

Solución de la guía taller de la A a la N

Grupo #6 :

Camilo Andrés Parra Cuenca

Daniela Alexandra Herrera Aponte

Edwin Eleider Amaya

Universidad Antonio Nariño

Gestión de requisitos de software

14/03/2025

CONTEXTO

LA ESPECIFICACIÓN DE UN PROBLEMA:

- a. Resuma el ciclo de vida de construcción de un programa.
- b. Explique los aspectos que hacen parte del análisis de un problema.
- c. Explique las etapas del proceso de solución de problemas.
- d. ¿Cuáles son los elementos que se deben entregar a un cliente?

SOLUCION

a. Resume el ciclo de vida de construcción de un programa.
El ciclo de vida en desarrollo de software consta de varias fases que permiten la construcción eficiente de un programa:

Análisis de requisitos: Se identifican y documentan las necesidades del usuario y las funcionalidades esperadas.

Diseño: Se define la arquitectura del software, estructura de datos y la interfaz de usuario.

Implementación: Se traduce el diseño en código utilizando un lenguaje de programación.

Pruebas: Se verifican errores y se asegura que el programa cumpla con los requisitos.

Despliegue: Se instala el software en el entorno del usuario final.

Mantenimiento y evolución: Se realizan actualizaciones y correcciones según sea necesario.

b. Aspectos que hacen parte del análisis de un problema.
El análisis de un problema en el desarrollo de software implica:

- Definición clara del problema: se debe identificar con precisión lo que se necesita resolver.

Identificación de requisitos: Se distinguen requisitos funcionales (que debe hacer el software) y no funcionales.

(rendimiento, seguridad y usabilidad.)

Identificación de restricciones: Se consideran limitaciones de tiempo, tecnología, presupuesto, etc.

Estudio de casos de uso: Se analizan los escenarios en los que se utilizará el software.

Factibilidad técnica y económica: Se determina si es viable desarrollar la solución con los recursos disponibles.

c. Etapas del proceso de solución de problemas.

Para resolver un problema en software, se siguen estos pasos:

- ① Comprensión del problema: Se analiza la situación y se identifican los requisitos.
- ② Diseño de la solución: Se elabora un plan o algoritmo para resolver un problema.
- ③ Implementación: Se programa la solución en un lenguaje adecuado.
- ④ Pruebas y depuración: Se ejecuta el software para detectar y corregir errores.
- ⑤ Optimización: Se mejora la eficiencia y desempeño del software.
- e. Documentación y mantenimiento: Se registran los detalles del sistema para futuras modificaciones.

d. Elementos que se deben entregar a un cliente.
Cuando se desarrolla software para un cliente se debe entregar.

Código fuente (si está acordado en el contrato).

Ejecutable de software listo para su uso

Manuales de usuario para facilitar operación y mantenimiento

Casos de prueba y documentación de pruebas para verificar el correcto funcionamiento.

Documentación del diseño y arquitectura para modificaciones

Licencias y acuerdos sobre el uso y distribución del software

Soporte técnico y mantenimiento, si está incluido en el contrato

EXPERIENCIA

e. Elabore la Tarea No. 1 (pág. 5 del texto guía), con el objetivo de identificar los aspectos que forma parte de un problema.

①

Ejemplo 8



Cliente

Usuario

Requerimiento funcional

Mundo del problema

Requerimiento no funcional

Objetivo: Identificar los aspectos que hacen parte de un problema.

El problema: una empresa de aviación quiere construir un programa que le permita buscar una ruta para ir de una ciudad a otra, usando únicamente los vuelos de los que dispone la empresa. Se quiere utilizar este programa desde todas las agencias de viaje del país.

La empresa de aviación.

Las agencias de viaje del país.

RT: dadas dos ciudades C1 y C2, el programa debe dar el itinerario para ir de C1 a C2, usando los vuelos de la empresa. En este ejemplo sólo hay un requerimiento funcional explícito. Sin embargo, lo usual es que en un problema haya varios de ellos.

En el enunciado no está explícito, pero para poder resolver el problema, es necesario conocer todos los vuelos de la empresa y la lista de ciudades a las cuales va. De cada vuelo es necesario tener la ciudad de la que parte, la ciudad a la que llega, la hora de salida y la duración del vuelo. Aquí debe ir todo el conocimiento que tenga la empresa que pueda ser necesario para resolver los requerimientos funcionales.

El único requerimiento no funcional mencionado en el enunciado es el de distribución, ya que las agencias de viaje están geográficamente dispersas y se debe tener en cuenta esta característica al momento de construir el programa.

Tarea 1



Cliente

Usuario

Mundo del problema
Requerimiento funcional

Requerimiento funcional
Mundo del problema

Requerimiento no funcional

Objetivo: Identificar los aspectos que forman parte de un problema.

El problema: un banco quiere crear un programa para manejar sus cajeros automáticos. Dicho programa sólo debe permitir retirar dinero y consultar el saldo de una cuenta.

Identifique y discuta los aspectos que constituyen el problema. Si el enunciado no es explícito con respecto a algún punto, intente imaginar la manera de completarlo.

El banco

Clientes del banco que utilizan el cajero.

El banco necesita un sistema confiable para operar los cajeros automáticos, asegurando que las transacciones sean seguras y eficientes.

El Programa debe permitir a los clientes retirar dinero y consultar el saldo de la Cuenta.

El Sistema debe ser seguro, rápido y estable para garantizar la protección de los datos de los clientes y evitar fraudes o accesos no autorizados.

f. Elabore la Tarea No. 2 (pág. 13), con el objetivo de identificar los requerimientos funcionales de un problema.

2



En el CD que acompaña el libro, puede encontrar el formulario que debe llenar un programador para especificar los requerimientos funcionales de un problema.

Tarea 2



Objetivo: Crear habilidad en la identificación y especificación de requerimientos funcionales. Para el caso de estudio 2, un simulador bancario, identifique y especifique tres requerimientos funcionales.

Requerimiento funcional 1	Nombre	Simulación del avance del tiempo.
	Resumen	El sistema debe permitir al usuario avanzar mes a mes para visualizar la evolución de su saldo y inversiones.
	Entradas	Acción del usuario sobre el botón de avanzar "»»"
Requerimiento funcional 2	Resultado	Se actualizan los saldos de las cuentas de ahorro y los CDT con los intereses generados, mostrando el nuevo saldo total.
	Nombre	Operaciones en cuenta corriente y ahorro.
	Resumen	El sistema debe permitir al usuario consignar o retirar dinero de su cuenta de ahorros y corrientes.
Requerimiento funcional 3	Entradas	Monto a consignar o retirar de las cuentas seleccionadas.
	Resultado	Se actualiza el saldo de las cuentas correspondientes reflejando la transacción realizada.
	Nombre	Apertura y cierre de CDT.
Requerimiento funcional 3	Resumen	El sistema debe permitir al usuario abrir un CDT definiendo el monto de inversión y cerrarlo cuando sea necesario.
	Entradas	Monto de inversión al abrir un CDT y acción de cierre del usuario.
	Resultado	Al abrir, el dinero se transfiere desde la cuenta corriente al CDT. Al cerrar, el saldo más los intereses acumulados pasan a la cuenta corriente.

g. Elabore la Tarea No. 3 (pág. 14), con el objetivo de identificar los requerimientos funcionales de un problema.



Tarea 3

Objetivo: Crear habilidad en la identificación y especificación de requerimientos funcionales. Para el caso de estudio 3, un programa para manejar un triángulo, identifique y especifique tres requerimientos funcionales.

Requerimiento funcional 1	Nombre	Dibujo del triángulo.
	Resumen	El sistema debe permitir visualizar un triángulo en la pantalla, definido por tres puntos con coordenadas (X, Y).
	Entradas	Coordenadas de los tres puntos del triángulo
	Resultado	Se muestra el triángulo en la pantalla con sus características definidas.
Requerimiento funcional 2	Nombre	Cálculo del Perímetro, área y altura.
	Resumen	El sistema debe calcular y mostrar el perímetro, el área y la altura del triángulo en función de sus coordenadas.
	Entradas	Coordenadas de los vértices del triángulo.
	Resultado	Se presentan los valores del perímetro, área y altura del triángulo.
Requerimiento funcional 3	Nombre	Modificación de puntos y colores
	Resumen	El sistema debe permitir cambiar los puntos del triángulo y modificar el color de las líneas y fondo.
	Entradas	Nuevas coordenadas de los puntos y valores RGB del color de las líneas y el fondo.
	Resultado	Se actualiza la visualización del triángulo con los cambios aplicados.

h. Elabore la Tarea No. 4 (pág. 17), con el objetivo de identificar las entidades del mundo del problema.



Tarea 4 Objetivo: Identificar las entidades del mundo para el caso de estudio 3: un programa que maneje un triángulo.
Lee el enunciado del caso y trate de guiarse por los sustantivos para identificar las entidades del mundo del problema.

Entidad	Triángulo	figura geométrica de 3 lados por 3 vértices. Puntos con coordenadas (x, y) un color de borde y un color de relleno.
Entidad	Punto.	Cada vértice del triángulo está representado por un punto con coordenadas en el espacio (x, y) .
Entidad	Color	Propiedad del triángulo que define su apariencia visual, expresada en valores RGB (Red, Green, Blue).

Punto de reflexión: ¿Qué pasa si no identificamos bien las entidades del mundo? Si no identificamos correctamente las entidades del mundo, el diseño del sistema será confuso e insuficiente. Podríamos representar datos importantes como simples atributos en lugar de objetos independientes, lo que dificultaría la manipulación del código triángulo y sus propiedades.

Punto de reflexión: ¿Cómo decidir si se trata efectivamente de una entidad y no sólo de una característica de una entidad ya identificada? Para determinar si algo es una entidad y no sólo un atributo, debemos preguntarnos si dicho elemento puede existir de forma independiente y tener múltiples propiedades propias. Si un elemento puede ser modelado con características propias y es necesario para la funcionalidad del sistema, entonces es una entidad.

5.2.2. Modelar las Características

Una vez que se han identificado las entidades del mundo del problema, el siguiente paso es identificar y modelar sus características. A cada característica que vayamos encontrando, le debemos asociar (1) un nombre significativo y (2) una descripción del conjunto de valores que dicha característica puede tomar.

En programación orientada a objetos, las características se denominan atributos y, al igual que las clases, serán elementos fundamentales tanto en el diseño como

en la implementación. El nombre de un atributo debe ser una cadena de caracteres no vacía, que empiece con una letra y que no contenga espacios en blanco.



Por convención, el nombre de los atributos comienza por una letra minúscula. Si es un nombre compuesto, se debe iniciar cada palabra simple con mayúscula.

- Elabore la Tarea No. 5 (pág. 20), con el objetivo de identificar las entidades de un caso de estudio.

5

20

Fundamentos de Programación, Alan F.



Tarea 5
Objetivo: Identificar las características de las entidades del caso de estudio 2, un simulador bancario.
 Para cada una de las cinco entidades identificadas en el caso de estudio del simulador bancario, identifique los atributos, sus valores posibles, y escriba la clase en UML. No incluya las relaciones que puedan existir entre las clases, ya que eso lo haremos en la siguiente etapa del análisis. Por ahora trate de identificar las características de las entidades que son importantes para los requerimientos funcionales.

Clase: CuentaAhorro

Atributo	Valores posibles	Diagrama UML
NombreCliente	Cadena de caracteres (String)	<pre> classDiagram class CuentaAhorro { +Nombre Cliente : String +CPolvo Cliente : int +Saldo Total : float } </pre>
CedulaCliente	Numero entero (int)	
SaldoTotal	Numero decimal (float)	

Clase: CuentaCorriente

Atributo	Valores posibles	Diagrama UML
Saldo	Numero decimal (float)	<pre> classDiagram class CuentaCorriente { +Saldo : float +Numero Cuenta : int } </pre>
NumeroCuenta	Numero entero (int)	

Clase: CuentaAhorros

Atributo	Valores posibles	Diagrama UML
Saldo	Numero decimal (float)	<pre> classDiagram class CuentaAhorros { +Saldo : float +Interes Mensual : float +Numero Cuenta : int } </pre>
InteresMensual	0.6 (float, fijo)	
NumeroCuenta	Numero entero (int)	

Clase: CDT

Atributo	Valores posibles	Diagrama UML
Saldo Invertido	Numero decimal (float)	<pre> classDiagram class CDT { +Saldo Invertido : float +Interes Mensual : float +Plazo Meses : int } </pre>
InteresMensual	Numero decimal (float, variable)	
PlazoMeses	Numero entero (int)	

j. Elabore la Tarea No. 6 (pág. 23), con el objetivo de reflexionar sobre el nivel de precisión de un algoritmo.

7

Utilice este espacio para anotar sus conclusiones: El algoritmo presenta ambigüedades y omite detalles clave. No explica cómo verificar el tiempo, qué hacer si la estación está cerrada, ni cómo elegir la dirección correcta del metro. Tampoco menciona cómo hacer transbordos o qué hacer en caso de emergencia. Además, asume que el usuario sabe leer mapas y señales, lo cual puede no ser cierto. Para mejorarlo, se deben agregar instrucciones más detalladas, opciones ante imprevistos y aclaraciones.

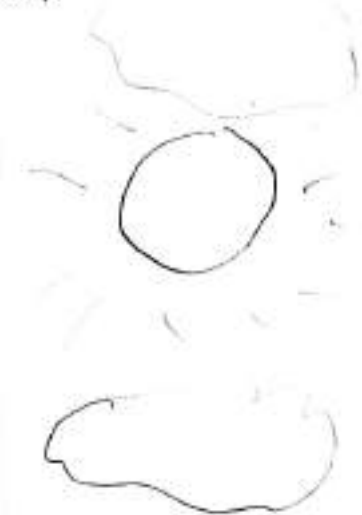
Tarea 7



Objetivo: Entender la complejidad que tiene la tarea de escribir un algoritmo.

Esta tarea es para ser desarrollada en parejas: (1) en el primer cuadrante haga un dibujo simple, (2) en el segundo cuadrante escriba las instrucciones para explicarle a la otra persona cómo hacer el dibujo, (3) lea las instrucciones a la otra persona, quien debe intentar seguirlas sin ninguna ayuda adicional, (4) compare el dibujo inicial y el dibujo resultante.

Dibujo:



Algoritmo:




ACCIÓN

TRABAJO PRELIMINAR: CASO DE ESTUDIO

Para los siguientes ejercicios:

- Dirígete al portal: <https://cupi2.virtual.uniandes.edu.co/nivel-1/ejemplos-n1>
- Selecciona uno de los ejemplos propuestos:
 - El empleado, El Triángulo
 - Simulador Bancario, Elecciones
- Ingresa a la opción 'Ir al ejemplo' del caso que decidan estudiar:



- Estudia los siguientes aspectos del ejemplo seleccionado: *Enunciado*, *Requerimientos funcionales (casos de uso)* y el *Modelo (clases del proyecto)*. A continuación, redacta el **enunciado del problema** y el **nombre cada uno de los requerimientos funcionales del proyecto**.
- Dibuja el respectivo **diagrama de casos de uso** del ejemplo elegido (consulta "requerimientos funcionales" en 'Archivo').
- Observa nuevamente el modelo conceptual del caso y escribe **el nombre de cada una las clases** identificando sus respectivas variables (atributos) y funciones:



K. Enunciado

Se requiere un programa en el que podamos simular el comportamiento de los productos de un cliente. El cual puede tener 3 productos básicos financieramente. En los cuales se van a manejar Cuenta de ahorros, corrientes y CDT, los 2 primeros se pueden dar cosas básicas como recibir consignaciones y realizar retiros. El CDT y es un contrato a un término específico por un valor, % de interés y en este caso pueden realizar retiros ni consignaciones parciales hasta la finalidad del tiempo establecido.

Requerimientos Funcionales (Casos de Uso)

1. Consultar saldo de la cuenta corriente

- Permite al cliente visualizar el saldo disponible en su cuenta cte.

2. Consultar el saldo de la cuenta de ahorros.

- Permite al cliente visualizar el saldo disponible en su cta de ahorros.

3. Consultar el saldo de un CDT.

- Permite al cliente visualizar el capital invertido y los intereses acumulados en su CDT.

4. Consultar el saldo total del Cliente.

- Muestra el saldo total sumando el saldo disponible de la cta cte, de la cta de ahorros y el CDT.

5. Invertir un CDT.

- Permite al cliente invertir una cantidad determinada de dinero, estableciendo un interés mensual acordado con el Banco.

6. Cerrar un CDT.

- Permite al cliente cerrar un CDT Activo, trasladando el capital y los intereses generados a la cta de.

7. Consignar dinero en la cuenta Corriente

- Permite al cliente consignar dinero en su cta de.

8. Retirar dinero de la cuenta corriente

- Permite al cliente retirar dinero de la cta de siempre y cuando tenga saldo disponible.

9. Consignar dinero en la cuenta de ahorros

- Permite al cliente consignar dinero en su cta de ahorros.

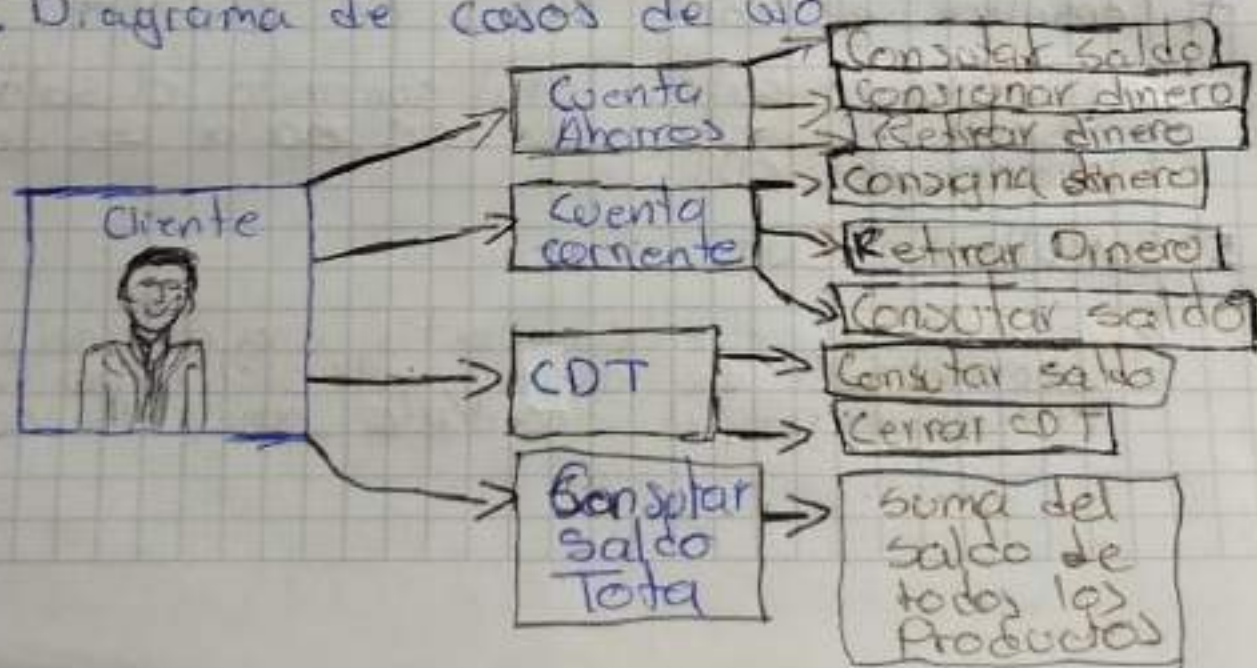
10. Retirar dinero de su cuenta de ahorros

- Permite al cliente retirar dinero de la cta de ahorros siempre y cuando tenga saldo disponible.

11. Avanzar un mes en la simulación

- Simula el paso del tiempo aplicando los intereses a la cta de ahorros y a los CDT activos.

L. Diagrama de casos de uso



M. Escribe el nombre de cada una de las clases identificando sus variables (Atributos) y Funciones:

1. Clase: Simulador Bancario

Atributos

- Cédula : Int.
- Nombre : String
- mesActual : Int

Metodos

- SimuladorBancario (cedula: int, nombre: string)
- darNombre(): String
- darCedula(): int
- darCuentaCorriente(): cuentaCorriente
- darCuentaAhorros(): cuentaAhorros
- darCDT(): CDT
- darMesActual(): Int
- calcularSaldoTotal(): Double
- invertirCDT (pMontos: double): Void
- retirarCuentaCorriente (pMontos: double): Void
- consignarCuentaCorriente (pMontos: double): Void
- retirarCuentaAhorros (pMontos: double): Void
- consignarCuentaAhorros (pMontos: double): Void
- avanzarExSimulacion(): Void
- cerrarCDT(): Void
- metodo1(): String
- metodo2(): String

2. Clase: CDT

Atributos

- valorInvertido: double
- interesMensual: double
- mesApertura: Int

Metodos:

- CDT()
- darInteresMensual(): double

- invertir (pMontoInvertido: double, pInteresMensual: double, pMes: int): void
- calcularValorPresente (pMesActual: int) double
- cerrar (pMesActual: int) double

3. Clase: CuentaCorriente

Atributos:

- Saldo: double

Métodos:

- cuentaCorriente ()
- darSaldo (): double
- consignarMonto (pMonto: double): void
- retirarMonto (pMonto: double): void

4. Clase: CuentaAhorros

Atributos:

- Saldo: double
- InteresMensual: double

Métodos:

- CuentaAhorros ()
- darSaldo (): double
- darInteresMensual (): double
- consignarMonto (pMonto: double): void
- retirarMonto (pMonto: double): void
- actualizarSaldoPorCadaMes (): void

n. Debes plantear 2 ideas de proyecto (problemas solubles y algorítmicos):

1: Simulador de Prestamos bancarios.

Requerimiento funcional 1	Nombre	Ingreso de datos del Prestamo.
	Resumen	El usuario debe ingresar el monto del Prestamo, la tasa de interes y el tiempo de pago.
	Entradas	Monto del Prestamo, tasa de interes, numero de meses.
	Resultado	Se almacena la información ingresada para calcular el Prestamo.
Requerimiento funcional 2	Nombre	Calculo de intereses y cuotas.
	Resumen	El sistema debe calcular el total a pagar, intereses y cuotas mensuales.
	Entradas	Datos del Prestamo.
	Resultado	Se muestra el costo total y las cuotas mensuales.
Requerimiento funcional 3	Nombre	Simulador con diferentes tasas de interés.
	Resumen	Permite al usuario comparar cuanto pagaría con diferentes tasas de interés.
	Entradas	Tasa de interes variable.
	Resultado	Se muestra una tabla comparativa con diferentes escenarios.
Requerimiento funcional 4	Nombre	Generación de Reporte.
	Resumen	El sistema debe generar un resumen del Prestamo en un archivo PDF.
	Entradas	Datos del Prestamo calculado.
	Resultado	Se genera un archivo con la información del Prestamo.

n. Debes plantear 2 ideas de proyecto (problemas solubles y algorítmicos):

Idea 2: Gestor de turnos de una clínica.

Requerimiento funcional 1	Nombre	Registro de pacientes.
	Resumen	Los pacientes deben registrarse en el sistema para el turno.
	Entradas	Nombre del paciente, cédula, motivo de consulta.
	Resultado	Se genera un turno en la lista de espera.
Requerimiento funcional 2	Nombre	Asignación de turnos según orden de llegada.
	Resumen	Los turnos deben asignarse en orden de llegada.
	Entradas	Lista de pacientes en espera.
	Resultado	Se notifica al paciente cuando es su turno.
Requerimiento funcional 3	Nombre	Notificación de turnos.
	Resumen	El sistema debe avisar cuando un paciente debe ir a su turno.
	Entradas	Lista de turnos activos.
	Resultado	Se muestra un aviso de pantalla o se envía un mensaje.
Requerimiento funcional 4	Nombre	Historial de pacientes atendidos.
	Resumen	Se debe registrar un historial de pacientes atendidos.
	Entradas	Datos del paciente y consulta realizada.
	Resultado	Se guarda en historial accesible para consultas futuras.