motivar al personal pueden ser investigadas en Internet o biblioteca digital. La cantidad de acciones sugeridas debe ser entre 5 y 10.

El sistema del caso se considera que tiene un nivel alto de complejidad, dado que incluye varios módulos.

Tema 8

Objetivo:
Identificar los posibles riesgos que pueden afectar a un proyecto de desarrollo de software.

Notas para la enseñanza del tema:
Según el manual CISA existen 3 tipos de análisis de riesgos: el cualitativo, donde la clasificación de los riesgos es subjetivo. El análisis semicualitativo, en las que las clasificaciones descriptivas están asociadas a una escala numérica, tratando de reducir el subjetivismo; y el análisis cuantitativo que se basa en valores numéricos para describir la probabilidad y los impactos del riesgo, haciendo uso de datos históricos, registros de la industria, teorías estadísticas, pruebas y experimentos.

En este curso sólo se utiliza el análisis de riesgos cualitativo por considerarse el más sencillo. Dejamos a consideración del profesor abordar algún otro tipo de análisis.

Notas para la actividad:

1. En las técnicas de identificación de riesgos, el participante debe expresar el proceso que establecen para reconocer los riesgos en un proyecto. Recordarles que estas técnicas también se utilizan para recopilar información.

2. La escala para determinar el impacto y la probabilidad podría cambiar siempre y cuando el participante incluya el criterio utilizado.

3. En la matriz RASCI es importante que sólo incluyan un responsable por riesgo.

Tema 9

Objetivo:
Practicar el cálculo de precios en proyectos de software.

Notas para la enseñanza del tema:
El ejercicio de análisis de costos de este tema está basado fundamentalmente en aquellos que se encuentran directamente relacionados con el esfuerzo del software, sin embargo, en muchos casos, los contratos con empresas incluyen otro tipo de costos como servicios de mantenimiento, capacitación, soporte técnico y adquisición de hardware. Dejamos a consideración del profesor mencionar la oportunidad de considerar este tipo de costos al momento de fijar un precio por el esfuerzo del software.

Notas para la actividad:
En el cálculo de los costos directos es importante que no se consideren los gastos del personal a nivel gerente, director y asistente, estos son considerados como costos indirectos.

Para la investigación sobre el concepto Freemium se recomienda el siguiente artículo:
Kumar, V. (2014). *Making Freemium Work*. Recuperado de:
https://hbr.org/2014/05/making-freemium-work

1.

Tema 10

Objetivo:
Revisar dos metodologías de Six Sigma para mejorar procesos de desarrollo de software.

Notas para la enseñanza del tema:
La siguiente tabla puede ser utilizada para identificar las herramientas estadísticas y de diagramación que pueden ayudar en cada una de las fases de la metodología DMAIC.

DMAIC Phase Steps	Tools Used
• **D - Define Phase:** Define the project goals and customer (internal and external) deliverables.	
• Define Customers and Requirements (CTQs) • Develop Problem Statement, Goals, and Benefits • Identify Champion, Process Owner, and Team • Define Resources • Evaluate Key Organizational Support • Develop Project Plan and Milestones • Develop High-Level Process Map	• Project Charter • Process Flowchart • SIPOC Diagram • Stakeholder Analysis • DMAIC Work Breakdown Structure • CTQ Definitions • Voice of the Customer Gathering
• **M - Measure Phase:** Measure the process to determine current performance; quantify the problem.	
• Define Defect, Opportunity, Unit, and Metrics • Detailed Process Map of Appropriate Areas • Develop Data Collection Plan • Validate the Measurement System • Collect the Data • Begin Developing Y = f(x) Relationship • Determine Process Capability and Sigma Baseline	• Process Flowchart • Data Collection Plan/Example • Benchmarking • Measurement System Analysis/Gage R&R • Voice of the Customer Gathering • Process Sigma Calculation
• **A - Analyze Phase:** Analyze and determine the root cause(s) of the defects.	
• Define Performance Objectives • Identify Value/Non-Value-Added Process Steps • Identify Sources of Variation • Determine Root Cause(s) • Determine Vital Few x's, Y = f(x) Relationship	• Histogram • Pareto Chart • Time Series/Run Chart • Scatter Plot • Regression Analysis • Cause-and-Effect/Fishbone Diagram • 5 Whys • Process Map Review and Analysis • Statistical Analysis • Hypothesis Testing (Continuous and Discrete) • Non-Normal Data Analysis
• **I - Improve Phase:** Improve the process by eliminating defects.	
• Perform Design of Experiments • Develop Potential Solutions • Define Operating Tolerances of Potential System • Assess Failure Modes of Potential Solutions • Validate Potential Improvement by Pilot Studies • Correct/Re-Evaluate Potential Solution	• Brainstorming • Mistake Proofing • Design of Experiments • Pugh Matrix • House of Quality • Failure Modes and Effects Analysis (FMEA) • Simulation Software
• **C - Control Phase:** Control future process performance.	
• Define and Validate Monitoring and Control System • Develop Standards and Procedures • Implement Statistical Process Control • Determine Process Capability • Develop Transfer Plan, Handoff to Process Owner • Verify Benefits, Cost Savings/Avoidance, Profit Growth • Close Project, Finalize Documentation • Communicate to Business, Celebrate	• Process Sigma Calculation • Control Charts (Variable and Attribute) • Cost-Savings Calculations • Control Plan

El-Haik, B. y Shaout, A. (2010). Software Design for Six Sigma: A Roadmap for Excellence. USA: John Wiley & Sons.

En la metodología DFSS no existe un criterio unificado sobre sus fases. Las fases descritas en este tema están basadas en el libro de texto de El-Haik, B. y Shaout, A., titulado Software Design for Six

Sigma: A Roadmap for Excellence. Es posible que los participantes encuentren otras fases en artículos relacionados con Six Sigma.

En la etapa 4 del desarrollo de un sistema, es recomendable utilizar la **matriz Pugh** para elegir la mejor opción de diseño. El orden al colocar los conceptos y los criterios puede ser intercambiado, es decir, colocar los criterios en las columnas y los conceptos o soluciones en los renglones.

Notas para la actividad:

El participante puede basar sus respuestas haciendo uso de otras fuentes de información, siempre y cuando las incluya dentro de las fuentes bibliográficas.

Dejamos a consideración del profesor sustituir esta actividad por una investigación de un caso de éxito en el que hayan implementado Six Sigma en un proyecto de software. El participante deberá explicar los antecedentes, proceso de implementación, métricas utilizadas y beneficios logrados.

Tema 11

Objetivo:
Reconocer la importancia que tiene la calidad del software y cómo gestionarla.

Notas para la enseñanza del tema:
La metodología TSP incluye los siguientes métodos como parte de la gestión de calidad:

- Capture-recapture defect estimates
- Estimating yield
- Test data analysis

El perfil de calidad propuesto por TSP es una guía de calidad, no quiere decir que tener un buen perfil se tenga la seguridad de evitar defectos, por lo que el líder del equipo de desarrollo debe complementar esta información con algunos otros indicadores de calidad.

En caso de que el participante tenga problemas para interpretar u obtener la gráfica del perfil de calidad, se pueden incluir los siguientes cálculos:

1. If detailed design time (DLDT) ≥ coding time (CT), P1 = 1.0.
   If DLDT < CT, P1 = DLDT/CT.

2. If detailed design review time (DLDRT) ≥ 0.5 * DLDT, P2 = 1.0.
   If DLDRT < 0.5 * DLDT, P2 = DLDRT/ (0.5 * DLDT).

3. If code review time (CRT) ≥ 0.5 * CT, P3 = 1.0.
   If CRT < 0.5 * CT, P3 = CRT/ (0.5 * CT).

4. If compile defects (CD) ≤ 10 defects/KLOC, P4 = 1.0.
   If CD > 10 defects/KLOC, P4 = 20/ (10 + CD).

5. If unit test defects (UTD) ≤ 5 defects/KLOC, P5 = 1.0.
   If UTD > 5 defects/KLOC, P5 = 10/ (5 + UTD).

Notas para la actividad:

El software diseñado para sistemas críticos pone como prioridad la seguridad, confiabilidad y protección, incluso por encima de otras características.

El participante puede tomar cualquiera de los estándares de calidad descritos en el tema. Dejamos a la consideración del profesor que cada participante describa un estándar diferente y lo publique en un blog de Blackboard.

No sería posible determinar que un principio de calidad es más importante que otro. En este caso lo evaluable será que la justificación que ofrezca tenga un sustento válido.

1.

Tema 12

Objetivo:
Reconocer las acciones que puedes llevar a cabo para generar un producto de software confiable.

Notas para la enseñanza del tema:
En este tema se revisan los procesos que dan confiabilidad al software.
Se recomienda al profesor considerar el tema 13.4 **Programación confiable** si el grupo de participantes tiene un perfil de ingeniería de software avanzado. El tema lo puede consultar en el libro de Sommerville, I. (2011). *Ingeniería de Software* (9ª ed.). México: Pearson Educación. En este tema, Sommerville hace algunas recomendaciones sobre la confiabilidad a nivel programación, sin llegar a utilizar un lenguaje de codificación específica.

Notas para la actividad:
Es importante que el participante respete los derechos de autor en las imágenes que utilice en su presentación, por lo que es recomendable que utilice imágenes libres de derechos de autor.

Se recomienda motivar a los estudiantes a no limitarse a leer durante su presentación. Cada filmina debe contener no más de siete renglones de información. Tamaño de letra 20.

Para realizar esta presentación puede hacer uso de las herramientas incluidas en PowerPoint para grabar la narración, o bien hacer uso de Office Mix.

Dejamos a consideración del profesor tomar las mejores presentaciones para mostrarlas durante clases o bien publicarlas en el foro de discusión Feed Up.

Tema 13

Objetivo:
Aplicar técnicas de verificación y validación del software como parte de las actividades relacionadas con el aseguramiento de la calidad del software.

Notas para la enseñanza del tema:
En el libro de Watts Humphrey existe otro método de verificación de código llamado Verificación analítica. Este tipo de verificación se basa en un análisis matemático para demostrar la validez de la lógica de un programa. Dada la complejidad del análisis, no se incluyó como parte de este tema.

Humphrey (2005) describe otros métodos automáticos de validación como ASTREE, SLAM, ESP, Vault, SPARK y Splint.

Notas para la actividad:

El participante debe analizar todos los posibles casos que se pueden presentar para verificar el diseño del módulo de autenticación.

El participante puede mencionar que una mejora sería que el mismo usuario generara su propia contraseña la primera vez que el usuario se autentique, permitiéndole configurar una contraseña que le sea más fácil de recordar en lugar de imponerle una.

Otra sugerencia sería evitar enviarle la contraseña que tiene configurada a través del correo sino una liga en la que pueda cambiarla si la olvidó. Esta liga estaría disponible por 24 horas. Si el usuario no reconfigura su contraseña la liga pierde vigencia.

También puede sugerir hacer uso de sus credenciales de Google o Facebook para que no tenga que utilizar nuevas.

1.

Tema 14

Objetivo:
Determinar métricas que sirvan para identificar áreas de oportunidad en el proceso de desarrollo de software, que impacten la calidad del producto.

Notas para la enseñanza del tema:
Es común que los conceptos de medida, medición y métrica sean utilizados frecuentemente como sinónimos, sin embargo existen diferencias sutiles. En este tema es importante diferenciar cada uno de estos conceptos, haciendo uso del glosario de términos.

Es muy posible que el participante encuentre una infinidad de métricas del software. Lo importante para su formación será reconocer su importancia, definir unas cuantas métricas y darles seguimiento.

Notas para la actividad:
En caso de que el participante no tenga una licencia de imindmap, puede descargar la versión de prueba desde imindmap.com o bien realizarlo a mano, tomarle una foto y colocarlo en su trabajo.

Dejamos a consideración del profesor cambiar el formato de mapa mental por un mapa conceptual en caso que los participantes aún no tengan la certificación de mapas mentales

Tema 15

Objetivo:
Utilizar elementos que permiten mejorar la calidad de los productos de software.

Notas para la enseñanza del tema:
La gestión de defectos que explica este tema es diferente a la que se establece en la etapa de pruebas. En este tema su enfoque es hacia el descubrimiento de defectos que no fueron encontrados durante las actividades de pruebas (pruebas unitarias y pruebas al código). La cultura de calidad es un tema que cualquier empresa requiere, incluso de aquellas dedicadas a otros giros diferentes al software. Es un tema de motivación del personal en el que se requiere hacer conciencia de los beneficios de la calidad, no sólo para el producto, sino para las personas. Existe un mejor ambiente de trabajo y se comparten los éxitos entre todos, lo que hace un círculo virtuoso.

La tabla de estimación de defectos vuelve a aparecer en este tema, en esta ocasión debido a que se trata de establecer un estándar para la calidad, mientras que en el tema 11 es sobre la planeación de la calidad.

Notas para la actividad:
En este tema se espera que el participante cree un prototipo funcional en el que pueda simular el formulario para ingresar la información de un defecto. Puede hacer uso de Access, VBA (Excel), Visual Studio o cualquier otra aplicación que permita diseñar formas.

Para generar un catálogo de fallas, le sugerimos dar algunos ejemplos del siguiente listado:

Defect category	Defect origin	Defect severity
Accessed or stored data incorrectly	coding	major
Add new capability	requirements	major
Ambiguous statement	coding	minor
Applicable standards not met	design	critical
Change in program requirements	requirements	major
Checking wrong variable	coding	major
Computational problem	coding	major
Conflicting item	design	major
Confusing item	design	minor
Data handling problem	coding	critical
Data problem	coding	minor
Data referenced out of bounds	coding	major
Dimensioned data incorrectly	coding	minor
Documentation problem	coding	minor
Document quality problem	documentation	minor
Duplicate logic	coding	minor
Embedded data in tables incorrect or missing	coding	critical
Enhancement	design	major
Equation insufficient or incorrect	coding	critical
External data incorrect or missing	coding	major
Extreme conditions neglected	coding	critical
Failure caused by a previous fix	coding	critical
Flag or index set incorrectly	coding	major
Forgotten cases or steps	coding	critical
Illogical item	design	major
Implement editorial changes	documentation	minor
Improve code efficiency	coding	major
Improve comments	documentation	minor
Improve usability	design	major
Incomplete item	coding	major
Incomplete statement	coding	critical
Inconsistencies	coding	critical
Inconsistent subroutine arguments	coding	major
Incorrect item	coding	critical
Incorrectly located subroutine called	coding	major
Initialized data incorrectly	coding	major
Input data incorrect or missing	coding	critical
Input or output timing incorrect	coding	critical
Interface or timing problem	coding	critical
Interrupts handled incorrectly	coding	critical
Iterating loop incorrectly	coding	major
Logic problem	coding	critical

1.

Evidencia

El participante deberá elaborar una evidencia (producto final) por medio de la cual demuestre el dominio de la competencia del curso, como elemento indispensable para conseguir la acreditación del mismo. Es decir, lo plasmado en la evidencia es aquello que buscamos que los estudiantes sean capaces de hacer bien.

Es importante insistir en que los participantes se tomen en serio la elaboración de las evidencias de sus certificados, pues con ellas pueden armar un portafolio interesante de proyectos que les servirá mucho al momento de buscar ingresar al mercado laboral.

Las instrucciones para la realización de la evidencia son las siguientes:

Competencia del curso: Utiliza métodos de estimación en proyectos de software bajo el enfoque de procesos del desarrollo de software.

Nombre de la evidencia: Estimación de software para Transcargo.

Descripción de la actividad: En esta evidencia diseñarás el software para una agencia de transporte estimando su tamaño y basándote en el análisis del sistema, además detallarás los elementos del proyecto de desarrollo e implementación necesarios. La evidencia será entregada en 2 etapas.

Requerimientos: MS-Word (para elaborar el trabajo escrito)

MS-Excel (para realizar cálculos sencillos)

MS-Project (para planeación del proyecto)

Entregables: A continuación se describen los entregables para cada etapa.

Avance 1 evidencia:
Análisis del sistema y diseño, estimación del software y la planeación.

Evidencia final:
Análisis de riesgos y aseguramiento de la calidad,

TransCargo es una agencia de transporte que ofrece sus servicios de logística para empresarios y agricultores que desean exportar sus productos a diferentes países de Latinoamérica.

Una de sus estrategias de negocio es apoyarse en las tecnologías de información para mantener un mejor control de sus operaciones.
Para ello han contratado a una empresa de localización GPS que instalará el equipo necesario que ubicará a cada unidad de autotransporte en tiempo real. El siguiente paso será desarrollar el software con las siguientes funciones:

- **Web de rastreo de producto:** se espera que los clientes de TransCargo puedan rastrear la ubicación de su producto a través de una página web.
- **Inventario de autotransportes:** le permitirá a mantener el control de las unidades de autotransporte que utiliza TransCargo para ofrecer sus servicios. El inventario debe controlar la cantidad de unidades, capacidad, tipo, modelo, marca, último mantenimiento, condiciones generales de la unidad y ubicación actual.
- **Control de Personal:** permite mantener el control de choferes y cargadores, mecánicos. Administra la información personal, sueldo, puesto, etc.
- **Otros datos:** ofrece información a los clientes sobre teléfonos y correo electrónico de contacto, ciudades de cobertura, información de la compañía, horarios de atención.

El desarrollo será realizado por personal interno que pertenece al área de sistemas de la empresa.

Cantidad	Puesto	Costo	Funciones
1	Gerente de sistemas	$125 por hora	Responsable de los proyectos de sistemas.
2	Analista de Sistemas	$70 por hora	Generan el documento de especificaciones de software.
1	Gerente de Aseguramiento de la Calidad	100 por hora	Responsable de las actividades de aseguramiento de la calidad del software: documentación, métricas de calidad.
1	Arquitecto de Software	$80 por hora	Responsable de los diseños de la arquitectura del software, interfaz gráfica, base de datos.
1	Coordinador de *Testing*	$90 por hora	Responsable de las actividades de verificación y validación del software.
5	Ingenieros de *Testing*	$50 por hora	Realizan actividades de verificación del código.
3	Ingenieros de software	$60 por hora	Realizan el código del sistema y pruebas unitarias. Implementan el sistema y dan soporte de la etapa de estabilización

Con base en proyectos anteriores, el gerente de sistemas ha estimado el siguiente tiempo según las etapas del ciclo de vida del desarrollo de un sistema.

	Estimación	
Etapa	**Mejor**	**Peor**
Análisis:	2 semanas	4 semanas
Diseño:	2 semanas	3 semanas
Construcción	4 semanas	8 semanas
Pruebas:	2 semanas	6 semanas
Pruebas de aceptación:	5 días	10 días
Implementación:	3 días	7 días
Soporte:	2 meses	4 meses

El gerente de sistemas tiene el siguiente registro de los factores de ajuste y el conteo a considerar para calcular los puntos de función del desarrollo:

Factor de ajuste	Valores
1	Esencial
2	Significativo
3	Moderado
4	Moderado
5	Incidental
6	Incidental
7	Moderado
8	Incidental
9	Moderado
10	Medio
11	No presente
12	Significativo
13	No presente
14	Incidental

Variable	Conteo
Entradas	35
Salidas	40
Consultas	25
Archivos	42
Interfaces	20
COCOMO II	
PERS	3
RCPX	4
RUSE	6
PDIF	5
PREX	3
SCED	1
FCIL	6

Consideraciones:

- Trabajan 8 horas diarias de lunes a viernes, sin contar días festivos considerados en la ley federal del trabajo.
- El costo está expresado en moneda nacional.
- El lenguaje que utilizarán será Java, en una arquitectura MVC "Modelo Vista Controlador".
- El valor de B utilizado para calcular el esfuerzo es 1.0.
- El monto de otros gastos diferentes a mano de obra se estima en $70,000 M.N.

Con base en estos datos realiza lo siguiente para entregar en el avance de tu evidencia.

1. Describe los **requerimientos funcionales y no funcionales** del sistema utilizando el siguiente formato. Puedes establecer los supuestos que creas necesarios para detallar los requerimientos.

No. de requerimiento	
Descripción:	
Razón fundamental:	
Criterio:	
Restricciones	
Alternativa:	

2. Diseña las pantallas como un prototipo del sistema que requiere la empresa. Considera que existirán interfaces públicas e interfaces de uso exclusivo interno.

3. Utilizando puntos de función como medidas del software calcula:

a. El esfuerzo requerido.
b. Defectos potenciales
c. Entrega de defectos
d. Esfuerzo COCOMO II
e. Costo del esfuerzo

4. El equipo de desarrollo de TransCargo desconoce el modelo del PSP y TSP. Ayuda al gerente a explicar las tres fases del modelo PSP para que el equipo lo utilice en este proyecto y las ventajas de usar TSP como parte la metodología del trabajo en equipo.

Como parte de la planeación del proyecto, describe cuál es el SOW (Statement of work) que debería documentar el área de sistemas.

5. Establece el calendario del proyecto tomando en consideración las recomendaciones de Watts Humphrey.

6. Describe cuál debería ser la estrategia de trabajo en equipo: Enfoque Big Bang o cascada o Enfoque por etapas o versiones incrementales. Justifica tu respuesta.

Evidencia final

7. Establece al menos 5 riesgos en los cuales debe enfocarse el equipo de desarrollo en el proyecto, utilizando el siguiente formato y genera la matriz RASCI considerando los puestos del equipo.

Nombre del riesgo y descripción	Impacto	Probabilidad	Asignación	Fecha de revisión

8. Elige el principal riesgo (mayor impacto y probabilidad) y describe qué plan de mitigación pueden realizar.

9. Explica la importancia que tiene trabajar con calidad en este proyecto.

10. Describe las actividades que deberá realizar el gerente de aseguramiento de la calidad para este proyecto para generar un producto de software confiable.

11. Elige una de las técnicas de verificación que le recomendarías al gerente de sistemas, explicando la técnica y agrega una justificación de tu elección.

12. ¿Cuál sería el plan de validación de podría seguir el equipo de desarrollo?

13. Elige 10 métricas de calidad del software del proyecto. Describe cómo las utilizarías.

14. Diseña un plan para crear una cultura de calidad donde detalles las actividades y los responsables de llevarlas a cabo.

La rúbrica con la que usted deberá evaluar la evidencia final es la siguiente:

Criterios	Descriptores						
	Avance 1						
	Excelente	**Sobresaliente**	**Aceptable**	**Suficiente**	**Insuficiente**	**No cumple con criterios**	**Puntos totales**
1. Análisis del sistema y diseño	**Equivalencia: 25 puntos** 1. Describe los requerimientos funcionales y no funcionales del sistema utilizando el formato solicitado de manera completa y clara. 2. Diseña al menos 3 interfaces gráficas del sistema a modo de prototipo, considerando los datos de entrada necesarios y controles de acceso	**Equivalencia: 20 puntos** 1. Describe los requerimientos funcionales y no funcionales del sistema utilizando el formato solicitado de manera parcial o poco clara. 2. Diseña al menos 3 interfaces gráficas del sistema a modo de prototipo. Incluye algunos datos de entrada necesarios y controles de acceso.	**Equivalencia: 15 puntos** 1. Describe los requerimientos funcionales utilizando el formato solicitado. No incluye requerimientos no funcionales del sistema. 2. Diseña al menos 3 interfaces gráficas del sistema a modo de prototipo, considerando los datos de entrada necesarios. No considera controles de acceso	**Equivalencia: 10 puntos** 1. Describe los requerimientos funcionales y no funcionales del sistema sin hacer uso del formato solicitado. 2. Diseña 2 interfaces gráficas del sistema a modo de prototipo, sin considerar datos de entrada indispensables para el sistema y el control de acceso.	**Equivalencia: 5 puntos** 1. Existen claras omisiones al describir los requerimientos funcionales y no funcionales que se consideran indispensables para el sistema. 2. Diseña una interfaz gráfica del sistema a modo de prototipo.	**Equivalencia: 0 puntos** 1. No cumple con el criterio	**25**
2. Estimación del software	**Equivalencia: 45 puntos** 1. Incluye la estimación de los puntos de función, esfuerzo requerido, defectos potenciales, entrega de defectos, esfuerzo COCOMO II y costo del esfuerzo.	**Equivalencia: 35 puntos** 1. Incluye la estimación de los puntos de función y 4 de las medidas de software.	**Equivalencia: 25 puntos** 1. Incluye la estimación de los puntos de función y 3 de las medidas de software.	**Equivalencia: 15 puntos** 1. Incluye la estimación de los puntos de función y 2 de las medidas de software.	**Equivalencia: 5 puntos** 1. Incluye la estimación deficiente sobre los puntos de función y las medidas de software.	**Equivalencia: 0 puntos** 1. No cumple con el criterio.	**45**
3. Planeación	**Equivalencia: 30 puntos** 1. Explica las metodologías PSP y TSP de manera completa. 2. Establece de forma clara el SOW del proyecto. 3. Incluye un calendario del proyecto considerando todas las actividades	**Equivalencia: 25 puntos** 1. Explica las metodologías PSP y TSP de manera parcialmente clara. 2. Establece de forma parcialmente claro el SOW del proyecto. 3. Incluye un calendario del proyecto considerando algunas las actividades	**Equivalencia: 20 puntos** 1. Explica una de las 2 metodologías (PSP y TSP) de manera clara. 2. Establece a grandes rasgos el SOW del proyecto. 3. Incluye un calendario completo del proyecto sin seguir las	**Equivalencia: 15 puntos** 1. Explica una de las 2 metodologías (PSP y TSP) de manera clara. 2. Incluye un SOW confuso o ambiguo en relación a los objetivos del proyecto. 3. Incluye un calendario del proyecto parcialmente completo sin seguir	**Equivalencia: 10 puntos** 1. No incluye una explicación de las metodologías PSP y TSP. 2. No incluye el SOW del proyecto. 3. Incluye un calendario sin considerar actividades importantes del proyecto.	**Equivalencia: 0 puntos** 1. No cumple con el criterio.	**30**

	importantes siguiendo las recomendaciones de Humphrey.	importantes siguiendo las recomendaciones de Humphrey.	recomendaciones de Humphrey.	las recomendaciones de Humphrey.			

Criterios	Descriptores						
	Evidencia final						
	Excelente	**Sobresaliente**	**Aceptable**	**Suficiente**	**Insuficiente**	**No cumple con criterios**	**Puntos totales**
4. Análisis de riesgos	**Equivalencia: 45 puntos** 1. Describe 5 riesgos importantes para proyecto haciendo uso del formato solicitado. 2. Justifica la selección de un riesgo importante y establece un plan de mitigación adecuado.	**Equivalencia: 35 puntos** 1. Describe al menos 4 riesgos importantes para proyecto haciendo uso del formato solicitado. 2. Selecciona un riesgo importante y establece un plan de mitigación adecuado.	**Equivalencia: 25 puntos** 1. Describe al menos 3 riesgos importantes para proyecto haciendo uso del formato solicitado. 2. Establece un plan de mitigación para un riesgo poco importante para el proyecto.	**Equivalencia: 15 puntos** 1. Describe los riesgos importantes del proyecto sin utilizar el formato solicitado. 2. Establece un plan de mitigación ambiguo para un riesgo del proyecto.	**Equivalencia: 5 puntos** 1. Describe los riesgos que no son importantes para el proyecto. 2. Selecciona un riesgo para el proyecto sin incluir un plan de mitigación.	**Equivalencia: 0 puntos** 1. No cumple con el criterio.	**45**
5. Aseguramiento de la calidad	**Equivalencia: 55 puntos** 1. Establece y describe con claridad la importancia y las actividades del aseguramiento de la calidad. 2. Describe el proceso de verificación y validación del software de forma completa. 3. Establece las métricas de calidad del software y el plan de la cultura de calidad de forma adecuada.	**Equivalencia: 40 puntos** 1. Establece y describe la importancia y las actividades del aseguramiento de la calidad. 2. Describe el proceso de verificación y validación del software de forma parcialmente completa. 3. Establece las métricas de calidad del software y el plan de la cultura de calidad de forma parcialmente adecuada.	**Equivalencia: 30 puntos** 1. Establece a grandes rasgos la importancia y las actividades del aseguramiento de la calidad. 2. Describe el proceso de verificación y validación del software de forma incompleta. 3. Describe algunas métricas de calidad sin detallar el plan de cultura de calidad.	**Equivalencia: 20 puntos** 1. Establece la importancia del aseguramiento de la calidad sin llegar a detallar sus actividades. 2. Describe el proceso de verificación de forma adecuada sin llegar a detallar el plan de validación. 3. Describe métricas que no están relacionadas con la calidad del software. 4. Establece un plan de cultura de calidad incompleto.	**Equivalencia: 10 puntos** 1. Lista algunas actividades del aseguramiento de la calidad sin describirlas. 2. Detalla el plan de validación sin llegar a describir técnicas de verificación. 3. Detalla el plan de cultura de calidad sin describir las métricas de calidad a utilizar.	**Equivalencia: 0 puntos** 1. No cumple con el criterio.	**55**