
Universidad
Tecmilenio®

Universidad Tecmilenio

Ingeniería de Software

Procesos de desarrollo de software

Semana 9

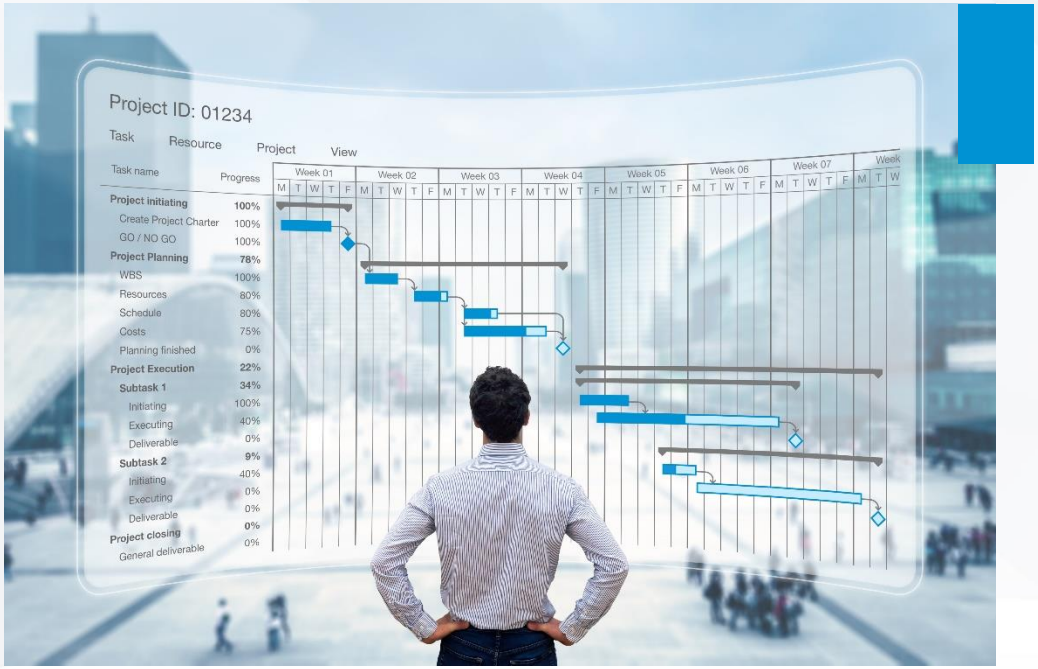
Bienestar-mindfulness

Te invito a realizar la siguiente actividad de bienestar-mindfulness antes de comenzar a revisar el tema:

https://youtu.be/1iPwm62dbxU

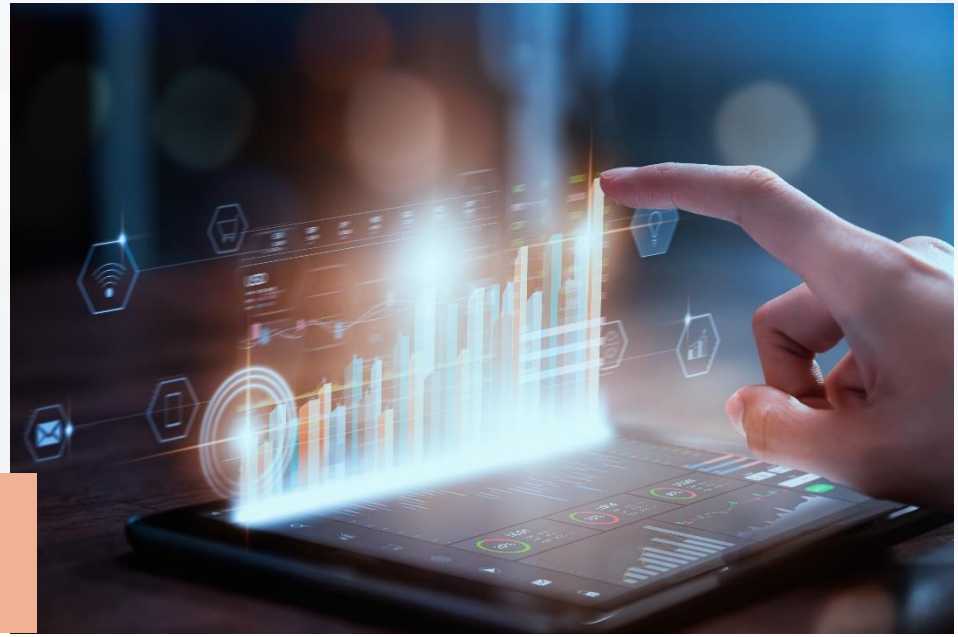
Introducción

En los proyectos de desarrollo de software, el proceso de mejora continua obliga a las organizaciones a buscar balancear tres elementos que son críticos en cualquier proyecto: las personas (sus habilidades, entrenamiento, y motivación), las herramientas y los procesos o métodos que utilizan para llegar a generar resultados.



Explicación

CMMI (Capacity Maturity Model Integration), permite reconocer las características que deben tener los procesos y provee de algunas líneas de acción que pueden seguir las empresas para crear y perfeccionar sus propios procesos de desarrollo de productos, lo que conlleva a estimaciones realistas de sus proyectos, minimizar sus costos, operativos y mejorar la calidad.



Explicación

Los niveles de proceso de madurez del software son:

Optimizado	Cuantitativamente gestionado	Definido
Gestionado	Inicial	Negligente
Obstructivo	Desdeñoso	Socavado

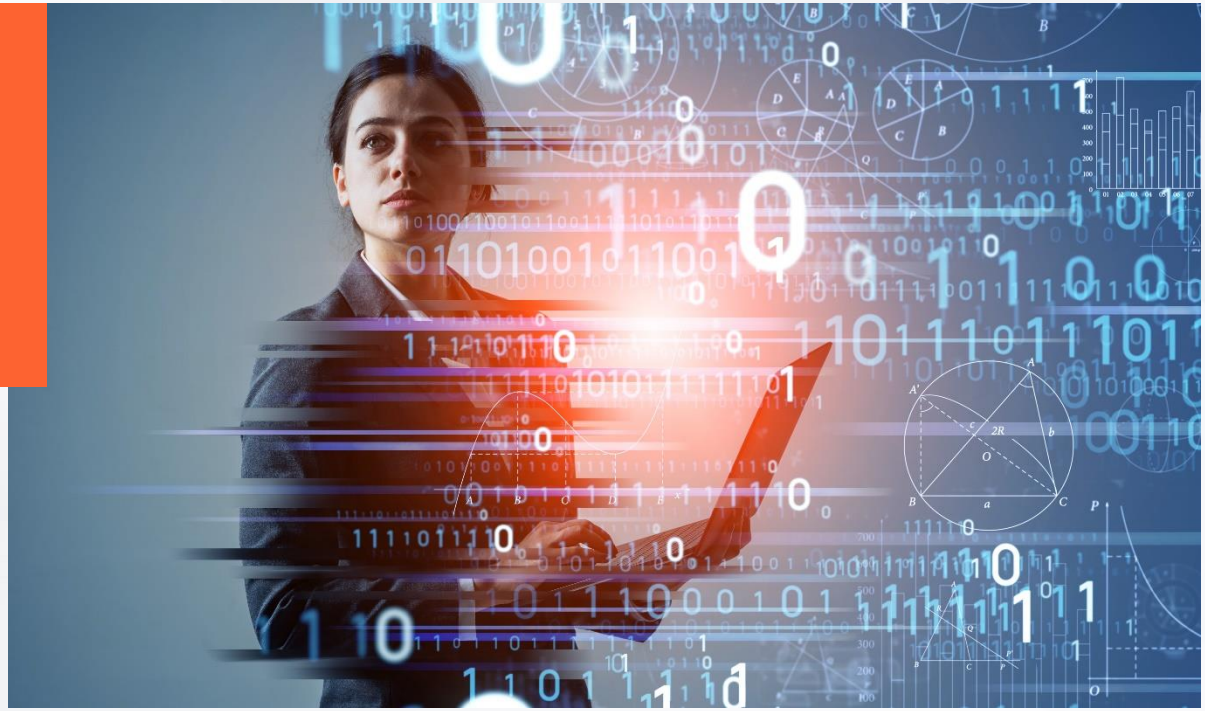
Explicación

Los principales modelos de desarrollo de software son:

Cascada	Sashimi	V model
Espiral	Prototipo	Caso
RAD	Desarrollo ágil	Proceso unificado

Ejercicio

Enlista cinco ventajas que ofrece el modelo de madurez de capacidades CMMI.



Cierre

El modelo de madurez de procesos (CMM) puede ser una herramienta muy útil para conocer el estado actual de los procesos de desarrollo de software que sigue una empresa, y de allí realizar un plan para escalar en los niveles de madurez que ayudarán a la productividad de cada desarrollo, lo que lleva a la eficiencia y a altos estándares de calidad.

Bibliográfia

- El-Haik, B. y Shaout, A. (2010). Software Design for Six Sigma: A Roadmap for Excellence. EE. UU: John Wiley & Sons.
- IDC. (2015). El mercado de software creció en 2014 pese a las condiciones macroeconómicas: IDC. Recuperado de http://ar.idclatin.com/releases/news.aspx?id=1914
- Jensen, R. (2014). Improving Software Development Productivity: Effective Leadership and Quantitative Methods in Software Management. EE. UU: Prentice Hall.

Universidad Tecmilenio

Ingeniería de Software

PSP
(Personal Software Process)

Semana 9

Introducción

PSP (Personal Software Process) es un modelo del proceso de desarrollo de software desarrollado por Watts Humphrey en 1997. Watts ideó este modelo con el propósito de que pudiera ser utilizado por un ingeniero de software en proyectos de desarrollos pequeños o en componentes que pudieran ser construidos por una sola persona.

Explicación

El proceso de PSP propone tres fases:

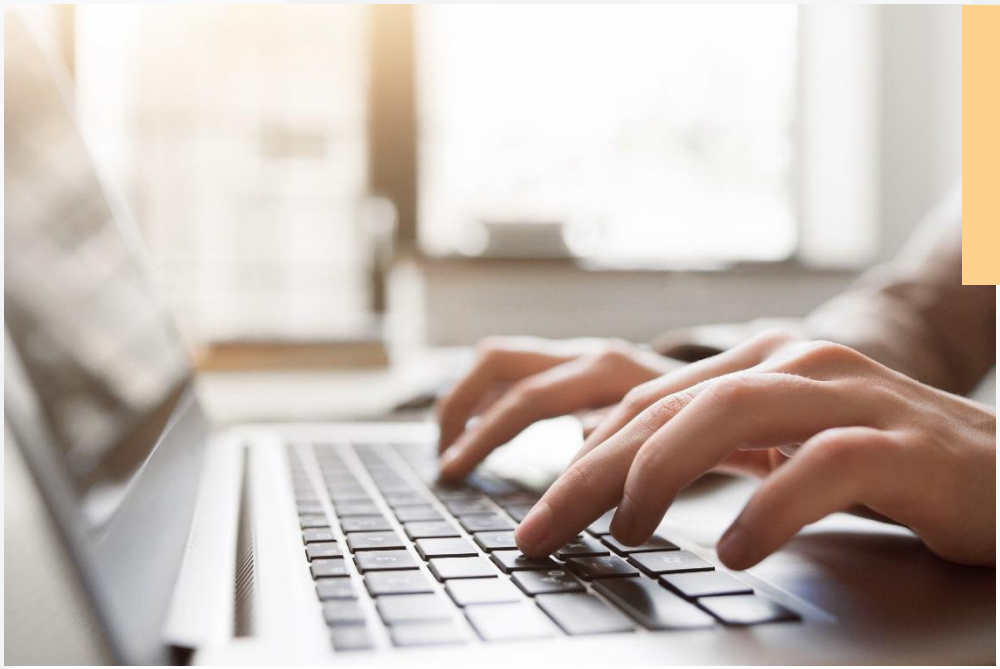
- Fase de planeación
- Fase de desarrollo
- Fase postmortem

Explicación

El PSP sugiere dos medidas del proceso de software: el tiempo transcurrido por fase y los defectos encontrados por fase.

Explicación

El proceso PSP exige una disciplina de documentación a los programadores de software que les permite analizar su productividad en un proyecto particular y es la base para implementar un proceso de mejora continua personal.



Ejercicio

Enlista cinco beneficios de medir el proceso de desarrollo de software.

Cierre

Mantener un registro del trabajo del programador será una actividad muy útil sobre todo como base para realizar estimaciones del tamaño del software y el esfuerzo que implica. Estas estimaciones serán una herramienta para planear futuros proyectos.

Bibliografía

- Humphrey, W. (2005). PSP(SM): A Self-Improvement Process for Software Engineers. EEUU: Pearson Education.

Universidad Tecmilenio

Ingeniería de Software

TSP
(Team Software Process)

Semana 9

Introducción

TSP (Team Software Process) es un proceso diseñado para guiar a equipos de personas en la planeación, diseño y desarrollo de sistemas de software de calidad.

Explicación

Los conceptos básicos del TSP son los siguientes:

Equipo de trabajo	Roles y responsabilidades	Equipos de desarrollo
Tamaño de los equipos	Cooperación e interdependencia	Líder del equipo

Explicación

Los equipos autodirigidos tienen las siguientes propiedades:

- Un sentido de propiedad y pertenencia.
- Comprometidos con el objetivo del equipo.
- Hacen suyo el proceso y el plan.
- Habilidades y disciplina.
- Comparten el compromiso.
- Están dedicados a alcanzar la excelencia.

Explicación

Los conceptos básicos del TSP son los siguientes:

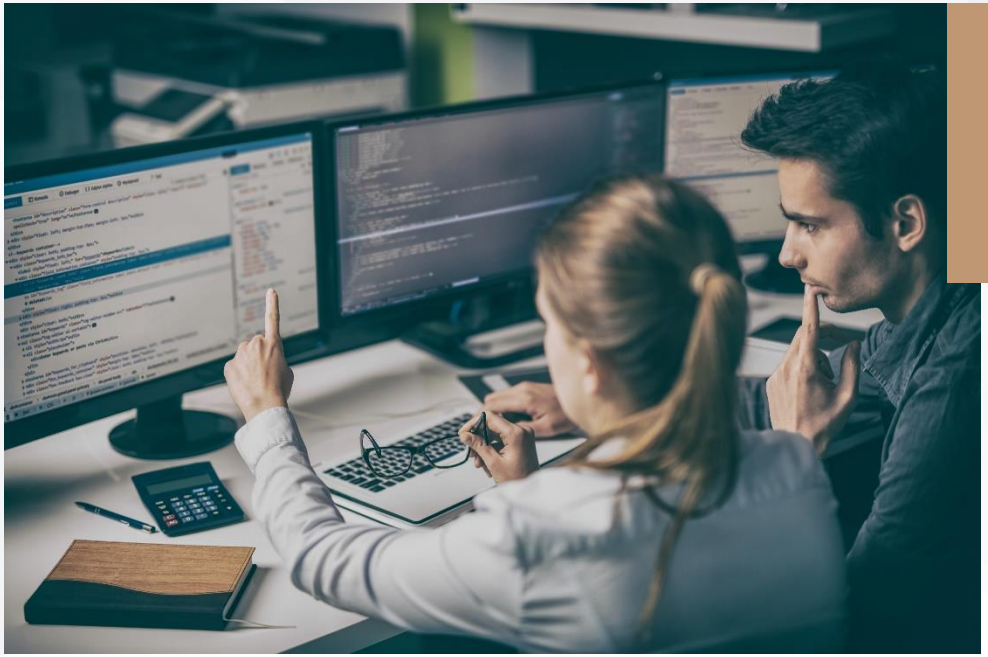
- Recursos inadecuados
- Problemas de liderazgo
- Metas imposibles
- Problemas de la moral del equipo

Ejercicio

Enlista las técnicas que puede utilizar un líder de proyecto para que el involucramiento de todos los miembros sea mas efectivo.

Cierre

TSP te ofrece algunos lineamientos para formar y administrar equipos de trabajo que se vuelvan autodirigidos, es decir, equipos que se encuentran convencidos de que solo trabajando unidos, colaborando y ofreciendo su mutuo apoyo, lograrán cualquier meta que se propongan.



Bibliografía

- Humphrey, W. (2006). TSP(SM) Coaching Development Teams. EE. UU: Pearson Education.
- Humphrey, W. (1999). Introduction to the Team Software Process(SM). EE. UU: Addison Wesley Longman.
- Humphrey, W., Chick, T., Nichols, W. y Pomeroy-Huff, M. (2010). Team Software Process (TSP), Body of Knowledge (BOK). Recuperado de http://www.sei.cmu.edu/reports/10tr020.pdf

Universidad Tecmilenio

Ingeniería de Software

Medidas del software

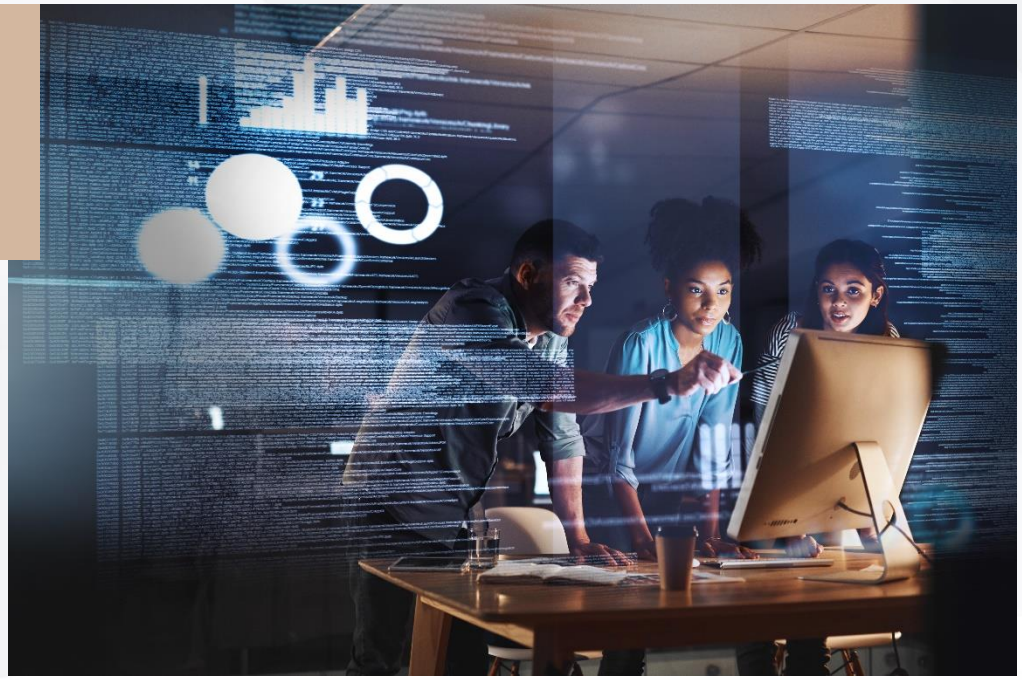
Semana 9

Introducción

El tamaño del software es una medida útil, cuando es precisa, específica, y cuantificable de forma automática.

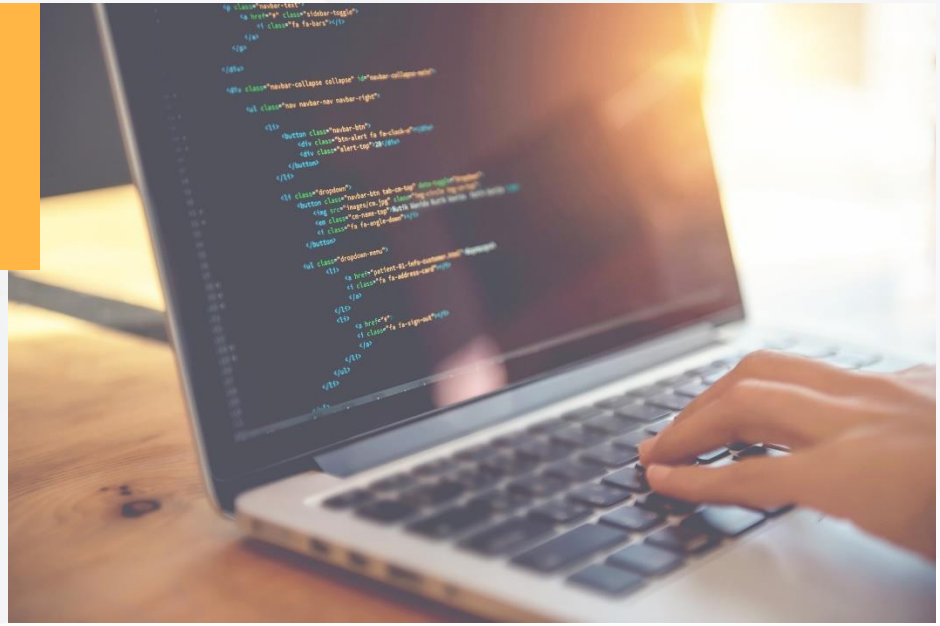
Explicación

Si el tamaño del software es más grande, se necesitará más tiempo para analizar los requerimientos del cliente, el diseño será más complejo, la cantidad de casos de prueba será mayor, al igual que el tiempo que le dediquen a documentar el software.



Explicación

La técnica de puntos de función se basa en medir la funcionalidad que ofrece el software al usuario. Para cuantificar la funcionalidad se requiere descomponer el sistema en partes pequeñas que puedan ser entendidas y analizadas tanto por el usuario como por el programador. Se divide entonces en cinco aspectos: entradas, salidas, archivos, consultas e interfaces que utiliza.



Explicación

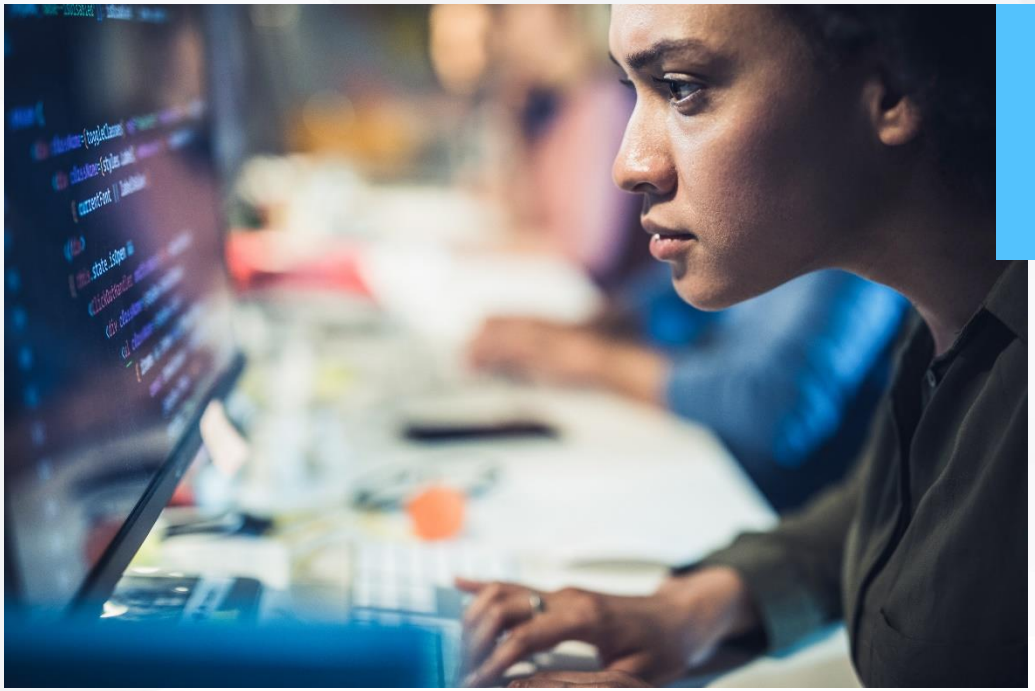
SLOC (Source lines of code) es una medida que está directamente relacionada con el esfuerzo del programador, y probablemente la más sencilla.

Ejercicio

Elaborar un mapa mental con las definiciones necesarias para evaluar el método para contabilizar el numero de líneas de código.

Cierre

En la actualidad, no existe una sola forma de estimar el tamaño del software que se utilizada por la industria de las compañías dedicadas al desarrollo del software. Cada una aplica la técnica que mejor le ha funcionado.



Bibliografía

- Chemuturi, M. (2009). Software estimation best practices, tools & techniques. EE. UU: J. Ross Publishing.
- Humphrey, W. (2005). PSP(SM): A Self-Improvement Process for Software Engineers. EE. UU: Pearson Education.
- IFPUG (2015). Function Point Alignment, Recuperado de: http://www.ifpug.org/Metric%20Views/MetricViewsJan2015.pdf