Aux. Eddy Escalante Ustariz

#73650500

CONCEPTOS IMPORTANTES

1. ¿Qué es un sistema de información basado en computadora?

Es un conjunto de elementos y recursos interrelacionados con el fin de lograr un objetivo en común y que usa como principal herramienta para su funcionamiento una computadora.

La finalidad del SIBC es captar, almacenar y distribuir información del entorno de una institución y de sus operaciones internas para el propósito de apoyar a las áreas o funciones de la institución y a la toma de decisiones, a la comunicación, control, etc.

2. Diferencia entre SIBC y un SI:

Es el tiempo costo y esfuerzo.

3. ¿Qué es un sistema?

Es un conjunto de elementos o partes que interactúan entre sí para lograr un propósito

4. ¿Qué es información?

Es una colección de datos organizados y es el resultado de procesar datos y se convierten en información y estos se convierten en conocimiento

5. ¿Qué es una computadora?

Es cualquier dispositivo electrónico que consta de un procesador y memoria.

6. 4 Niveles de requerimiento de un software

- **Requerimiento**: Representa lo que el SW va a hacer y se lo representa en forma natural o en un lenguaje natural, en la ETAPA DE DEFINICIONES.

- **Requisito**: Representa lo que el SW debería hacer, son los resultados de PROCESAR los requerimientos**,** aparece en la ETAPA DE REQUISITOS, y se lo representa en MODELOS (diagramas de casos de uso) para ser entendido por los desarrolladores. (La diferencia entre Requerimiento y Requisito es el TIEMPO).

- **Especificaciones de Requisitos de SW:** Es una descripción completa y detallada de cada uno de los casos de uso, aparece en la ETAPA DE ANÁLISIS.

- **Funciones:** La implementación de cada uno de los requisitos, también son las especificaciones representadas en un lenguaje de programación, aparece en la ETAPA DE IMPLEMENTACION.

  Lo común es que los 4 hablan de la funcionalidad del software, la diferencia es el tiempo en que se usan.

7. **Metodología**
   - Conjunto de procedimientos técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a desarrollar un nuevo software
   - Una metodología puede seguir una o varias modelos de ciclo de vida es decir el ciclo de vida indican que es lo que hay que obtener a lo largo del desarrollo del proyecto pero no como hacerlo. La metodología indica cómo hay que obtener los distintos productos parciales o finales.
8. **Método**

   Tiene dos elementos
   - Debe definir un método (conjunto de pasos)
   - Debe definir un lenguaje, un conjunto de notaciones para representar resultados. Un método define un proceso.

9. **Modelo**

   Es un ámbito de desarrollo de SW, también es una representación abstracta de la vida real, abstracción es la capacidad de razonar y entender.

10. **Caso de uso**
    - Es un requisito de SW ≈ requerimiento del cliente, también es un hecho completo, es un requisito funcional.
    - Siempre representa un requisito que el SW debe realizar, debe ser una función del SW.
    - Siempre es un hecho totalmente completo, es un conjunto de tareas, siempre es un proceso.
    - Se debe a 4 estructuras: En paralelo, Iterativos, Secuencial, Alternativa.

11. **Escenario**

    Un conjunto ordenado de flujos desde el inicio hasta una de las salidas del caso de uso.

    Ejemplo

    En un sistema de RRHH (recursos humanos), para el caso de uso “Contratar Empleado”, los siguientes escenarios pueden darse:

    * **Escenario norma**l: Contratar una persona que no pertenece a la compañía
    * **Escenario alternativo**: Contratar a una persona que pertenece a la compañía, pero en un departamento distinto.
    * **Escenario fallido**: No pude ser encontrada ninguna persona calificada.

12. **Flujo de suceso**

    Describe como interactúa el sistema con los actores cuando se lleva a cabo un caso de uso.

13. **Modelo de caso de uso**

    Es un modelo del sistema que describe lo que hace el sistema para cada tipo de usuario.

14. **Requisito funcional:**

- **Actor**
  Es una entidad que interactúa con el SI
- **Interactuar:**
  Actor genera datos para el SI
  SI genera información para el actor
- **Artefacto**
  Pieza de información tangible que es creada, modificada y usada por los trabajadores al realizar actividades, representa un área de responsabilidad. Puede ser un modelo, un elemento de un modelo o un documento.
- **Estereotipo**
  Es algo que identifica a un objeto

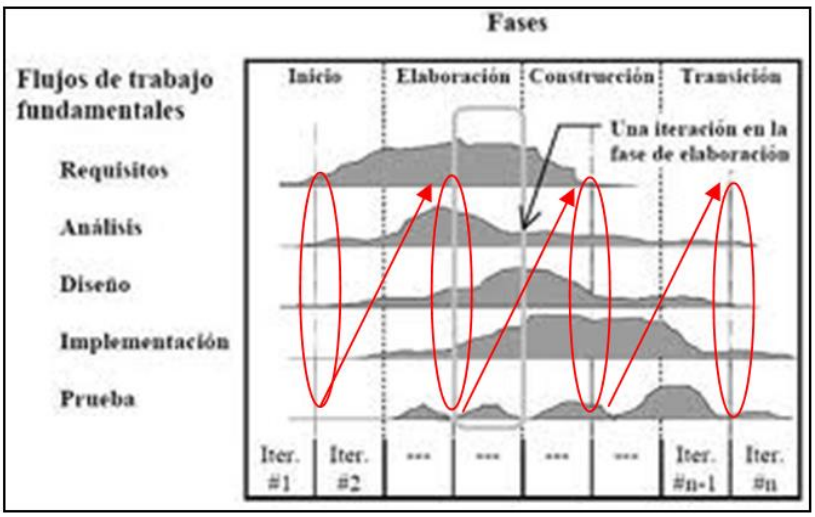
Aux. Eddy Escalante Ustariz

#73650500

PREGUNTAS CERO

1. **¿En un determinado ciclo del PUDS, por lo menos cuantas veces debería haber hecho: análisis, diseño e implementación?**

Por lo menos 4 veces y máximo n



2. **¿Cuál de las dos afirmaciones siguientes es correcta?**

- Tengo aun modelos, pero aun no tengo código
- Tengo código, pero aun no tengo modelos

R.- Ninguna de las afirmaciones es correcta!!

3. **¿Qué es una versión operativa?**

R.- es un software listo para entrar en producción.

4. **¿Qué es una versión beta?**

R.- es una versión de software que se genera en la fase de construcción y que no ha sido probada.

5. **¿Qué es una versión trivial?**

R.- es una versión de software que está completamente terminada y se le da al cliente para probarla.

6. **¿En la fase de la elaboración tengo modelos pero todavía no tengo códigos falso o verdadero?**

R.- falso porque el recorrido es extremadamente vertical

7. **¿Cuáles son las relaciones entre actores?**
    - Asociación
    - Dependencia
    - Generalización

    R.- Son las generalizaciones!!

8. **¿Cuáles son los tipos de Actores?**

R.-Como no nos interesa su comportamiento si no sus características y la forma de interactuar:

- Personas
- Organizaciones
- Otros Sistemas
- HW (lectores de huellas digitales, sensores)

9. **¿Hay flujos entre casos de uso?**

R.-Jamás hay flujos entre casos de uso, porque siempre intervienen un actor

10. **Elementos de SI basados en computadoras**

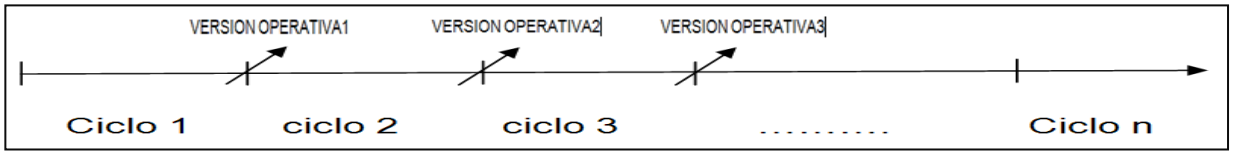
- ✓ **HW**: es todo el apoyo electrónico (escaners, fingers, router). La parte física RAM contiene los datos de la parte electrónica.
- ✓ **SW**: materia prima para el sistema, son los elementos de entrada.
- ✓ **Personas**: es la que da el estimulo al sistema, también requiere la información del sistema, son quienes interactúan con el sistema (no necesariamente **usuario**). Son quienes van a manejar el sistema.
- ✓ **Procesos**: Son las políticas del negocio definidas por las empresas.
- ✓ **Datos**: son las entradas necesarias que el sistema requiere para generar información.
- ✓ **Documentación**: es el respaldo del sistema, el respaldo de los métodos y es esencial.

La documentación puede ser diversa:

- ➢ como fue desarrollado, como funciona, como se actualiza, como se mantiene

PROCESO UNIFICADO DE DESARROLLO DE SOFTWARE (PUDS)

1. Ciclo de vida del PUDS



2. Características del PUDS

- **Dirigidos por Casos de Uso**
  - ✓ Los casos de uso inician los procesos de desarrollo, representan los requisitos funcionales, guían el proceso de desarrollo, guían su diseño, implementación y prueba.
  - ✓ Todos deben cumplir con la funcionalidad del sistema representada por los casos de uso.
  - ✓ Todo lo que hacemos se debe hacer en concordancia con los casos de uso que se gana? ¿Cual es el beneficio?
  - ✓ Nos garantiza el desarrollo de un software que cumpla todo los requerimientos del cliente, un software correcto que haga lo que se quiera que haga.

- **Centrado en la arquitectura**
  - ✓ Es una vista del diseño completo con las características más importantes resaltadas dejando los detalles de lado. La arquitectura es el esqueleto del sistema.
  - ✓ La arquitectura establece una estructura, se utiliza para preparar el software al cambio, facilita los cambios, hacer posibles los cambios, el esfuerzo e impacto sea menor.
  - ✓ Se consigue que el software sea robusto y viva más tiempo se garantiza que la flexibilidad del software sea escalable. Por ej. La arquitectura en 3 capas permite que sea flexible y abierto a los cambios.

- **Iterativo e incremental**
  - ✓ Dividir un proyecto en pequeñas partes o mini proyectos cada mini proyecto es una iteración que resulta un incremento, las iteraciones hacen referencia a pasos en el flujo de trabajo o crecimiento del producto.
  - ✓ Este proceso en cada iteración o fase de desarrollo se incrementa tanto en:
  - ✓ Incremento funcional: La versión operativa 2 por ejemplo tiene más funciones que la versión operativa 1
  - ✓ Incremento no funcional: se puede mejorar las estructuras de datos.
  - ✓ Un incremento debe ser visible.

3. Flujo de trabajo captura de requisitos

- **Actividades**
  - **Encontrar actores y casos de uso**
    - **Propósito** delimitar el sistema de su entorno esbozar quienes y que actores actúan en el sistema
    - **Resultado** glosario de términos

- **Priorizar casos de uso**
  - **Propósito** el propósito es priorizar los casos de uso más importantes para abordar en las primeras iteraciones
  - **Resultado** la vista de la arquitectura de modelos de casos de uso
- **Detallar un caso de uso**
  - **Propósito** describir el flujo de sucesos en detalle incluyendo como comienza termina e interactúa con los actores.
  - **Resultado** es el caso de uso detallado
- **Estructurar el modelo de casos de uso**
  - **Propósito** es hacer el diseño de las relaciones de los casos de uso con los actores, organizar los casos de uso sueltos en un solo diagrama y solo se tienen 2 elementos actores y casos de uso.
    Se tiene 3 relaciones asociación, relación y dependencia
    Entre actores solo se puede hacer generalización
    Entre casos de uso solo se puede hacer dependencia y asociación
  - **Resultado** se tiene como resultado el modelo de caso de uso estructurado
- **Esbozar Interfaz de usuario**
  - **Propósito** se utiliza simplemente con el propósito de validar los requisitos de software
    Los casos de usos tienen 2 propósitos para ingresar datos y para generar reportes
  - **Resultado** se obtiene como resultado el prototipo de interfaz de usuario lógico y físico

4. ¿Cuándo un software está muerto?

R.- Cuando no tiene una buena arquitectura y cuando al hacerse mantenimiento demanda mayor tiempo y esfuerzo.

Aux. Eddy Escalante Ustariz

#73650500

LENGUAJE UNIFICADO DE MODELADO-UML

Uml es un lenguaje de comunicación para modelar sw orientado a objeto, para comunicar entre ingeniero de sw, es un lenguaje unificado por la necesidad de ESTANDARIZAR, es un lenguaje para representar resultados

Es un leguaje para visualizar, especificar, construir y documentar sistemas de SW

1. **Características del UML**
   - ✓ **Visualizar:** gracias a uml podemos visualizar los artefactos del SW dado que un sistema de información es abstracta e intangible en su totalidad, por ese motivo surge la necesidad de representar sus partes, comportamientos, arquitectura, ed, etc.
   - ✓ **Especificar:** Todas las partes de un sw pueden ser especificadas con uml, podemos detallarlo sin ambigüedades
   - ✓ **Construir:** Cuando creamos modelos hacemos SW, no solo cuando escribimos código
   - ✓ **Documentar:** Respaldamos con documentos.

2. **¿Razones por las cuales se debería modelar (ventajas)?**

- Para entender con claridad lo que se quiere desarrollar
- Facilita la comunicación
- Dar seguridad en la implementación
- Reduce riesgos y costos de implementación

3. **¿En qué etapa se modela y cuáles son las ventajas de modelar?**

- En la etapa de análisis; los requisitos descubiertos son correcto
- En la etapa de diseño; si la solución que describí es apropiada

4. **¿Cuál sería el impacto negativo de desarrollar software sin modelar?**

Si no se tiene un modelo probablemente no es lo que se quería y habrá que deshacerlo.

5. **¿Cuáles son los bloques de construcción de UML?**

Son elementos, relaciones y diagramas

6. **¿Qué son los elementos?**

Son abstracciones de cosas reales o cosas ficticias

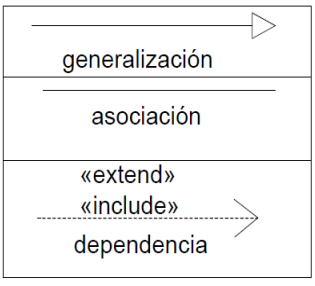
7. **¿Qué son las relaciones?**

Son las que relacionan a los elementos entre si

8. **¿Qué son los diagramas?**

Son colecciones de elementos con sus relaciones

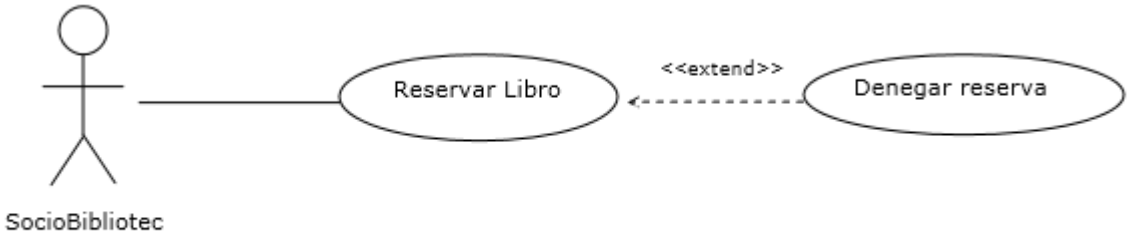
9. **¿Tipos de relaciones de UML?**



10. Relación de extensión

Las extensiones tienen las siguientes características:

- Representan una parte de la funcionalidad del caso que no siempre ocurre.
- Son un caso de uso en sí mismas.
- No necesariamente provienen de un error o excepción.
- Una extensión es un caso de uso en sí mismo, mientras que una alternativa no.
- Una alternativa es un error o excepción, mientras que una extensión puede no serlo.

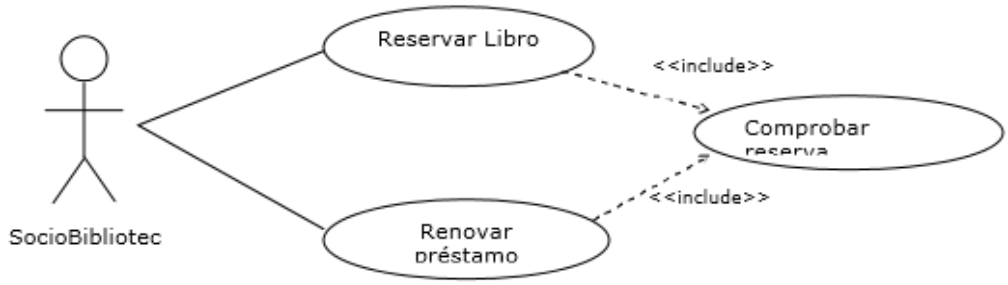


La flecha en el caso de las relaciones "extend" va hacia el caso de uso "original".

11. Relación include

Include: Se puede incluir una relación entre dos casos de uso de tipo "include" si se desea especificar comportamiento común en dos o más casos de uso.

12.



En la imagen anterior tanto "Reservar Libro" como "Renovar préstamo" hacen algo en común "Comprobar reserva".

13. ¿Cómo se clasifican los diagramas?

- **Diagramas de estructura**
  - ✓ Diagramas de clase
  - ✓ Diagrama de componente
  - ✓ Diagrama de objeto
  - ✓ Diagrama de estructura compuesta
  - ✓ Diagrama de despliegue

  - ✓ Diagrama de paquetes
- **Diagramas de comportamiento**
  - ✓ Diagrama de actividad
  - ✓ Diagrama de casos de uso
  - ✓ Diagrama de estado
- **Diagramas de interacción**
  - ✓ Diagrama de secuencia
  - ✓ Diagrama de comunicación
  - ✓ Diagrama de tiempo
  - ✓ Diagrama de interacción

14. ¿Cuál es el propósito del vocabulario UML?

El vocabulario y las normas del UML indican cómo crear y leer modelos bien formados gramaticalmente, pero no dicen que modelos deben crearse ni cuando hacerlo

15. ¿Qué es un artefacto?

Es una información que es utilizada o producida mediante un proceso de desarrollo de software. Los artefactos de UML se especifican en forma de diagrama.

Aux. Eddy Escalante Ustariz

#73650500

PROCESO DE SOFTWARE

1. ¿Qué es un proceso de software?
Realizar un conjunto de actividades que se deben llevar a cabo para desarrollar un software es decir obtener un producto como ser análisis diseño, implementación y prueba.

2. ¿Cuál es el propósito del análisis?
Es el propósito del análisis es lograr descubrir, describir especificar lo que el software debe realizar de manera detallada.

3. ¿Cuál es el resultado o producto que se debe realizar en el análisis?
Es el resultado es un documento denominado ERS (especificación de requisitos de software).

4. ¿Qué es la especificación de requisitos de software?
Es una descripción de manera completa detallada de cada uno de los requisitos de software sin ambigüedad acerca de los que el software debe hacer.

5. ¿Tareas o actividades que se deben llevar a cabo en el análisis?
Definir una metodología por ejemplo PUDS según el proceso unificado las actividades son 9 relacionadas con los flujos de trabajo

- **Captura de requisitos**
    - Encontrar actores y casos uso
    - Priorizar casos de uso
    - Detallar casos de uso
    - Esbozar interfaz usuario
    - Estructurar modelos de casos de uso

- **Análisis**
    - Análisis de la arquitectura
    - Análisis de un caso de uso
    - Analizar una clase
    - Analizar un paquete

6. ¿Trabajadores e involucrados de un proceso de análisis?
Arquitecto, analista de casos de uso, ingenieros de casos de uso

7. ¿Problemas en un sistema (errores)?

- Que el software no haga lo que debería hacer
- De nada sirve un gran programador sino tiene la capacidad de analizar lo que el software debe hacer.

8. **¿Qué es un modelo inicial?**
   Es un modelo base que representa el dominio del problema sobre el cual se va a trabajar. No representa nada de manera directa con el sw. Representa el alcance hacer acerca de donde él se va aplicar. Puede ser un modelo de dominio (diagrama de clases) o modelo de negocio (diagrama de actividades).

9. **Modelo de negocio**

Es una técnica para comprender los procesos de negocio de la organización. Describe los procesos del negocio de una empresa en términos de caso de uso del negoció y actores de negocio.

¿Cuando se lo hace?
Cuando hay que desarrollar un software de gestión.
¿Dónde se lo hace?
Se lo realiza antes de los requerimientos, antes del flujo de análisis, antes de todo.
¿Cómo se hace?
Utilizando el diagrama de actividades organizado en calles.

10. **Modelo de dominio**

El objetivo del modelado de dominio es comprender y describir las clases más importantes dentro del contexto del sistema.

¿Cuándo se hace?
Cuando hay que hacer un software de gestión donde no haya muchos actores
¿Dónde se hace?
En el momento de querer comprender el problema, las clases son conceptos de lo que estamos modelando, no son tablas ni archivos.
¿Cómo se hace?
Utilizando diagramas de clases.
Ambos modelos sirven para entender las políticas de negocio de una empresa, es decir el contexto del sistema.

11. **¿Qué es un diagrama de caso de uso?**
    Muestra lo que se supone lo que el sw debería hacer.

12. **¿Qué es un diagrama de actividades?**
    Modela las políticas de negocio de una determinada empresa. Toda política de negocio tiene un conjunto de negocio.

Aux. Eddy Escalante Ustariz

#73650500

PROCESO UNIFICADO DE DESARROLLO DE SOFTWARE (PUDS) –PARTE II

FLUJO DE TRABAJO DE ANALISIS			
PROPOSITO: Poder llegar a obtener una descripción completa y detallada del uso del sw.			
RESULTADO			
TRABAJADORES	**RESPONSABILIDADES (RESULTADO)**	**ACTIVIDADES**	**PROPOSITO**
Arquitecto	Modelo de análisis Descripción de la arquitectura	Análisis de la arquitectura	Organizar el sw, aplicar la base “divide y venceras” separando por temas.
Ing. Caso de uso	Realización de CU-análisis	Análisis de caso de uso	Describir de manera detallada que debería hacerse en el CU. Se usan los siguientes diagramas: tiempo, comunicación, actividad, de secuencia, de estado Realización de caso de uso
Ing. De componente	Clase y paquete de análisis	Análisis de clases	Identificar y mantener las responsabilidades de la clase. Identificar los atributos y relaciones de la clase Captura requisitos especiales sobre la realización de clase
		Análisis de paquete	Garantizar que el paquete sea lo más independiente posible de otros paquetes. Garantizar que el paquete cumple con sus objetivos. Describir las dependencias para poder estimar el efecto de cambios futuros.

FLUJO DE TRABAJO DE DISEÑO			
PROPOSITO: Formular los modelos que se centrar en requisitos no funcionales. El diseño es una etapa de soluciones y tiene dos sentidos uno hacia adelante y otro hacia atrás.			
RESULTADO: El producto del diseñado es la descripción del diseño de sw (DDS)			
TRABAJADORES	**RESPONSABILIDADES (RESULTADO)**	**ACTIVIDADES**	**PROPOSITO**
Arquitecto	Modelo de Diseño Despliegue Descripción Arquitectura (Vista Modelo Diseño y Despliegue)	Diseño de la arquitectura	Lograr que el sw sea flexible. El diseñador realiza la arquitectura desde dos puntos: el grado de acoplamiento (-) y el grado de cohesión (+).
Ing. Caso de uso	Realización de CU-Diseño	Diseño de un caso de uso	Comprende las etapas de: a. Identificar clases del diseño participantes b. Descripción de las interacciones entre objetos del diseño c. Identificar subsistemas e interfaces participantes d. Descripción de interacciones entre subsistemas e. Captura de requisitos de implementación del Caso de uso
Ing. De componente	Clases, Subsistema e Interfaz de Diseño	Diseño de una clase	Definir para una clase sus operaciones, atributos, relaciones, métodos (que son realizados por las operaciones), ciclo de vida (máquina de estados), dependencias, requisitos relevantes a su implementación, interfaces, etc.
		Diseño de un Subsistema	Mantenimiento de las dependencias entre subsistemas Mantenimiento de interfaces proporcionadas por el subsistema Mantenimiento de los contenidos de los subsistemas

FLUJO DE TRABAJO DE IMPLEMENTACION			
PROPOSITO: Transforma en código maquina lo diseñado. También asume la redacción de los manuales de usuario y todos los demás artefactos que se definan como parte del sistema en explotación más que del proyecto.			
RESULTADO:			
TRABAJADORES	**RESPONSABILIDADES (RESULTADO)**	**ACTIVIDADES**	**PROPOSITO**
Arquitecto	Modelo de Implementación Despliegue; Descripción Arquitectura	Implementación de la Arquitectura	Esbozar el modelo de implementación y su arquitectura mediante: a. Identificación de componentes arquitectónicamente significativos tales como Componentes ejecutables. b. La asignación de componentes a los nodos en las configuraciones de redes relevantes. Esto se hace asignando un componente ejecutable a cada clase activa
Integrador de sistema	Plan de Integración	Integrar el sistema	Crear un plan de integración de construcciones Integrar cada construcción antes de que sea sometida a pruebas de integración
Ingeniero de componentes	Componente, Interfaz, Implementacion Subsistema	Implementar un subsistema	Para asegurar que un subsistema cumpla su papel en cada construcción.
		Implementar una clase	Implementar una clase de diseño en un componente fichero.
		Realizar prueba de unidad:	Realizar la prueba de unidad es probar los componentes implementados como unidades individuales.

FLUJO DE TRABAJO DE PRUEBA			
PROPOSITO: Las pruebas tienen por objetivo encontrar defectos en el sw. Hay dos tipos de pruebas: de caja negra y de caja blanca			
RESULTADO:			
TRABAJADORES	**RESPONSABILIDADES (RESULTADO)**	**ACTIVIDADES**	**PROPOSITO**
Diseñador de pruebas	Modelo de pruebas Casos de prueba, procedimiento de pruebas , evaluación de pruebas y plan de pruebas	Planificar prueba	Consiste en describir una estrategia de la prueba, estimar requisitos para la prueba, recursos humanos y sistemas necesarios; y planificar el esfuerzo de la prueba
Ing. De componente	Componente de pruebas	Diseñar prueba	Consiste en identificar y describir los casos de prueba para cada construcción, para luego identificar y estructurar los procedimientos de prueba especificando como realizar los casos de prueba.
Ing. Pruebas de integración	Defecto	Implementar prueba	Consiste en automatizar los procedimientos de prueba creando componentes de prueba si esto es posible.
		Realizar pruebas de integración	Realizar las pruebas de integración relevantes ejecutando los procedimientos o componentes de prueba correspondientes. Comparar los resultados de las pruebas con los resultados esperados e investigar resultados no esperados. Informar defectos a los ingenieros de componentes las fallas encontradas. Informar los defectos a los diseñadores de pruebas, quienes usarán los defectos para evaluar los resultados de las pruebas.
Ing. Pruebas sistema	Defecto	Realizar pruebas de sistema	Una vez finalizadas las pruebas de integración se realizan las pruebas de sistema de forma similar.

		Evaluar la prueba	Se comparan resultados de la prueba con resultados esperados. Para esto se utilizan métricas: - Compleción de la prueba –Fiabilidad

FASES DEL PROCESO UNIFICADO DE DESARROLLO DE SOFTWARE	
FASE	**DESCRIPCION**
INICIO	Se desarrolla una descripción del producto final a partir de una buena idea y se presenta el análisis de negocio para el producto
ELABORACION	Se especifican en detalle la mayoría de los casos de uso del producto y se diseña la arquitectura del sistema. El resultado de esta fase es una línea base de la arquitectura, y la disposición del director de planificar las actividades y estimar los recursos necesarios para terminar el proyecto
CONSTRUCCION	Se crea el producto. Aquí la línea base de la arquitectura crece hasta convertirse en el sistema completo. Al final de esta fase, el producto tiene todos los casos de uso que la dirección y el cliente han acordado para el desarrollo de esta versión. Sin embargo, puede tener defectos
TRANSICION	Cubre el periodo comprendido durante el cual el producto se convierte en versión beta. Esta fase corrige errores antes de la entrega. El equipo de mantenimiento divide los errores en 2 categorías: los que tienen suficiente impacto en la operación para justificar una versión incrementada y los que pueden corregirse en la siguiente versión normal.

RESUMEN DE PRACTICO SOBRE DIAGRAMAS		
DIAGRAMA	**DESCRIPCION**	**NOTACION**
SECUENCIA	El diagrama de secuencia es un tipo de diagrama de interacción que muestra el orden de las interacciones entre las partes del sistema. Se compone de una colección de participantes, es decir, las partes del sistema (normalmente, objetos) que interactúan entre sí durante la secuencia, mediante el envío de mensajes	**FRAGMENTO COMBINADO**(contenedor donde se colocan fragmentos o porciones de una interacción) **OPERADORES DE INTERACCION** (Ref, Assert, Loop, Break, Alt, Opt, Neg, Par, Critical). **MENSAJES DE CREACION Y DESTRUCCION** (Para mostrar la creación de un participante se envía un mensaje síncrono <<create>> a la línea de vida del participante o se dibuja el rectángulo correspondiente en el punto donde se ha creado sobre el eje vertical de tiempo. Cuando un participante se destruye, se marca con una X mayúscula y se termina su línea de vida en el punto en que se destruyó. La marca de destrucción puede ir precedida de un mensaje síncrono <<destroy>> enviadoal participante.**)** **MENSAJE SÍNCRONO**(Sólo desencadena una operación cuando el destinatario acepta el msje, y el emisor del msje se bloquea hasta ese momento). **MENSAJE ASÍNCRONO**(Representa un envío de msje donde no hay retorno explícito y no se interrumpe la ejecución del emisor) **MENSAJE DE RETORNO**(Es una representación

		opcional para mostrar explícitamente que el flujo de control vuelve al emisor original del msje)
ESTADO	Los diagramas de estado capturan los ciclos de vida de los objetos, subsistemas y sistemas. Los principales conceptos de un diagrama de estado son los eventos y los estados. Muestran cómo reaccionan los objetos a los eventos y cómo cambian su estado interno. Se ejecutan en concurrencia y pueden cambiar de estado independientemente.	**ESTADO**(Es una condición o situación en la vida de un objeto durante la cual satisface alguna condición , realiza alguna actividad o espera algún evento, rectángulo de esquinas redondeadas) **EVENTOS**(Un evento es algo que ocurre en un momento del tiempo, no tiene duración.) **TRANSICIONES**(cambio de estado causado por un evento, representado por una flecha) **PSEUDOESTADO INICIAL**(se representa con un círculo sólido, que puede etiquetarse para indicar diferentes condiciones iniciales) **PSEUDOESTADO FINAL (**se representa con una diana, que puede etiquetarse para distinguir condiciones finales)
DESPLIEGUE	Los diagramas de despliegue muestran las relaciones físicas entre los componentes físicos y lógicos del sistema final(HW y SW)	
TIEMPO	Es una representación especial de Interacción enfocada en especificar los tiempos de envío de mensajes entre los participantes en la interacción. Su principal propósito es mostrar el cambio de estado de un participante a lo largo del tiempo, en respuesta a los eventos aceptados. El diagrama de tiempo se lee de izquierda a derecha. El participante se muestra a la izquierda del diagrama y, a continuación, se listan sus posibles estados, seguidos por una representación gráfica de las transiciones entre los estados.