# 

Profesor:   
Kendy Briceño

Estudiantes:

David Diaz

Alan Argotte

Mauricio Fragachán

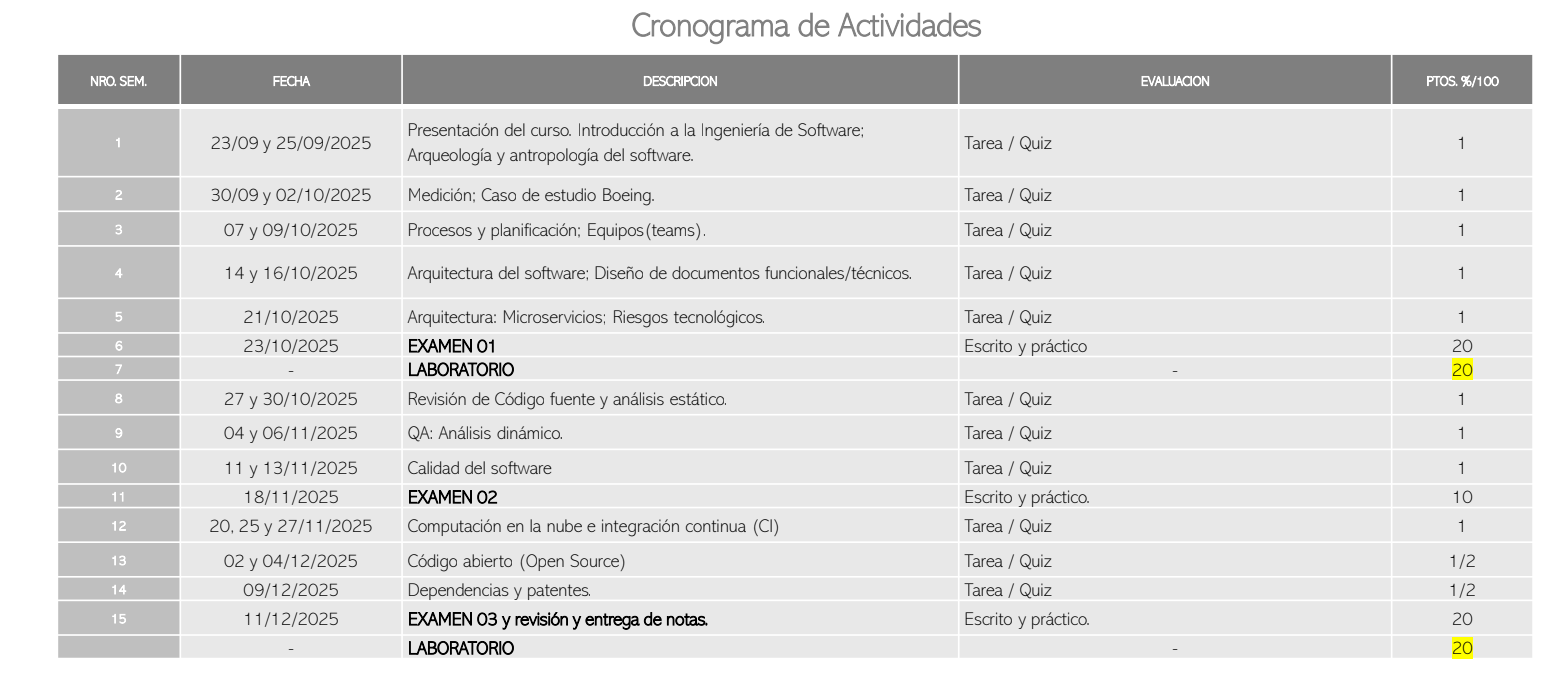
Ian García

José Revete

Jesús Gutiérrez

Documento de Ingeniería de Software

**Planificación**



Para contacto con el profesor:

Oficina 202B

AUL 207

kendybriceno@usb.ve

**Índice**

1. [Clase 1. Título](#Clase_1)

**Clase 01.**

**Clase 02.**

*Análisis del Cap. 1 de Ingenieria de Software en Google*

*Los ingenieros de software no manejan costos, pero sus acciones -aparentemente aisladas- tienen un efecto en la organización que implican costos financieros y de tiempo.*

*La psicología industrial y la parte de talento humano se centran en el ámbito social de una organización. El asunto de las relaciones es importante considerarlo en el desarrollo de software. Por ejemplo, en Google con una armonía en el trabajo se traduce en mayor eficiencia para los ingenieros.*

*La toma de decisiones es un ejercicio personal o hecho por el líder del equipo. A través de los datos se buscan tomar mejores decisiones, a pesar de que no estén siempre presentes. Muchas veces los datos no siempre nos dan un indicio de lo correcto que se debe realizar, pues los datos cambian y aquello que era verdad en un instante ya no aplica para el momento actual.*

*Programar debe ser un proceso eficiente. Casi siempre nunca comenzamos desde 0 y debemos analizar el código fuente base del cual se basa para crear una nueva solucion. Si ya hay algo que existe, deberíamos poder utilizarlo, modificarlo para que haga aquello que queremos hacer, pero no tener mil versiones de un código que hace, en esencia, lo mismo.*

¿Qué es arqueología? Ciencia encargada de cuidar las antigüedades. En el contexto de software, consiste en analizar, probar, escrudiñar código creado por otras personas o equipos en un tiempo pasado. **Analizar, comprender, reconstruir** (se pregunta, ¿qué debe cambiar en el código base? Podemos **reconstruir, cambiar o ajustar**)

¿Qué es antropología? Ciencia, disciplina encargada de estudiar al humano a nivel de cultura. En el contexto de software, se refiere al estudio de culturas y políticas de una organización. Tiene el propósito de estudiar las costumbres para poder usarlas en el momento presente como ayuda para resolver problemas. **Estudiar, entender, mejorar** (a partir de la experiencia de usuario, se define si es satisfactoria la solución del problema a los usuarios)

¿Cómo entender un código fuente? ¿Qué hacemos antes de aplicar esa arqueología del código? Debe uno seguir un estándar: debemos saber qué es lo que queremos hacer: reconstruir, cambiar o ajustar.

¿Cuándo modificas, clonas y posees un repositorio? Para **probar** o **reemplazar** o por **seguridad** (uso local del repositorio para inspeccionar y probar). Respecto al asunto de seguridad, se le conoce como trazabilidad al proceso de verificar el código fuente, encontrar logs, identificar constantes fuera de lo común.

En la práctica hacemos una recolección de información dinámica en la arqueología del software. Es decir, probamos el código. Y probamos el código, modificándolo, con base a la recolección de información estática (documentación, observación, …).

Recolección de información dinámica: probar código fuente verificando las entradas y las salidas.

Recoleccion de informacion estática: observar el código fuente, leer documentación…

Reingenieria: ajuste relacionado a la recolección de informacion dinámica para aplicar ese conocimiento a los cambis en el código fuente.

**Clase 03. Metodologías ágiles y métricas**

**Medición del Software**

La medición es un análisis cuantitativo. En medición del software, con el fin de analizar el comportamiento y mejorarlo u optimizar una pieza de software, debemos buscar las métricas adecuadas para evaluar el acercamiento a los objetivos propuestos.

KPIs (Key Performance Indicator) y OKRs son valores que nos sirven para tomar decisiones. Son indicadores generales del comportamiento de la organización orientado a resultados. Su diferencia radica en que los OKRs miden los resultados respecto a los objetivos propuestos contextualizados en una metodología ágil; los KPIs, son métricas generales para medir los resultados de una organización.

La productividad en desarrollo de software se mide sobre los ingenieros y los equipos de desarrollo.

¿Sobre qué impacta el rendimiento del software? Sobre la infraestructura que soporta ese software

¿Sobre qué medimos la seguridad del software? Sobre su robustez.

¿Y la experiencia del usuario? Qué tan intuitivo es el software para el usuario manejarlo

**Productividad del desarrollador**

* Tiempo de entrega

Métrica: TPE

* Cantidad de código

Métrica: saber cuántas líneas de código tiene ese software nos ayuda a orientarnos sobre la sostenibilidad y mantenimiento del código. Importante: ¿cuáles son las reglas para hacer el conteo de las líneas?

* Trabajo en curso

Métrica: cantidad de tareas en desarrollo. Es una carga individual por cada miembro del equipo. Existen métricas más profundas que también considerar el tiempo y complejidad de las tareas.

* Tasa de éxito del sprint

Métrica: porcentaje de los objetivos alcanzados

* Número de versiones

Métrica: cuántas actualizaciones se han hecho sobre una solución digital. Esta métrica es indicador de la productividad en el desarrollo de software. Sin embargo, es incompleta: uno debe evaluar las otras métricas para determinar si la productividad ha aumentado o ha disminuido.

**Rendimiento del Software**

Mediciones sobre la infraestructura que soporta al software que creamos como ingenieros usando herramientas indicadas. Usualmente uno no requiere hacer estas métricas, sino que usamos estas herramientas para automatizar dicho proceso de medición.

* Tiempos de respuesta. Métrica: tiempo que tarda un componente físico en realizar una operación solicitada. Está relacionado con la memoria y el CPU.
* Tasa de errores. Métrica: conocer tasa de errores que el software tiene durante su ejecución. El software puede tener errores y aun así ejecutar la solución.
* Tiempo de disponibilidad. Métrica: asociado a la estabilidad del software. Es el tiempo de actividad menos los tiempos en el que no está disponible el software (tiempo online menos tiempo offiline).
* Escalabilidad. Métrica: qué tanto ha aumentado el código o la carga ejecutable sostenida por el código. Afecta a la cantidad de código y el número de versiones. **¿Cómo determino la escalabilidad de un software?**

**Defectos y seguridad**

Errores y vulnerabilidades son las métricas que son mayormente evaluadas para analizar los defectos y la seguridad del software.

Tasa de defectos. Métrica técnica. Se hace una evaluación sobre el software, recorriendo las líneas de código para conseguir la tasa de defectos sobre líneas de código. Se relaciona con la métrica de cantidad de código.

Tiempo de resolución de defectos. Métrica táctica operativa. Afecta al trabajo en curso. La resolución de defectos se realiza en un periodo de tiempo distinto al tiempo de desarrollo. Sin embargo, existe una relación entre ambos tiempos a través de la cual se puede obtener una conclusión

Tasa de reaparición de defectos. Métrica táctica operativa. Durante los ciclos de prueba, existe un comportamiento defectuoso que tiende a ocurrir.

Tasa de escape de defectos. Está entre las pruebas integrales y el reléase del software. Evaluando esta métrica uno puede más a o menos saber en qué parte del ciclo de pruebas ocurren los errores. Estas etapas son las etapas de pruebas unitarias, de integración y end-to-end.

**Métricas de seguridad**

A través del hacking ético uno conoce qué tan bien se comporta en términos de seguridad un software.

Número de vulnerabilidades.

Tiempo de respuesta e incidentes. Medida individual hecha para cada vulnerabilidad.

Tasa de parcheo. Relacionada a las métricas de productividad del desarrollador, pues afecta el rendimiento de los equipos. Mide la cantidad de parches hechas sobre el software. Su nombre sugiere que es una solución temporal. Una tasa alta de parcheo sugiere problemas en la producción de software.

Prueba de penetración y análisis de seguridad. Son pruebas de estrés.

**Experiencia de usuario**

Métricas asociadas al uso del usuario cuando utiliza el software.

Tasa de conversión. Métrica evaluada sobre el área de Marketing

Tasa de abandono del cliente. Métrica evaluada sobre el área de Marketing

Retención de usuario. Métrica evaluada sobre el área de Marketing

Tiempo de tarea.

**Clientes asiduos**

Tasa de errores. Ligado con la tasa de conversión y tiempo de tarea.

Tasa de abandono de la plataforma.

**Marco GQM (Goal-Question-Metric)**

Question – Qué preguntas debo hacerme para saber si mi objetivo fue cumplido

Metric – Variable o conjunto de variables que responden a las preguntas.

**Tarea** determinar las preguntas según el marco GQM en el caso de estudio de Formula SAE USB leyendo el ejemplo y generar un paper de este análisis. Mandar a primera hora del jueves y debe mandarlo el líder.