

Evolución de Software Inteligente

Estrategia: No es una materia tan teórica. Está bien tener algunos apuntes de repaso para antes del examen, pero es más importante entregar tareas

▼ Primer Parcial

▼ 09-08-2021

Profesor: Juan Pedro Cardona Salas

Terminología y definiciones de reingeniería

Los ejercicios parten de sistemas de software incompleto

Software en Java Web de altas, bajas y cambios en un negocio de abarrotes

Sobre los parciales

- Primer parcial: Se evalúa terminología e inicio de Moose
- Segundo Parcial: Documentación generada con Moose y avances del software corregido
- Tercer parcial: Entrega final del sistema corregido

En todos los parciales hay exámenes teóricos

▼ 10-08-2021

No hubo clase

▼ 11-08-2021

Instrucciones generales

Sesiones de Teams los miércoles (o con aviso al inicio de la semana)

Puede haber evaluaciones teóricas intermedias que formen parte del teórico del parcial

No hay sesión esta semana, tenemos actividad

Definiciones

Transformación del software con el tiempo

Software: Según la definición del IEEE, citada por [Lewis 1994] "software es la suma total de los programas de computadora, procedimientos, reglas, la documentación asociada y los datos que pertenecen a un sistema de cómputo".

Se espera que el sistema esté orientado a ser usado eficientemente lo que lo convertiría en un producto de software diseñado para un usuario.

Definición de ingeniería de software: Es la rama de la ingeniería que aplica los principios de la ciencia de la computación y las matemáticas para lograr soluciones costo-efectivas (eficaces en costo o económicas) a los problemas de desarrollo de software, es decir, "permite elaborar consistentemente procesos correctos, productos correctos, utilizables y costo-efectivos" [Cota 1994].

Parafraseando las definiciones tenemos

Software	ingeniería de software
Software no solo es código, software también es manuales, software también son los datos, y sobretodo software es la articulación de varios elementos para que pueda ser bien usada por un usuario.	Es la aplicación de otras áreas de conocimiento (matemáticas y ciencias de la computación) para lograr sistemas buenos, bonitos y baratos.

▼ 12-08-2021

No hubo clase

▼ 13-08-2021

No hubo clase

▼ 16-08-2021

Definiciones

Proceso de ingeniería de software: Conjunto de etapas parcialmente ordenadas con la intención de lograr un objetivo, en este caso, la obtención de un producto de software de calidad

Proceso de desarrollo de software: Aquel en que las necesidades del usuario son traducidas en requerimientos de software, estos requerimientos transformados en diseño y el diseño implementado en código, el código es probado, documentado y certificado para su uso operativo

Las habilidades de un ingeniero de software son:

- Technical skills:
 - Programar
 - Aprender
 - Tecnología
- Non technical skills:
 - Comunicación
 - Habilidad del modelado
 - Trabajo en equipo

Abstracción: Capacidad de resumir las cualidades esenciales

La cualidad principal de un ingeniero de software puede ser experiencia y buen juicio pero:

1. La experiencia llega con el tiempo
2. El buen juicio llega al aprender de los errores

▼ 17-08-2021

No hubo clase

▼ 18-08-2021

Ciclos de Vida

Modelos de ciclo de vida:

- Abstracción de las fases o estados por los que pasa un producto de software a lo largo de su vida
- Sucesión de etapas por las que atraviesa un producto de software a lo largo de su existencia
- Representa una posible aproximación a la producción de software
- Debe de:
 - Determinar el orden de las fases
 - Establecer los criterios de transición entre fases

No hay un modelo de ciclo de vida que sirva para todos los proyectos. Esto depende de:

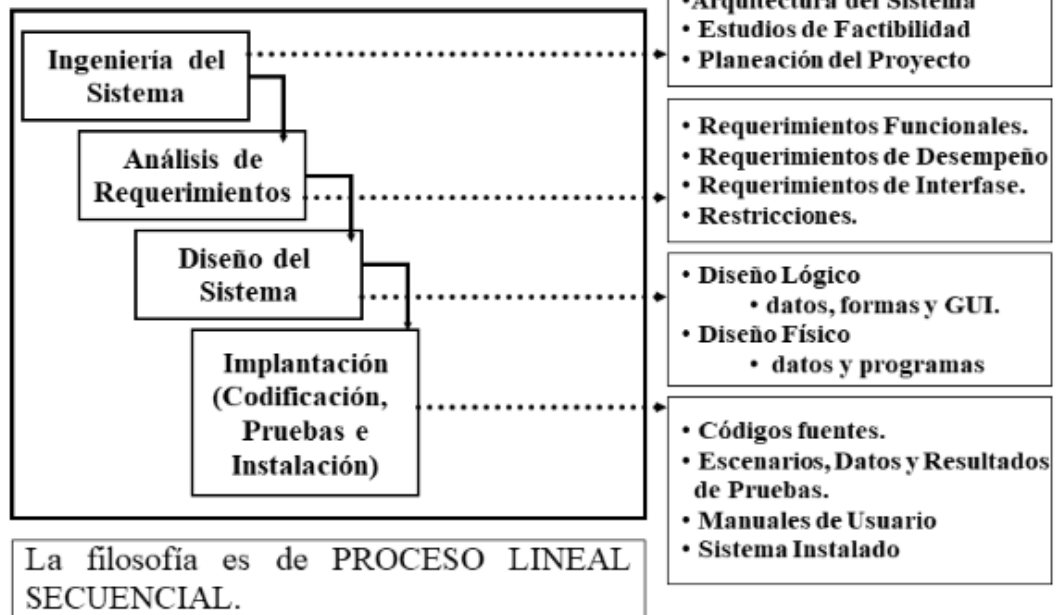
- La cultura de organización
- El deseo de asumir riesgos
- El área de aplicación
- La volatilidad de los requisitos
- Comprensión de los requisitos
- Experiencia previa



Ciclos

▼ Ciclo de Cascada

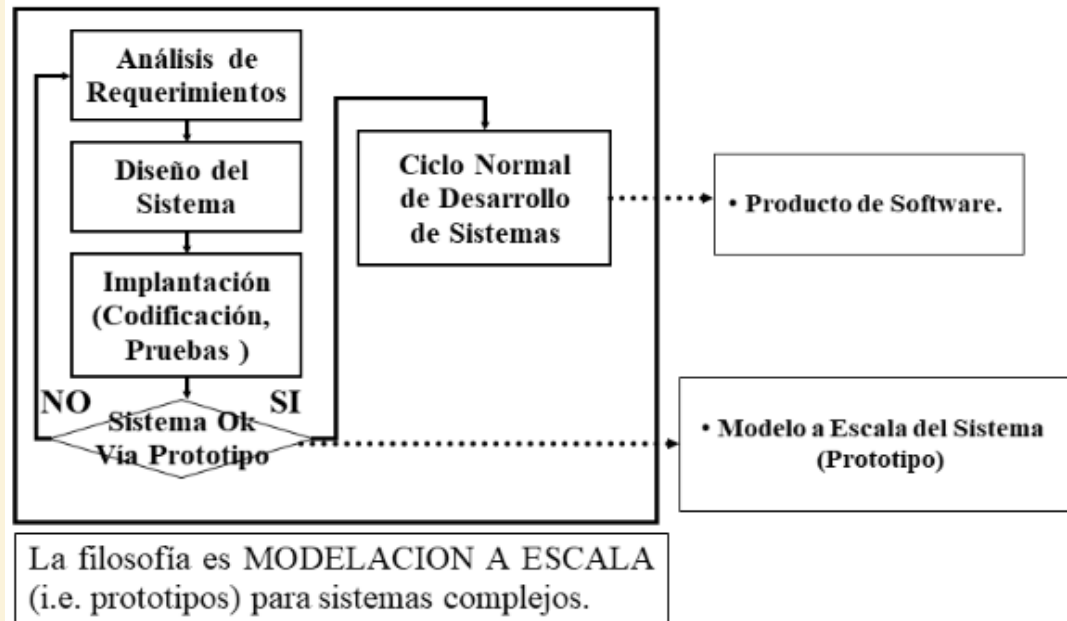
Ciclo de Cascada.



Críticas:

- Los proyectos raramente siguen un flujo secuencial propuesto
- Difícil de establecer todos los requerimientos al principio del proceso
- El software se deteriora y resulta cada vez más difícil de mantener

▼ Ciclo de Prototipeados

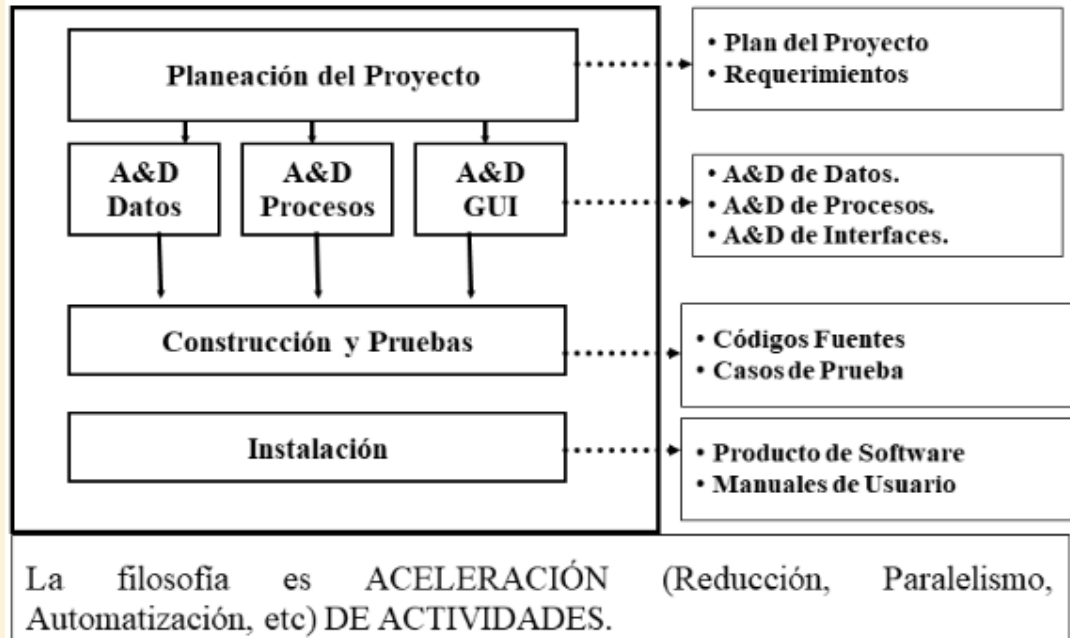


Útil para cuando el cliente no sabe bien lo que quiere o no se comprende bien lo que quiere

Tipos de prototipos:

- Desechable
Sólo se incorporan aspectos peor entendidos
- Maqueta
Ejemplo de interfaz para mostrar entrada y salida
- Prototipo evolutivo
Fácil de ampliar y modificar

▼ Ciclo de RAD (Rapid Application Development)



Fortalezas:

- Reduce tiempo, aumenta productividad y fomenta cohesión grupal
- Involucra al cliente y reduce riesgos

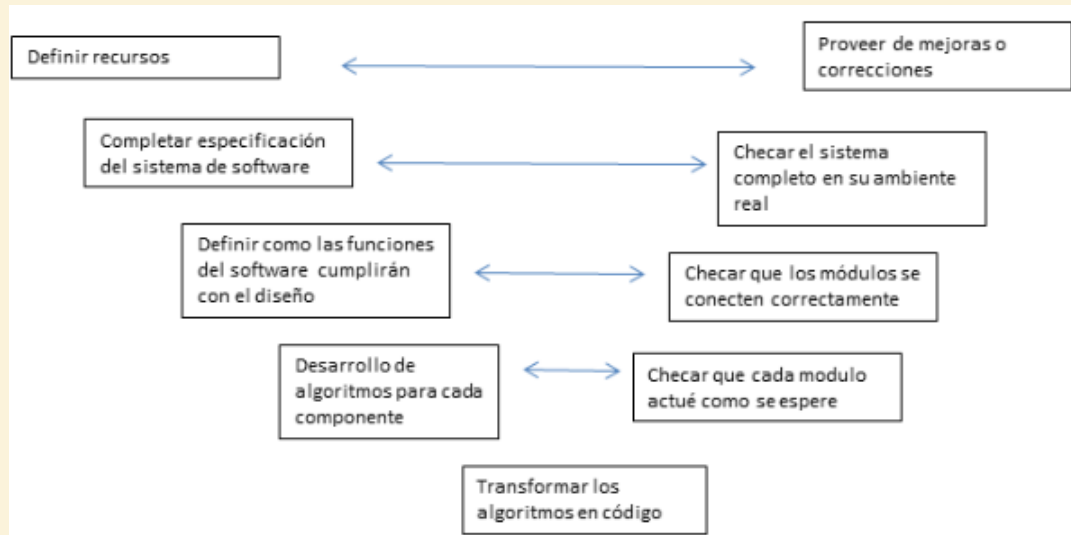
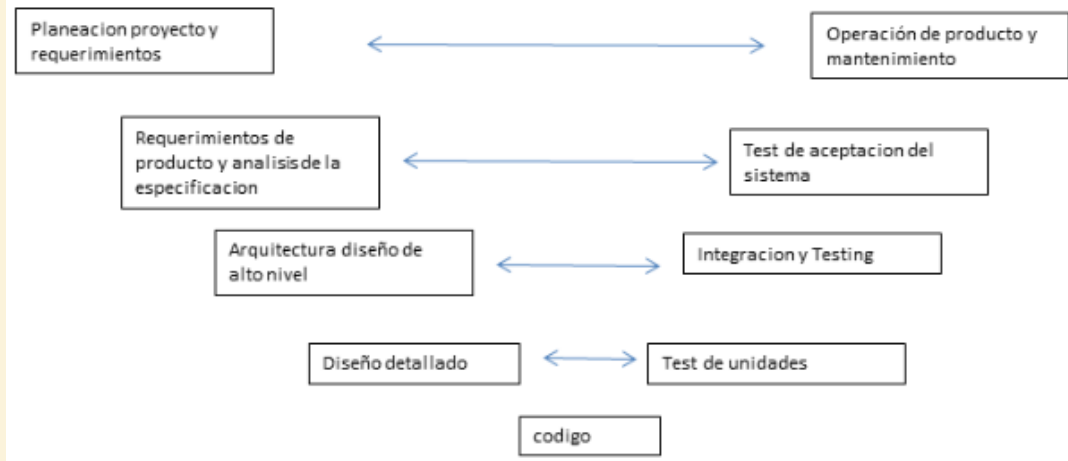
Debilidades:

- Riesgo de nunca cerrar que quede completo
- Difícil para sistemas heredados

Usar RAD cuando:

- El usuario se involucre
- La funcionalidad a entregar es incremental
- No se requiere un sistema de alto rendimiento
- Riesgos técnicos bajos
- Requerimientos conocidos a un 90%

▼ Modelo V



Variante del modelo en cascada

Las pruebas son diseñadas y ejecutadas paralelamente con su correspondiente fase de desarrollo

Fortalezas:

- Énfasis en verificación y validación
- Cada producto es estable
- Sencillo y fácil de usar

Debilidades:

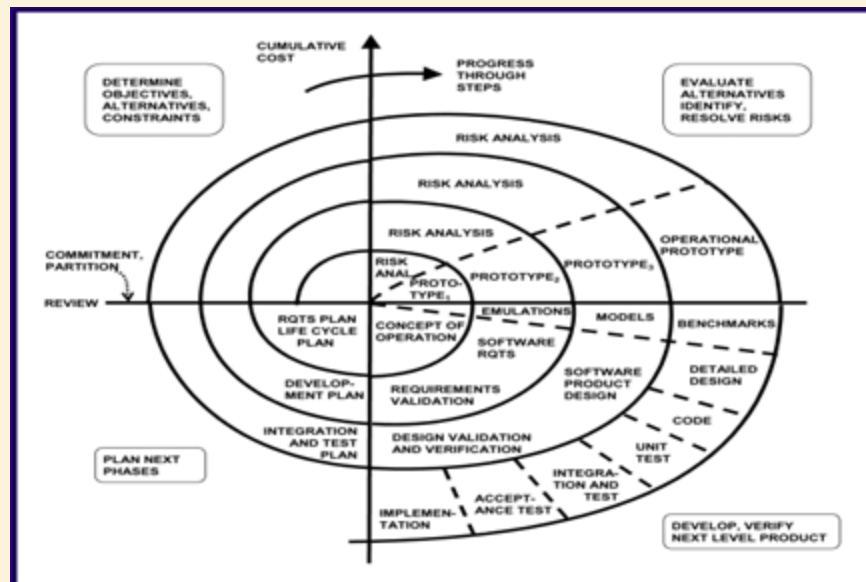
- No es manejable con eventos paralelos

- No maneja iteraciones por fase
- No contiene actividades de análisis de riesgos

El Modelo V es útil cuando:

- Los sistemas requieren alta estabilidad por alto uso
- La mayoría de los requerimientos son conocidos
- Se pueden modificar los requerimientos antes de la fase de análisis y diseño
- La tecnología es conocida

▼ Modelo Espiral



Agrega análisis de riesgos, integra RAD, prototipos y cascada

Cada ciclo involucra la misma secuencia de pasos como el modelo de cascada

1. Cuadrante 1 de la espiral

Determina objetivos, alternativas y restricciones

Objetivos: Funcionalidad, rendimiento, interfaz, determina factores críticos

Alternativas: Construir, reusar, comprar, subcontratar

Restricciones: Costo, tiempo, interfaz

2. Cuadrante 2 de la espiral

Evalúa alternativas, identifica y resuelve riesgos

Identifica y resuelve riesgos

3. Cuadrante 3 de la espiral

Desarrolla el producto al siguiente nivel

1. Crea un diseño
2. Se revisa el diseño
3. Se desarrolla código
4. Se revisa el código
5. Se testea el producto

4. Cuadrante 4 de la espiral

Planea la siguiente fase

Desarrolla un plan de proyecto

Desarrolla un plan de administración de la configuración

Desarrolla un plan de pruebas

Desarrolla un plan de instalación

Fortalezas:

- Ayuda a detectar errores sin mucha inversión
- El usuario ve el sistema temprano por los prototipos
- Retroalimentación temprana
- Los costos acumulados son revisados frecuentemente

Debilidades:

- Tiempo gastado en planear y refijar objetivos
- Puede ser difícil definir objetivos
- Modelo complejo

Es útil el modelo espiral cuando:

- La creación de prototipo es adecuada
- Los costos y evaluación de riesgos es importante
- Hay proyectos con riesgo de medio a alto
- Cambios importantes son esperados
- Requerimientos complejos

- ▼ 19-08-2021

hasta ahora tenemos las definiciones de :

- software
 - ingenieria de software
 - habilidades (skills) ingeniero de software
 - tecnicas y no tecnicas
- la cualidad mas importante el mantenimiento por los costo\$\$\$
- ciclos de vida 6 (repaso) y

tarea preguntas sobre el texto

- 3 preguntas acerca de ciclo de vida utilizar, el foro pendiente
- entregar por equipo maximo 4, con numero de lista,

por hoy solo extendere un poco lo que haremos las siguientes semanas

quedan mas definiciones

les entrego la herramienta Moose

- o la herramienta Moose
- o un sistema de ABC negocio de abarrotes, incompleto
- abarrotes -> informacion UML Unified Modelling

```
quedan mas definiciones
les entrego la herramienta Moose
les entrego un sistema de ABC negocio de abarrotes, incompleto
Moose -> abarrotes      -> informacion UML  Unified Modelling Language
clases, atributo
```

Software Intelligente

quedan mas definiciones en general
definiciones para poder explotar el Moose

Les entrego la herramienta Moose
les entrego un sistema de ABC negocio de abarrotes, incompleto
Moose -> abarrotes -> informacion UML Unified Modelling Language
clases, atributos, packages, clases esta usando
reporte entregas parciales observaciones 2do parcial
despues del reporte analisis
programacion de abarrotes hasta corriendo

Todo Moose es en equipo

▼ 20-08-2021

No hubo clase

▼ 23-08-2021

Habilidades de un Ingeniero de Software

- Buen juicio y experiencia
- Abstracción
- Modelación
- Mantenimiento de software

Definiciones

Evolución de software: El 90% del costo promedio se da en la fase de mantenimiento, aun cuando el sistema sea exitoso y apegado a los requerimientos

Mantenimiento: Modificar el software preservando su integridad

Tipos de mantenimiento de software:

- Mantenimiento adaptativo: Modificación de un producto de software que después de entregado se va a modificar por cambios en el entorno de software (migración, actualización por tecnología o equipo hardware) (18%)
- Mantenimiento perfectivo: Modificación de un producto de software por nuevos requerimientos (60%)
- Mantenimiento correctivo: Modificación reactiva de un producto de software que en su uso se descubrieron problemas (17%)
- Mantenimiento preventivo: Modificación de un producto de software que después de entregado se va a modificar por problemas potenciales que estos puedan hacerse realidad (5%)

▼ 24-08-2021

Leyes de Lehman

Evolución de los costes del mantenimiento de software:

Fechas	% Mantenimiento
Años 70	35% - 40%
1980 – 1984	55%
1985 – 1989	75%
Años 90	80% - 90%

Tipos de software

- **Specify-program**: Escrito de acuerdo a una especificación exacta de qué hace el programa
- **Problem-program**: Escrito para implementar ciertos procedimientos que completamente determina lo que el programa puede hacer
- **Evolving-program**: Escrito para ejecutar algunas actividades del mundo real. Fuertemente ligado a ambientes cambiantes y necesita adaptarse a requerimientos variables

Leyes de Lehman de la evolución del software

Las siguientes características del software tarde o temprano lo harán obsoleto:

1. **Cambio continuo:** El sistema es continuamente adaptado y progresivamente llega a ser menos satisfactorio
2. **Incremento complejidad:** Se vuelve más complejo y difícil de manejar hasta que se efectúe un mantenimiento de rediseño
3. **Crecimiento continuo:** Debido al crecimiento, el mantenimiento debe ser continuo a fin de mantener su funcionalidad y satisfacción al cliente
4. **Declinación de la calidad:** La calidad del sistema va en declive aún con un mantenimiento riguroso
5. **El sistema retroalimenta al usuario:** El sistema genera nuevas expectativas por los resultados positivos alcanzados, pero algunas de esas nuevas expectativas no aportan al objetivo y se empieza a contaminar el sistema con funcionalidades innecesarias

Pizarra del martes

Repaso de los apuntes

el otro punto critico del software es el mantenimiento

en el libro de Fred Brook "Mythical man month" señala el mantenimiento como lo mas importante, y es porque ocupa la mayoria y la mayoria de los co\$to\$

fred brook tambien señala man-month productividad, esta productividad no es lineal 1 semana hace 1,000 lineas en 4 semanas haras 4,000 FALSO depende de otros factores como complejidad, entendimiento del problema, de los requerimientos

les subi el libro completo de Fred Brooks

si el mantenimiento es importante entonces ubicar los tipos de mantenimiento

1 adaptativo: cambios como migracion

2

clasificación de tipos de software

S-program o Specify-program son programas

P-program son mas como rutinas, no mucho mantenimiento

E-program Evolving program, mas como sistema, es un conjunto de decisiones aplicadas a un objetivo, y como los objetivos cambian en las instituciones, entonces van a necesitar mas trabajo de mantenimiento

leyes de Lehman de evolucion de software, parafraseando son las causas de cambio del software

1 cambio continuo, requerimientos cambian

2 incremento complejidad

3 crecimiento continuo

4 declinación de la calidad, necesidad de mantenimiento

5 sistema retroalimenta al usuario (dueño o programador)

ya logramos algunos objetivos del sistema, y ahora creemos que podemos

AGREGAR algo mas, crecimiento del sistema requiere mas mantenimiento

no siempre es recomendable I

▼ 25-08-2021

El error

Un tipo de error tiene gran influencia en la degradación de software o declinación de la calidad de software y se le conoce como emparchamiento que es una corrección sobre corrección

- Problema \implies Especificación correcta de requerimientos \implies Diseño correcto \implies Programa correcto \implies Funciones correctas \implies Producto perfecto
- Problema \implies Especificación incorrecta de requerimientos \implies Diseño basado en especificación errónea \implies Programa basado en especificación errónea \implies Errores ocultos \implies Producto imperfecto
- Problema \implies Especificación correcta de requerimientos \implies Diseño erróneo \implies Programa basado en un diseño erróneo \implies Errores incorregibles y soluciones a base de parches \implies Producto imperfecto
- Problema \implies Especificación correcta de requerimientos \implies Diseño correcto \implies Programa correcto \implies Funciones correctas \implies Producto imperfecto

Los errores corregibles “nacen” en la fase de código

Los errores incorregibles se generan en la fase de diseño

Errores ocultos provienen de una especificación errónea

Sistemas heredados

Sistema heredado: Sistema informático que ha quedado anticuado pero continúa siendo utilizado por el usuario y no se quiere o no se puede reemplazar o actualizar de forma sencilla

Un sistema es antiguo cuando es mayor a 10 años

Reemplazar un sistema heredado es arriesgado porque

1. No hay especificación
2. Las reglas de negocio no están documentadas
3. Los procesos de negocio y los de sistema pueden no ser correspondientes
4. Puede ser que haya diferentes implementaciones en diferentes lenguajes y equipos (algunos obsoletos)
5. Los datos pueden estar en distintos equipos en distintos formatos

▼ 26-08-2021

Fases:

1. Especificación
2. Diseño
3. Programación

Problema la definición			
1 Especificación correcta Requerimientos		2 Especificación incorrecta Req.	
Diseño correcto		Diseño erróneo	Diseño basado en especificación errónea
Programa correctos	Programas erróneos	Programas basados en un diseño erróneo	Programas basados en especificación errónea
Funciones correctas	Errores corregibles	Errores incorregibles solución local Parche	Errores ocultos Ni tan ocultos salen en pruebas Ni tan error, no falla, solo no hace lo que debe
producto perfecto	Productos Imperfectos		

Si consideramos las fases como :
 1 especificación
 2 Diseño
 3 programación

Si el error es en programación, te regresas una fase y lo corriges, **no tan grave**
 Si el error es en diseño, te regresas 2 fases y lo corriges, **es más grave**
 Si el error es en especificación te regresas 3 fases y lo corriges **es muy grave**

Todos estos casos son códigos corregibles por que el problema fue bien aclarado, entendido

Tarea

Tarea contestar un cuestionario de manera individual de los apuntes de la semana

Lo suben al foro preguntas

PDF contener preguntas y respuestas no. De lista (si no tiene numero, o si se equivocan de numero)

▼ 27-08-2021

No hubo clase

▼ 30-08-2021

Síntomas de Degradación de Software

1. **Pollution**

El sistema incluye varios componentes que no son necesarios y que no aportan a los objetivos del negocio

2. **Embedded knowledge**

El conocimiento de la aplicación y su evolución está tan esparcido en los programas que no se pueden generar en la documentación

3. **Poor lexicon**

Los nombres de los componentes tienen poco significado o son muy inconsistentes con el significado de los componentes

4. **Coupling**

Los programas y los componentes están demasiado unidos por llamadas y flujos de ejecución y redes de datos. Coupling es el grado de interdependencia de un sistema de software, mayor acoplamiento significa mayor complejidad y más difícil de entender y mantener

5. **Layered architectures**

Es cuando la arquitectura del sistema consiste en varias soluciones que no pueden ser distinguidas o identificadas, aún con una arquitectura eficiente inicial, la superposición de otras soluciones y/o arquitecturas durante el mantenimiento daña la calidad del sistema

Alternativas de un sistema heredado

1. **Mantener el sistema heredado**

Prolongar el tiempo de vida de los sistemas heredados mediante la estandarización y documentación de procesos, optimización de flujo de datos (reducir fragmentación), rutinas de manejo de datos factorizadas, de manera de mantener funcionalidad y separación de datos e interfaces, procesamiento de reglas de negocio modulares con soluciones costo-efectivas

2. **Convertirlo a un sistema integrado y monolítico**

Reemplazar los sistemas legados, con proceso de negocio estándares para los diferentes requerimientos de la organización, como ERP

3. Utilizar arquitectura orientada a servicios

Aplicaciones publicadas como servicios sin tener que depender de una macroestructura tan compleja, aun así hay riesgos

4. Wrapping

Inicialmente no tocar código sino conectar dinámicamente una nueva presentación y alguna transformación de datos, y paralelamente realizar ingeniería inversa y reingeniería es difícil y riesgoso

▼ 31-08-2021

Términos de reingeniería

Forward Engineering: Proceso tradicional de convertir requerimientos a partir de abstracciones lógicas de alto nivel hasta el diseño y posteriormente a la implementación física

Reverse Engineering: Regularmente aplicado a mejorar un producto. Proceso de analizar un sistema para identificar sus componentes y sus interrelaciones

Reengineering: Análisis y alteración de un sistema para reconstruirlo en otra forma e implementación de esa nueva forma

Reverse engineering y reengineering son tipos de mantenimiento

Objetivos de ingeniería inversa

Reducir complejidad de sistemas grandes

Redocumentación:

- Generador de diagramas UML
- Generador referencias entre clases y paquetes para identificar los cambios, impactos y ramificaciones para aumentar el entendimiento y aumentar el reúso de código

Objetivos de reingeniería

Unbundling: Modularizar, dividir en módulos los sistemas monolíticos

Rendimiento: Que haga lo que haga, bien y rápido

Port: Aumentar interacción con otras plataformas, portabilidad, wrapping

Extracción de diseño para mejorar mantenimiento y portabilidad

Recuperación de diseño: Obtener interrelación entre clases y métricas de software como líneas de código, número de parámetros, coupling, con herramientas de visualización que pueden ser analizadores estáticos o analizadores dinámicos (tracers)

Explotación de nuevas tecnologías, lenguajes, estándares y librerías

Reingeniería de datos: Actualizar modelos mediante normalización y evitar redundancia de datos

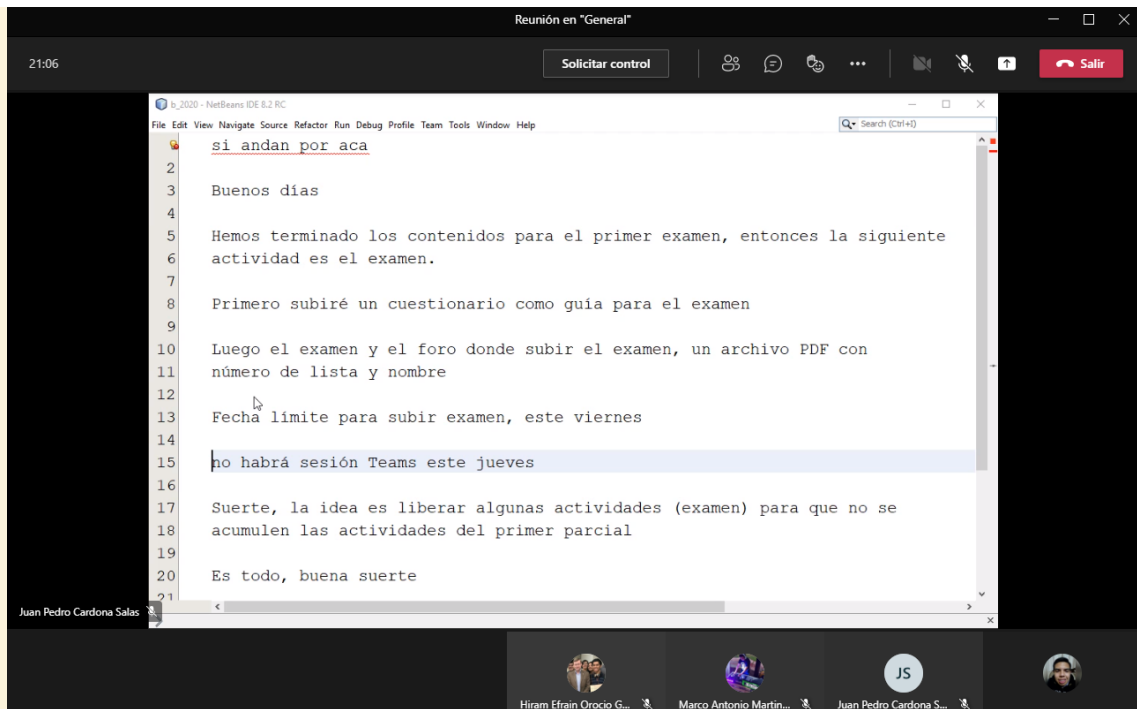
Técnicas de reingeniería: Reestructuración de código

1. Mediante actualización automática de código
2. Mediante refactorización

Pizarra del martes

Repaso de ayer y hoy

▼ 01-09-2021



🔥 AVISO DE EXAMEN 🔥

Contenidos terminados para el primer examen

Cuestionario como guía del examen

Fecha límite para subir el examen: El viernes

No habrá sesión el jueves

Cuestionario de repaso

Archivo 1_22

🔥 EXAMEN 🔥

Archivo 1_23

▼ 02-09-2021

Realización del examen

▼ 03-09-2021

▼ Segundo Parcial

▼ 06-09-2021

Moose: Herramienta para evolución de software. Extrae información de un sistema de software en Java. Es del tipo de analizador estático, al contrario de un analizador dinámico que obtiene información de un sistema cuando está en ejecución. Significa alce

- Plataforma de software y análisis de datos
- Ayuda a programadores en análisis rápidos y sencillos
- Basado en el lenguaje Pharo es código abierto bajo BSD/MIT
- Pharo es un lenguaje de programación basado totalmente en objetos como Smalltalk

MSE: Formato genérico para importar-exportar de Moose y para expresar modelos en general, similar a XML en vez de <> usa () guardando modelos FAMIX

FM3: Es el meta-meta model en que se basa Moose. Representación abstracta en términos de reducción

FAMIX: a Familia de meta-modelos para análisis de software, el análisis es de tipo estático, es decir, es sobre código fuente (sin ejecutar) y orientado a sistemas orientados a objetos

▼ 07-09-2021

Inconvenientes del profesor y se canceló la clase

▼ 08-09-2021

Actividad

Extraer información del sistema de software en Java existente
"ProjectAbarrotes.zip"

No tenemos información a nadie para preguntar sobre este sistema (como un sistema real de legado) pero es un sistema web Java y es un sistema CRUD

Verveinej es el apellido de un programador alemán que hizo una herramienta parser llamada "Verveinej-Master"

PENDIENTE

▼ 09-09-2021

practica es obtener 1 archivo .mse = modelo informacion de un sistema de software en java

bajar de Aula virtual bajan verveinej tool (carpeta)
bajar a c:

bajar de aula virtual project abarrotos
proyecto en netbeans, entran a la carpeta
 build
 lib

entran a carpeta src y de copian la carpeta java

paso 3
la carpeta copiada java la dejan en la carpeta verveinej

paso 4
entran a linea de comando cmd la pantalla de ms-dos
entran a la carpeta verveinej
le dan comando verveinej.bat java

el error "en este momento" es porque la carpeta debe estar en raiz, no estar en otra carpeta

el resultado un archivo output.mse moose 6.0 moose 6.1
archivo explicatorio output.mse los integrantes list y nombre
y lo pasan a PDF foro

▼ 10-09-2021

No hubo clase

▼ 13-09-2021

No hubo clase por seminario de servicio social

▼ 14-09-2021

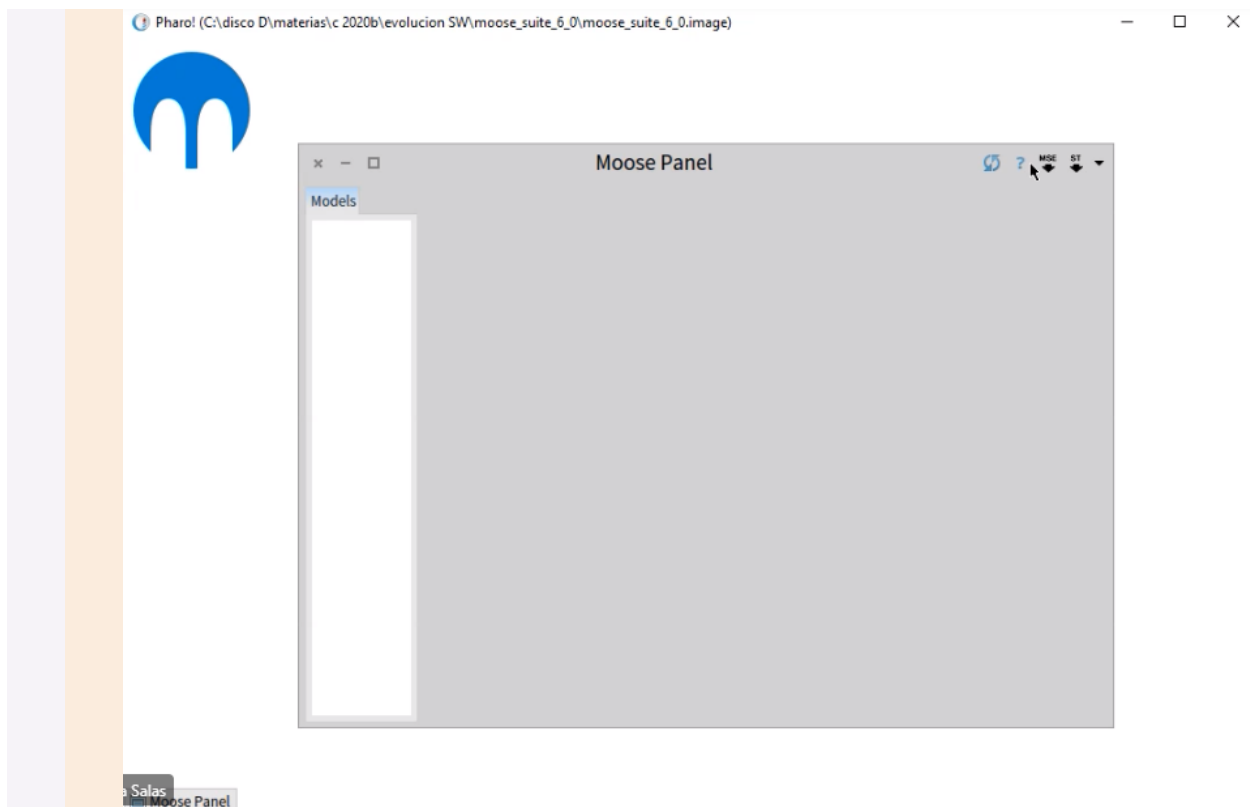
1era tarea

la actividad de generar el archivo output.mse
informacion de un sistema hecho en java
lo generamos con verveinej.extractor
verveinej
verveinej-master
las herramientas esta en aulavirtual

tarea 2

2da herramienta Moose, link del sitio 6.1 no es estable
pero en aula virtual ya deje google drive para bajar
la carpeta (no se instala)

entramos a la carpeta moose 6.0 que estan bajando
desde windows explorer ejecutan Pharo.exe plataforma
desde ahi corre Moose



Tarea 2: Cargar output desde Moose

tarea 1 output en el encabezado nombres y no. de lista
tarea 2 evidencia con screenshot de la pantalla de Moose
habiendo cargado el archivo output.mse|

▼ 15-09-2021

Instalación de Moose

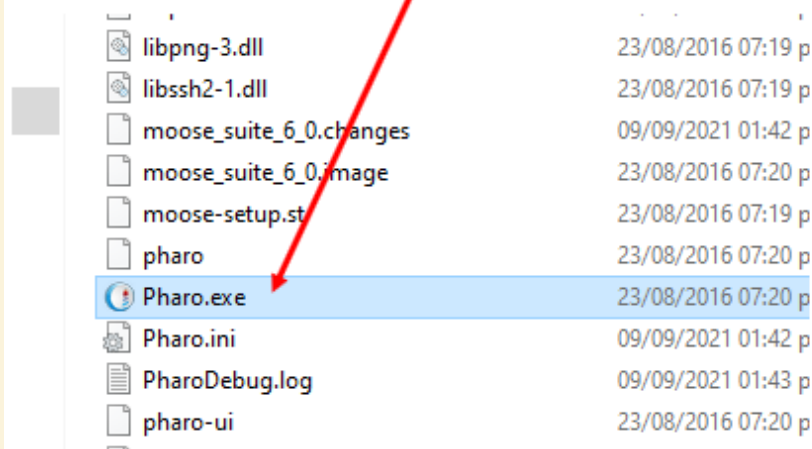
La última vez dije que les daría la liga del sitio oficial para bajar la herramienta Moose pero ... La versión que está en el sitio oficial es versión 6.1 y la probé y no funciona muy bien

Yo tengo la versión 6.0 que si funciona y desde aquí la pueden bajar

https://drive.google.com/drive/folders/1N2O-M0coWpaw5fmiQ_KzUHMQC8McF-ts?usp=sharing

es una carpeta y no es necesario instalar, solamente se corre desde Windows explorer

ejecutan Pharo.exe, este de aquí



Y aparece esto, le dan click aquí para leer el archivo output.mse que ustedes extrajeron en la anterior clase



▼ 20-09-2021

No hubo clase

▼ 21-09-2021

All model classes - All model classes (16)



Nombre de cada uno de ellos

▼ 22-09-2021

Revisión del documento Moose Options LUNES

Explicación de la tarea Moose de ayer

Lo mismo para reporte complexity

Complementa la explicación de la tarea Moose de ayer

▼ 23-09-2021

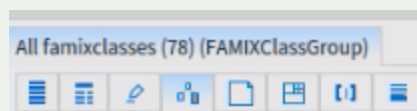
Trabajar en el portafolio

Investigar qué es ORM Hibernate

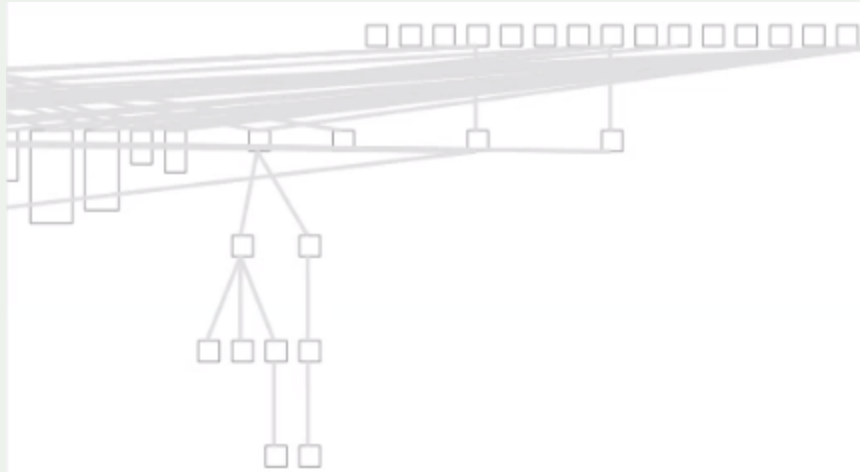
La altura del nodo (clase) indica el numero de métodos
La anchura del nodo (clase) es el numero de atributos
El color mas oscuro es mas líneas de código
La mayoría de las clases tiene mucho código
Pero podemos suponer (a reserva que lo verifiquen ustedes) que es mucho código de get set,
no representa mucho problema

Ahora:

All classes - All famixclasses (78)



Observamos esto



▼ 24-09-2021

No hubo clase

▼ 27-09-2021

No hubo clase

▼ 28-09-2021

ayer se vio un reporte de Moose de cada paquete y sus clases

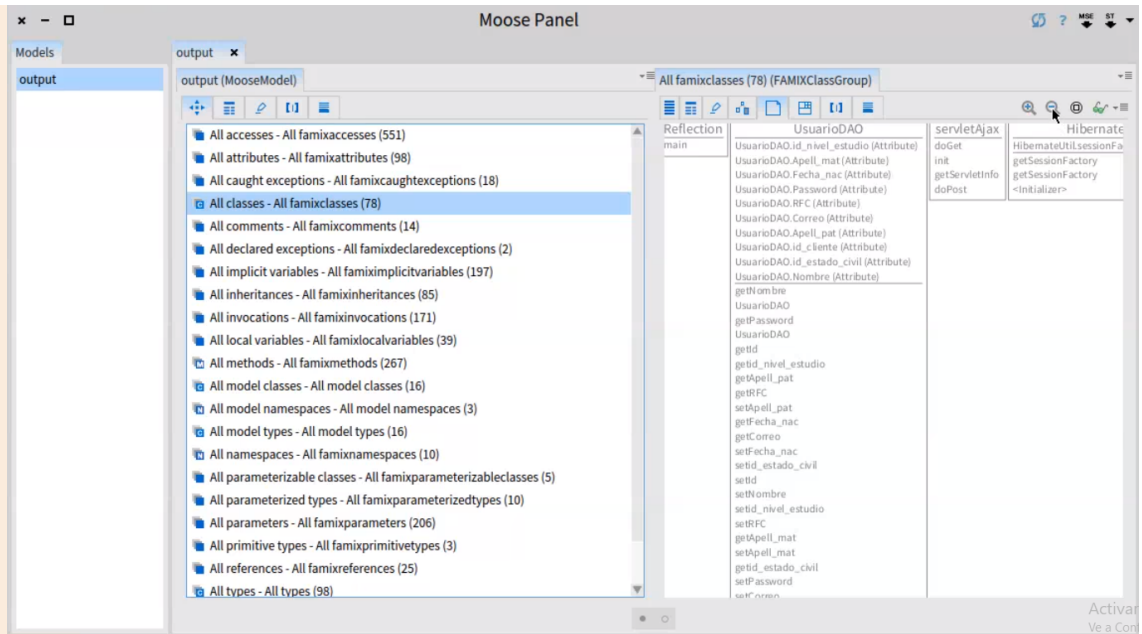
yo mismo las clases por paquete pero el reporte entonces hay que bajarlo e incluirlo a portafolio

lo voy a hacer

lo de hoy es obtener las graficas UML Unified Modelling Language
cuadro con atributos String Integer
metodos

All classes

All famix classes



apunte en aula Virtual

hoy

hay que obtener 9 clases diagrama UML paquete mapeos

6 UML de paquete Beans

el apunte ya esta en aula virtual

la tarea, se pueden repartir la tarea, el jefe de equipo decide

pero va a tomar algo de tiempo

entonces proximo jueves no hay sesion T

hoy martes
hay que obtener 9 clases diagrama UML paquete mapeos
6 UML de paquete Beans
el apunte ya esta en aulavirtual

la tarea, se pueden repartir la tarea, el jefe de equipo decide

pero va a tomar algo de tiempo
entonces proximo jueves no hay sesion Teams
van a subir el portafolio actualizado

complexity
global
nest paquete-clase
UML **class diagram**

le van a quitar comentarios mios, y sera la primera revision de portafolio, fecha limite el viernes

▼ 29-09-2021

No hubo clase

▼ 30-09-2021

No hubo clase

▼ 01-10-2021

No hubo clase

▼ 04-10-2021

No hubo clase

▼ 05-10-2021

Sesión que dijo el profesor que repetiría la próxima semana

▼ 06-10-2021

No hubo clase por el Congreso de Ciencias Exactas

▼ 07-10-2021

No hubo clase por el Congreso de Ciencias Exactas

▼ **08-10-2021**

No hubo clase por el Congreso de Ciencias Exactas

▼ **11-10-2021**

No hubo clase

▼ **12-10-2021**

No hubo clase

▼ **13-10-2021**

No hubo clase

▼ **14-10-2021**

Explicación del examen

▼ **15-10-2021**

No hubo clase

▼ **18-10-2021**

Buenos días

Fui notificado de que comenzamos presencial martes y jueves de cada semana, entonces el examen será entregado en sesión presencial el próximo martes 19 octubre en el laboratorio del edificio 61, a las 13:00 horas.

La entrega del examen será la única actividad de este martes, los que no puedan o no quieran ir, se ponen de acuerdo para que lo entregue un compañero de clase.

▼ **19-10-2021**

Entrega de exámenes presencial

▼ **Tercer Parcial**

▼ **20-10-2021**

No hubo clase

▼ **21-10-2021**

No hubo clase

▼ **22-10-2021**

No hubo clase

▼ **25-10-2021**

No hubo clases por la evaluación del segundo parcial de Procesamiento de Imágenes

▼ **26-10-2021**

Avisos

▼ **27-10-2021**

No hubo clase

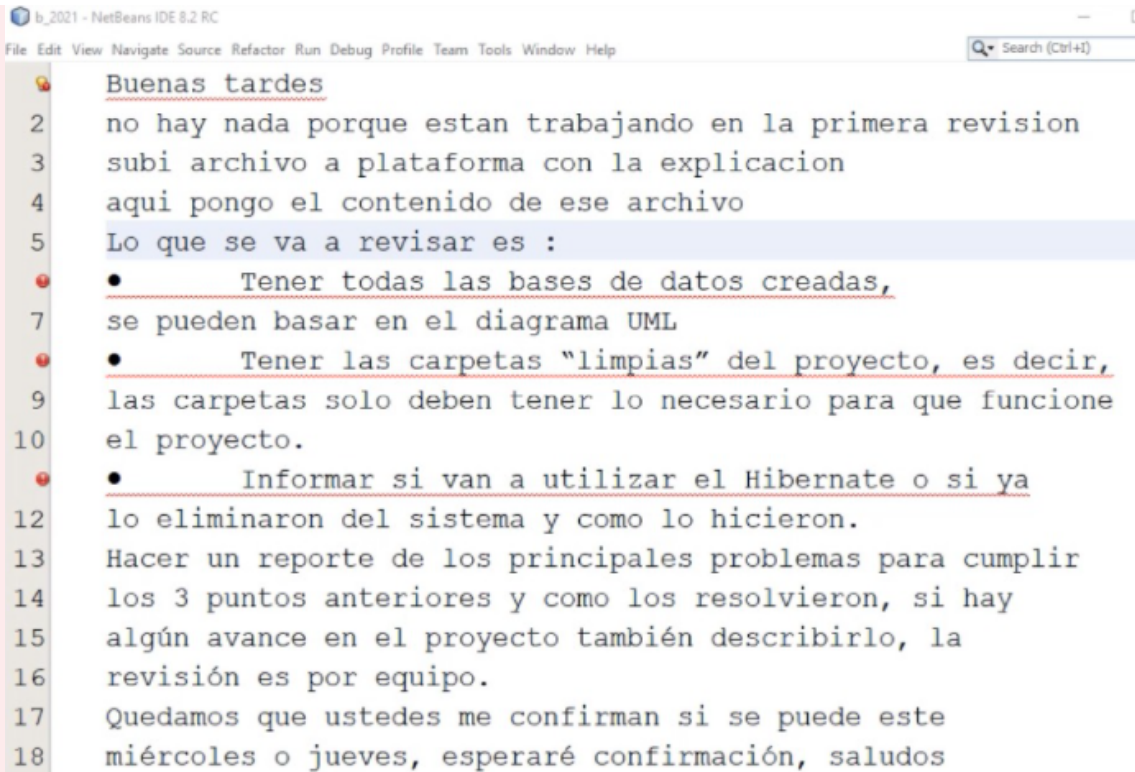
▼ **28-10-2021**

No hubo clase

▼ **29-10-2021**

No hubo clase

▼ **01-11-2021**



```
b_2021 - NetBeans IDE 8.2 RC
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Buenas tardes
2 no hay nada porque estan trabajando en la primera revision
3 subi archivo a plataforma con la explicacion
4 aqui pongo el contenido de ese archivo
5 Lo que se va a revisar es :
6 • Tener todas las bases de datos creadas,
7 se pueden basar en el diagrama UML
8 • Tener las carpetas "limpias" del proyecto, es decir,
9 las carpetas solo deben tener lo necesario para que funcione
10 el proyecto.
11 • Informar si van a utilizar el Hibernate o si ya
12 lo eliminaron del sistema y como lo hicieron.
13 Hacer un reporte de los principales problemas para cumplir
14 los 3 puntos anteriores y como los resolvieron, si hay
15 algún avance en el proyecto también describirlo, la
16 revisión es por equipo.
17 Quedamos que ustedes me confirman si se puede este
18 miércoles o jueves, esperaré confirmación, saludos
```

▼ 02-11-2021

No hubo clase. Asueto

▼ 03-11-2021

No hubo clase

▼ 04-11-2021

No hubo clase

▼ 05-11-2021

No hubo clase

▼ 08-11-2021

Explicación de la entrega otra vez

▼ 09-11-2021

Repetición de lo que viene repitiendo el profe lo que ha dicho las últimas dos semanas

▼ **10-11-2021**

No hubo clase

▼ **11-11-2021**

No hubo clase

▼ **12-11-2021**

No hubo clase

▼ **15-11-2021**

No hubo clase. Asueto

▼ **16-11-2021**

No hubo clase

▼ **17-11-2021**

No hubo clase

▼ **18-11-2021**

No hubo clase

▼ **19-11-2021**

No hubo clase

▼ **22-11-2021**

No hubo clase

▼ **23-11-2021**

Revisión parcial del proyecto

▼ **24-11-2021**

▼ **25-11-2021**

▼ **26-11-2021**