UNIVERSIDAD PERUANA DE CIENCIAS E INFORMÁTICA

FACULTAD DE CIENCIAS E INGENIERÍA

CARRERA PROFESIONAL DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

TRABAJO DE SUFICIENCIA PROFESIONAL

"Propuesta Metodológica para la Optimización del Desarrollo de Software Basada en Diagramas de Flujo"

AUTOR:

Bach. Rodriguez Morocho, Ronald Fernando

PARA OPTAR EL TITULO PROFESIONAL DE:

INGENIERO DE SISTEMAS E INFORMÁTICA

ASESOR:

Mg. Villanueva Dávila, Ronald Heber

ORCID: 0009-0003-7429-3985

DNI: 10305132

LIMA- PERÚ

2025

INFORME DE SIMILITUD

UPCI
CAMINO AL ÉXITO
UNIVERSIDAD PERUANA DE CIENCIAS E INFORMÁTICA

INFORME DE SIMILITUD

N°015-2025-UPCI-FCI-REHO-T

A : **MG. HERMOZA OCHANTE RUBEN EDGAR**
Decano (e) de la Facultad de Ciencias e Ingeniería

DE : **MG. HERMOZA OCHANTE, RUBEN EDGAR**
Docente Operador del Programa Turnitin

ASUNTO : Informe de evaluación de Similitud de Trabajo de Suficiencia Profesional: **BACHILLER RODRIGUEZ MOROCHO, RONALD FERNANDO**

FECHA : Lima, 30 de Junio de 2025.

Tengo el agrado de dirigirme a usted con la finalidad de informar lo siguiente:

1. Mediante el uso del programa informático **Turnitin** (con las configuraciones de excluir citas, excluir bibliografía y excluir oraciones con cadenas menores a 20 palabras) se ha analizado el Trabajo de Suficiencia Profesional titulada: **"PROPUESTA METODOLÓGICA PARA LA OPTIMIZACIÓN DEL DESARROLLO DE SOFTWARE BASADA EN DIAGRAMAS DE FLUJO"**, presentado por el Bachiller **RODRIGUEZ MOROCHO, RONALD FERNANDO.**
2. Los resultados de la evaluación concluyen que el Trabajo de Suficiencia Profesional en mención tiene un **ÍNDICE DE SIMILITUD DE 5%** (cumpliendo con el artículo 35 del Reglamento de Grado de Bachiller y Título Profesional UPCI aprobado con Resolución N° 373-2019-UPCI-R de fecha 22/08/2019).
3. Al término análisis, el Bachiller en mención **PUEDE CONTINUAR** su trámite ante la facultad, por lo que el resultado del análisis se adjunta para los efectos consiguientes

Es cuanto hago de conocimiento para los fines que se sirva determinar.

Atentamente,

MG. HERMOZA OCHANTE, RUBEN EDGAR
Universidad Peruana de Ciencias e Informática
Docente Operador del Programa Turnitin

Adjunto:

**Resultado de similitud*

DEDICATORIA

A Dios, por guiarnos cada paso que dimos en este viaje académico y darnos la fortaleza y entendimiento en este logro académico.

AGRADECIMIENTO

Agradecemos a esta institución por brindarnos la educación y las oportunidades que han moldeado nuestro futuro. Cada día en este campus ha sido una experiencia enriquecedora. Nuestro tiempo aquí ha sido un viaje de crecimiento y aprendizaje que siempre valoraremos.

DECLARACIÓN DE AUTORÍA

Nombres: Ronald Fernando

Apellidos: Rodriguez Morocho

Código: 1304000502

DNI: 41739081

Declaro que, soy el autor del trabajo realizado y que es la versión final que he entregado a la oficina del Decanato de la Facultad de Ingeniería de la Universidad Peruana de Ciencias e Informática.

Asimismo, declaro que he citado debidamente las palabras o ideas de otros autores, refiriendo expresamente el nombre de la obra y página o páginas que me sirvieron de fuente.

ÍNDICE

INTRODUCCIÓN

El avance acelerado de la tecnología ha modificado sustancialmente los enfoques empleados en el diseño, desarrollo e implementación de sistemas informáticos, haciendo indispensable la incorporación de estrategias que optimicen el ciclo de vida del software. En este contexto, el presente trabajo —"Propuesta metodológica para la optimización del desarrollo de software basada en diagramas de flujo"— plantea el uso sistemático de diagramas de flujo como herramienta para mejorar la eficiencia en las etapas de análisis, diseño, codificación y depuración.

Los diagramas de flujo permiten una representación clara y estructurada de la lógica de los algoritmos, facilitando tanto su comprensión como la detección temprana de errores lógicos y redundancias. Esto contribuye a una separación efectiva entre el diseño algorítmico y su posterior implementación en un lenguaje de programación, lo cual es clave para la producción de software robusto, mantenible y de alta calidad.

La investigación aborda la estandarización de los diagramas de flujo conforme a normativas técnicas internacionales, como la ISO 5808:1985 y la DIN 66001:1996, y analiza su aplicabilidad en la mejora de procesos en entornos de desarrollo software. Además, se plantea una metodología estructurada que contempla la definición de objetivos, un enfoque sistemático para su aplicación y una arquitectura de contenido orientada a la validación del modelo propuesto.

Se espera que los resultados de este estudio proporcionen una base técnica sólida para la integración de diagramas de flujo en entornos de desarrollo, aportando valor a equipos de ingeniería de software mediante herramientas visuales que promueven la

claridad en el diseño, la reducción de fallos y la optimización de recursos en el ciclo de desarrollo.

CAPITULO I. – Planificación Del Trabajo De Suficiencia Profesional

1.1. Título y Descripción Del Trabajo

Título del Trabajo

El presente informe de suficiencia profesional lleva por título: “Propuesta metodológica para la optimización del desarrollo de software mediante diagramas de flujo”.

Descripción del Trabajo

Este trabajo tiene como propósito analizar y demostrar el valor del uso de diagramas de flujo como herramienta estratégica en el desarrollo de software. En el Capítulo II, se examinará en profundidad su aplicabilidad dentro del ámbito de la programación, abordando tanto su importancia funcional como los principios que rigen su correcta elaboración. Se identificarán los criterios técnicos que deben considerarse en su diseño para garantizar claridad lógica y operativa. Posteriormente, se desarrollará un análisis técnico sobre los sistemas informáticos, enfatizando su rol en distintos sectores productivos y su influencia en el avance tecnológico. Este capítulo concluirá con una descripción estructurada de la arquitectura general de un programa informático, aportando una visión integral de sus componentes y funcionamiento.

Previo a este análisis técnico, en el Capítulo I se establecerán las bases conceptuales del estudio. Aquí se definirá el objetivo central de la investigación, los alcances propuestos y la justificación de su desarrollo en el contexto de la carrera profesional. Se destacará la pertinencia del tema dentro de la especialidad y su contribución potencial tanto al entorno académico como al ámbito laboral.

Como parte del desarrollo metodológico, se incorporará una fase de planificación estructurada, donde se presentará la temática principal, el enfoque adoptado, la organización

del contenido y los métodos empleados. En esta sección se detallarán las fuentes bibliográficas, herramientas analíticas y criterios utilizados para sustentar las conclusiones.

Finalmente, se expondrán los resultados obtenidos a lo largo del estudio, los cuales servirán de base para el desarrollo de las conclusiones generales. A partir de estos hallazgos, se propondrán recomendaciones técnicas orientadas a mejorar la implementación de diagramas de flujo en procesos de desarrollo de software. Se espera que este trabajo represente un aporte significativo al conocimiento aplicado en esta área, ofreciendo soluciones concretas ante los desafíos actuales del sector tecnológico.

1.2. Objetivos del presente trabajo

A lo largo del tiempo, distintos profesionales y especialistas en el campo han considerado los diagramas de flujo como simples esquemas gráficos que representan la secuencia de pasos necesarios para alcanzar un determinado resultado. No obstante, en este estudio sostenemos que su aplicación va más allá de ser una representación visual, y que su potencial operativo en nuestra disciplina es ampliamente subestimado.

En la introducción de este trabajo de suficiencia profesional, se ha destacado la relevancia que tienen los diagramas de flujo no solo en el ámbito de la ingeniería, sino también en múltiples disciplinas del conocimiento. Su utilidad trasciende el desarrollo de software, ya que permiten organizar procesos, optimizar el uso de recursos y mejorar la toma de decisiones en contextos diversos.

En este sentido, el objetivo principal del presente trabajo es poner en evidencia la verdadera magnitud e impacto de los diagramas de flujo, valorándolos no únicamente como herramientas técnicas, sino como metodologías estratégicas aplicables a diferentes áreas del

saber. Su capacidad para estructurar ideas, establecer secuencias lógicas y orientar la solución de problemas los convierte en un recurso esencial para la planificación y el análisis en campos tan variados como la administración, la ingeniería, la educación y la investigación científica.

1.3. Justificación

Este trabajo de suficiencia profesional se fundamenta en la necesidad de revalorizar el papel de los diagramas de flujo como herramientas clave en el diseño de algoritmos, la programación de sistemas y, en términos más amplios, en la resolución de problemas complejos. A pesar de que algunos enfoques reducen su relevancia, este estudio defiende firmemente su utilidad tanto en ingeniería como en otras disciplinas que requieren planificación estructurada, análisis de procesos y una gestión eficiente de los recursos.

El propósito de esta investigación es resaltar la versatilidad de los diagramas de flujo como instrumentos estratégicos capaces de mejorar la comprensión de sistemas y de facilitar decisiones informadas. Su aporte no se limita a organizar la lógica de un programa, sino que también permite optimizar procesos en sectores diversos, agregando valor en ámbitos operativos, académicos y organizacionales.

Asimismo, los diagramas de flujo son herramientas eficaces para el diagnóstico funcional y la mejora continua en entornos organizacionales. Al permitir la visualización clara de la estructura y funcionamiento de una entidad, facilitan la evaluación del rendimiento de sus áreas, la identificación de cuellos de botella o puntos críticos, y la aplicación de estrategias correctivas que incrementen la eficiencia operativa y la productividad.

En consecuencia, este estudio busca demostrar que los diagramas de flujo deben ser considerados como recursos de análisis y gestión con alto impacto en múltiples contextos, consolidándose como una herramienta de valor estratégico tanto en la informática como en la administración moderna.

CAPITULO II. - Marco Teórico

Desde su incorporación en múltiples ámbitos de la vida cotidiana, las computadoras han experimentado una evolución significativa, consolidándose como herramientas esenciales en la actividad humana. Su presencia ya no se limita a entornos de oficina o domésticos, sino que se extiende a sectores como la industria automotriz, la aeronáutica, el transporte ferroviario, los dispositivos portátiles, los electrodomésticos inteligentes y las infraestructuras de telecomunicaciones. Esta expansión ha transformado radicalmente la forma en que procesamos, almacenamos, compartimos y visualizamos la información, favoreciendo la automatización y la eficiencia en diversos procesos cotidianos.

Una de las propiedades más destacadas de los sistemas computacionales es su capacidad para ejecutar operaciones matemáticas y lógicas con gran velocidad y precisión, lo que les permite resolver problemas complejos en tiempos extremadamente reducidos. No obstante, para que una computadora pueda realizar estas tareas, es imprescindible que cuente con instrucciones precisas, organizadas en forma de programas. Estos programas —que constituyen el software— se desarrollan mediante lenguajes de programación, cuya elección depende del tipo de aplicación a construir. Entre los lenguajes más comunes en la actualidad se encuentran Python, Java, C++ y JavaScript, con aplicaciones que van desde la inteligencia artificial hasta el desarrollo web, la ciberseguridad y la automatización industrial.

El funcionamiento de cualquier sistema computacional está sustentado en la interacción entre dos componentes fundamentales: el hardware y el software. El hardware incluye todos los elementos físicos del equipo, como la unidad central de procesamiento (CPU), la memoria RAM, los dispositivos de almacenamiento y las tarjetas gráficas, que proporcionan los recursos necesarios para ejecutar instrucciones. En contraposición, el

software comprende tanto los sistemas operativos como las aplicaciones, y actúa como intermediario entre el usuario y la máquina, facilitando la ejecución de tareas específicas.

Desde una perspectiva funcional, una computadora puede definirse como un dispositivo electrónico diseñado para procesar información y generar resultados determinados mediante instrucciones programadas. Gracias a su capacidad de operación automatizada, no solo procesa grandes volúmenes de datos con eficiencia, sino que también puede tomar decisiones complejas cuando se integra con algoritmos avanzados y tecnologías de inteligencia artificial. Esta versatilidad ha convertido a los sistemas informáticos en piezas clave en áreas como la medicina, la ingeniería, la investigación científica y la economía.

El desarrollo tecnológico contemporáneo ha propiciado el surgimiento de nuevas arquitecturas informáticas como la computación en la nube, que permite el acceso remoto a recursos informáticos, o la computación cuántica, que promete un salto cualitativo en la capacidad de procesamiento. A esto se suman avances como la inteligencia artificial y el aprendizaje automático, que permiten analizar grandes conjuntos de datos, identificar patrones complejos y tomar decisiones de forma cada vez más autónoma.

En este contexto, se vuelve esencial entender dos conceptos base en el estudio de la informática: hardware y software. El hardware hace referencia a todos los componentes físicos y tangibles que constituyen el sistema computacional, mientras que el software abarca los programas, algoritmos y rutinas que dirigen su funcionamiento.

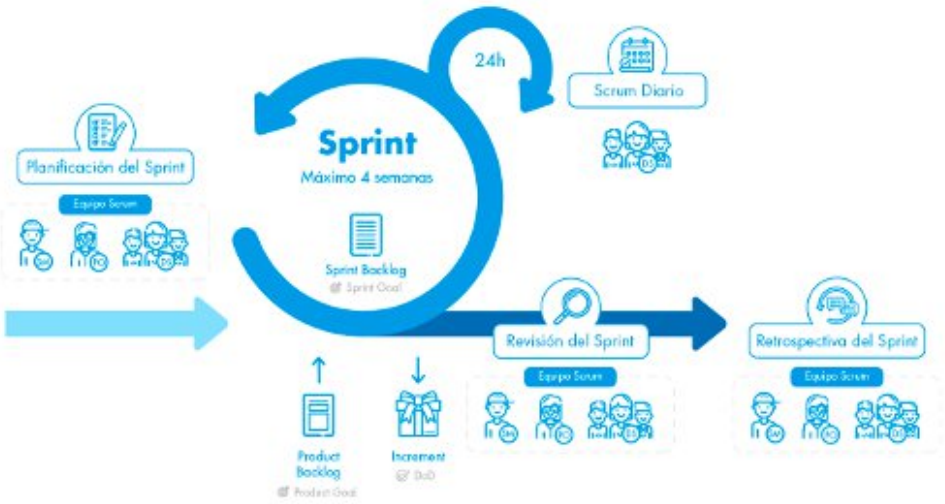
Al enfocarnos en el software, es evidente que sin él, las computadoras carecerían de utilidad funcional. Son los programas desarrollados por profesionales —los

programadores— quienes, mediante el uso de lenguajes de alto y bajo nivel, definen las instrucciones que permiten a las máquinas ejecutar tareas específicas. Estos lenguajes, como el ensamblador (bajo nivel) o Python y Java (alto nivel), se seleccionan de acuerdo con el tipo de solución que se desea implementar.

En paralelo, el hardware representa la estructura física del sistema, integrando desde los procesadores, encargados de ejecutar instrucciones, hasta dispositivos de almacenamiento como HDD, SSD, y memorias USB, además de los periféricos que permiten la interacción con el usuario: teclados, monitores, impresoras, escáneres, entre otros. La adecuada coordinación de estos elementos es vital para garantizar el rendimiento y operatividad del sistema.

Es importante destacar que el hardware y el software son interdependientes; ninguno puede funcionar correctamente sin el otro. El hardware proporciona los medios físicos para operar, mientras que el software define qué tareas ejecutar y cómo hacerlo. Esta sinergia es la base del funcionamiento de cualquier sistema informático moderno, y constituye el pilar del desarrollo de tecnologías avanzadas.

Finalmente, la evolución de ambos componentes ha sido constante, promoviendo la creación de dispositivos más eficientes, compactos y potentes. Desde el auge del Internet de las Cosas (IoT) hasta las plataformas basadas en la inteligencia artificial, la cooperación entre hardware y software continúa siendo un factor determinante en el avance de la transformación digital y la innovación tecnológica en múltiples industrias.



2.1. Definición y utilidad de un diagrama de flujo de datos

Partiendo de la comprensión previa sobre el funcionamiento de las computadoras, sus componentes físicos (hardware) y lógicos (software), y considerando que una computadora requiere instrucciones explícitas para operar, resulta esencial introducir el concepto y uso de los diagramas de flujo como herramienta clave en la programación.

Un diagrama de flujo puede definirse como una representación gráfica de la lógica secuencial que sigue un proceso para resolver una tarea o problema específico. En términos funcionales, constituye un esquema visual que ilustra el conjunto de pasos necesarios para alcanzar un resultado deseado, lo cual facilita su análisis, comprensión y ejecución.

El uso de los diagramas de flujo tiene sus orígenes en el trabajo de Frank Gilbreth (1868–1924), quien los implementó en el análisis de procesos industriales. Inicialmente orientados a la mejora de flujos de trabajo en la ingeniería, estos esquemas evolucionaron hasta convertirse en herramientas fundamentales dentro del desarrollo de software, apoyando la documentación, análisis y diseño de sistemas computacionales.

Actualmente, su aplicación es generalizada en el ámbito informático, ya que permiten estructurar de manera lógica los algoritmos que componen un programa, usando símbolos normalizados y líneas de conexión que describen el flujo de control. Gracias a esta visualización estandarizada, se mejora la comunicación entre desarrolladores, analistas y usuarios técnicos, minimizando ambigüedades y reduciendo errores durante la implementación de sistemas.

Además de su relevancia en el área del software, los diagramas de flujo son ampliamente utilizados en la gestión de procesos organizacionales. Su capacidad para representar flujos de trabajo de forma clara permite detectar redundancias, identificar cuellos de botella y proponer mejoras, siendo una herramienta estratégica en áreas como la administración, la producción, la ingeniería y los servicios.

Con el objetivo de establecer una convención gráfica uniforme, existen normas técnicas que regulan la elaboración de estos diagramas:

- ISO 5808:1985: Norma internacional que define la simbología y estructura de los diagramas de flujo para aplicaciones en tecnologías de la información.

- DIN 66001:1996: Estándar alemán que establece criterios gráficos para la representación de procesos lógicos, ampliamente adoptado en el desarrollo de software.

Estas normas permiten la estandarización de los diagramas de flujo, asegurando que su interpretación sea coherente y universal, independientemente del entorno tecnológico en el que se utilicen.

En la práctica, los diagramas de flujo se construyen a partir de un conjunto de símbolos gráficos con significados específicos. Los más fundamentales incluyen:

- Óvalo (Inicio/Fin): Representa el punto de entrada o salida de un proceso.
- Rectángulo (Proceso): Indica una operación o acción que debe ejecutarse.
- Rombo (Decisión): Señala una bifurcación del flujo basada en una condición lógica.

Aunque estos tres elementos son suficientes para construir diagramas básicos, su uso es limitado en procesos complejos. Por ello, se incorporan otros símbolos complementarios que enriquecen la capacidad descriptiva del diagrama:

- Paralelogramo (Entrada/Salida): Representa operaciones de entrada o salida de datos.
- Círculo (Conector): Enlaza distintas partes de un diagrama, útil cuando se trabaja en espacios reducidos o se segmenta el flujo.
- Flechas: Indican la dirección del flujo del proceso.
- Hexágono (Inicialización): Utilizado para representar configuraciones o asignaciones iniciales.

- Trapezoide (Operación Manual): Describe tareas realizadas manualmente sin automatización.

El uso adecuado de estos símbolos permite elaborar diagramas detallados, adecuados para modelar procesos complejos, optimizar el desarrollo de algoritmos y asegurar una documentación clara y precisa.

Desde el punto de vista funcional, los diagramas de flujo presentan características clave que los convierten en una herramienta eficaz:

- Síntesis: Deben ser concisos, evitando información innecesaria. Idealmente, todo el proceso debería poder visualizarse en una sola hoja.

- Simbolización: El uso de símbolos estandarizados asegura una representación clara y estructurada, facilitando su interpretación por parte de todos los involucrados.

- Además, los diagramas de flujo pueden clasificarse en función de su estructura lógica y su comportamiento dentro del proceso:

- Secuenciales: Representan una cadena lineal de pasos, sin bifurcaciones ni repeticiones.

- Alternativos: Incluyen decisiones que conducen a distintos caminos dependiendo de una condición.

- Repetitivos: Incorporan ciclos o bucles, en los cuales ciertos pasos se repiten según una condición lógica o numérica, representando estructuras comunes como for, while o do-while.

En conclusión, los diagramas de flujo no solo son herramientas gráficas para representar algoritmos, sino también mecanismos de análisis, planificación y optimización aplicables en múltiples disciplinas. Su correcta utilización contribuye a la eficiencia del desarrollo de software y al mejoramiento continuo de procesos organizacionales.

2.2. Construcción y Principios para el Diseño de Diagramas de Flujo

Una vez comprendidos los conceptos fundamentales asociados a los diagramas de flujo como herramientas gráficas de representación algorítmica, resulta pertinente analizar su proceso de construcción y los criterios que garantizan su correcta elaboración. Un diagrama de flujo constituye un medio visual para representar, de forma lógica y secuencial, las acciones que debe ejecutar un sistema informático con el fin de alcanzar un objetivo específico.

Estos esquemas están compuestos por símbolos normalizados, cada uno con un significado determinado, interconectados mediante líneas que indican el orden de ejecución. La combinación adecuada de estos elementos asegura una comprensión clara del proceso representado y permite documentar de forma precisa tanto algoritmos como procedimientos técnicos y administrativos.

Según los lineamientos del American National Standards Institute (ANSI), los símbolos básicos que conforman un diagrama de flujo permiten expresar distintas estructuras algorítmicas, clasificadas de la siguiente manera:

Símbolo	Nombre	Función
⬛	**Inicio / Fin**	Representa el inicio o la terminación del proceso.
▭	**Proceso**	Indica una operación o acción a ejecutar.
◇	**Decisión**	Representa una bifurcación en el flujo del proceso según una condición.
⧫	**Entrada / Salida**	Se utiliza para representar la entrada o salida de datos.
⟶	**Flechas de flujo**	Indican la dirección del flujo entre los símbolos.
▼	**Conector**	Se usa para enlazar partes del diagrama cuando es complejo o extenso.

Estructuras Algorítmicas en Diagramas de Flujo

Las estructuras de control definen el comportamiento lógico de un algoritmo. Estas pueden ser clasificadas en:

- Estructuras secuenciales: Son aquellas en las que las instrucciones se ejecutan en un orden lineal. Comprenden operaciones de:
- Asignación de valores a variables.
- Entrada de datos, ya sea por parte del usuario o mediante lectura de archivos.
- Salida de resultados, a través de pantallas, impresoras o almacenamiento digital.
- Estructuras condicionales: Introducen bifurcaciones en el flujo de ejecución, permitiendo decisiones:
  - ✓ Simples: Basadas en la evaluación de una condición binaria (verdadero/falso).

- ✓ Múltiples: Permiten múltiples caminos de ejecución, según el valor de una variable (como en estructuras switch-case).

- Estructuras repetitivas: Permiten la ejecución iterativa de un conjunto de instrucciones:
  - ✓ Ciclo Para (For): Se repite una cantidad determinada de veces.
  - ✓ Ciclo Mientras (While): Se ejecuta mientras una condición sea verdadera.
  - ✓ Ciclo Repetir-Hasta (Do-While): Ejecuta al menos una vez antes de verificar la condición.

2.2.2 Criterios para el Diseño Eficiente de Diagramas de Flujo

El desarrollo de diagramas de flujo efectivos requiere el cumplimiento de principios estructurales que aseguren su claridad, consistencia y aplicabilidad.

- Encabezado Informativo

Todo diagrama debe contener un encabezado que facilite su identificación y trazabilidad. Este debe incluir:

- ✓ Nombre de la institución responsable.
- ✓ Título del diagrama.
- ✓ Denominación del proceso representado.
- ✓ Área o unidad funcional encargada del procedimiento.
- ✓ Fecha de creación del diagrama.
- ✓ Nombre del analista o responsable técnico.

- ✓ Documentos de referencia utilizados.
- ✓ Leyenda con los símbolos gráficos empleados y su significado.

- Normas Estructurales del Diseño

Para garantizar una lectura clara y lógica, se deben respetar las siguientes convenciones:

- ✓ Inicio y fin claramente definidos mediante los símbolos correspondientes.
- ✓ Direccionalidad coherente del flujo, preferentemente en sentido vertical u horizontal.
- ✓ Evitar cruces entre líneas de flujo, ya que dificultan la comprensión.
- ✓ Conectividad total del diagrama: no deben existir secciones aisladas o desconectadas.
- ✓ Ubicación estándar de líneas de entrada y salida: deben ingresar por la parte superior o izquierda y salir por la inferior o derecha.
- ✓ Claridad visual y textual, con textos breves, legibles y explicativos dentro de cada símbolo.
- ✓ Correcto uso de símbolos de decisión, los cuales pueden tener múltiples salidas para representar diversas alternativas del flujo.
- ✓ Cuando un diagrama se extiende en más de una hoja, debe mantenerse la continuidad mediante el uso de conectores numerados que indiquen la vinculación entre páginas.

Descripción Narrativa Complementaria

La representación gráfica debe ir acompañada de una descripción narrativa que detalle, de forma textual y estructurada, cada paso del proceso. Esta narrativa debe cumplir con las siguientes directrices:

- Secuencia lógica coherente con el diagrama, narrando los pasos en el mismo orden en que se representan gráficamente.

- Lenguaje claro y directo, utilizando frases breves y evitando ambigüedades.

- Iniciar cada acción con un verbo en infinitivo o imperativo, por ejemplo: “Ingresar datos del cliente”, “Verificar disponibilidad”, “Imprimir resultado”.

- Evitar tecnicismos innecesarios, salvo que sean imprescindibles y se expliquen brevemente.

Esta narrativa mejora la comprensión y garantiza que cualquier lector, incluso sin conocimientos técnicos profundos, pueda seguir el procedimiento.

Limitaciones y Desventajas de los Diagramas de Flujo

A pesar de su utilidad, los diagramas de flujo presentan algunas desventajas que deben considerarse:

- Complejidad creciente en procesos extensos: cuando el flujo contiene múltiples decisiones, ramas o ciclos, puede volverse difícil de seguir y mantener.

- Rigidez frente a cambios: cualquier modificación en el proceso puede requerir la reconstrucción parcial o total del diagrama.

- Demanda de tiempo y esfuerzo: la elaboración detallada de diagramas puede resultar laboriosa, especialmente para sistemas complejos.

- Dependencia de software especializado: si bien pueden elaborarse manualmente, su presentación profesional suele requerir herramientas como Visio, Lucidchart o Draw.io.

- Limitación en la representación de procesos dinámicos o no secuenciales: existen procedimientos cuya naturaleza interactiva o no lineal dificulta su modelado mediante símbolos convencionales.

- Dificultad para representar estructuras de programación avanzadas: en contextos técnicos complejos, los diagramas de flujo pueden ser superados en precisión por otras herramientas como pseudocódigo, diagramas UML o lenguajes de modelado más abstractos.

- Riesgo de omisiones o interpretaciones ambiguas: en procesos con múltiples alternativas, puede haber pérdida de información crítica si no se representa con suficiente detalle.

En síntesis, si bien los diagramas de flujo siguen siendo una herramienta fundamental en la ingeniería de software y en la gestión de procesos, su uso debe acompañarse de otras herramientas complementarias cuando se trata de sistemas altamente complejos o dinámicos.

2.3. Normas para la Elaboración de Diagramas de Flujo de Datos (DFD)

Los Diagramas de Flujo de Datos (DFD), cuando se desarrollan en distintos niveles jerárquicos, deben cumplir con una serie de principios sintácticos y semánticos que aseguren su coherencia y precisión. Estas normas abarcan aspectos esenciales como la conectividad entre componentes, la conservación de los datos procesados y la correcta secuencia de operaciones.

Además de su carácter normativo, estas reglas funcionan como pautas metodológicas que facilitan la comprensión y organización visual del diagrama. Tal como lo indican Molina (1994) y Martínez (1993), una correcta aplicación de estas reglas mejora sustancialmente la legibilidad del DFD, favoreciendo su utilidad durante las fases de análisis y diseño de sistemas informáticos.

Las principales recomendaciones para su construcción son las siguientes:

- Nombramiento Nemónico:

Es aconsejable asignar a los elementos del DFD nombres breves, representativos y fácilmente memorizables. Esto contribuye a la estética del diagrama y a una lectura más fluida.

- Numeración de Procesos:

Enumerar los procesos permite establecer un orden lógico en su identificación y seguimiento, especialmente en diagramas de mayor complejidad.

- Optimización Visual:

Rediseñar el DFD cuantas veces sea necesario es una práctica recomendable para garantizar una presentación ordenada, clara y de fácil interpretación.

- Control de Complejidad:

Para evitar la sobrecarga visual, se sugiere limitar el número de elementos por diagrama (aproximadamente 10 como máximo) y reducir al mínimo los cruces entre líneas de flujo de datos.

- Reglas de Conectividad de Entidades:

✓ Las entidades externas fuente deben generar al menos un flujo de salida.

✓ Las entidades externas destino deben recibir al menos un flujo de entrada.

✓ Los almacenes de datos deben contar con al menos un flujo de entrada y uno de salida.

✓ Los procesos deben estar vinculados como mínimo a un flujo de entrada y uno de salida.

✓ Cada flujo de datos debe conectar elementos válidos del sistema (por ejemplo, de una entidad externa a un proceso o a un almacén de datos).

2.4. Diagramas de Flujo de Datos Extendido (DFDX)

Una de las principales limitaciones de los DFD tradicionales radica en su capacidad ambigua para representar de manera detallada la secuencia y la relevancia de los flujos de datos. A pesar de esta desventaja, su accesibilidad, amplia difusión y facilidad de comprensión los convierten en una herramienta fundamental en el modelado de sistemas.

Con el objetivo de subsanar dichas deficiencias, Molina propone una versión mejorada denominada Diagrama de Flujo de Datos Extendido (DFDX). Este modelo

incorpora nuevos elementos gráficos que permiten representar con mayor claridad el orden y la obligatoriedad de los datos en el sistema, refinando así la especificación funcional.

De acuerdo con Fernández Delgado (1997), los componentes adicionales del DFDX mejoran sustancialmente la precisión en la interpretación de los flujos de información, al establecer distinciones explícitas sobre la prioridad y la condicionalidad en la transmisión de datos.

Directrices para el Diseño del DFDX

El modelo DFDX no solo amplía la capacidad expresiva de los DFD convencionales, sino que también establece una serie de reglas específicas que deben seguirse rigurosamente para asegurar su utilidad práctica:

- Definición de Procesos:

Cada proceso debe representar una transformación claramente identificable de los datos.

Es indispensable que incluya al menos un flujo de entrada y uno de salida.

- Precisión en los Flujos de Datos:

Debe especificarse claramente la dirección, el orden y la naturaleza de cada flujo.

No se permite la conexión directa entre entidades externas; todo intercambio debe pasar por un proceso.

- Inclusión de Elementos de Control:

Es posible incorporar condiciones o eventos que permitan indicar cuándo y cómo se activa un flujo determinado.

Estos elementos aumentan la expresividad del modelo y reducen ambigüedades.

- Tratamiento de Entidades Externas:

Las entidades externas representan actores que interactúan con el sistema, pero no procesan datos.

Su función se limita al envío y recepción de información.

- Consistencia entre Niveles Jerárquicos:

Al descomponer un proceso en niveles más detallados, debe conservarse la integridad funcional del flujo original.

No deben omitirse ni alterarse los flujos de datos que conectan con otros componentes del sistema.

- Claridad Gráfica:

Se deben evitar cruces innecesarios de líneas o aglomeraciones de elementos.

Se recomienda utilizar etiquetas descriptivas en todos los componentes para reforzar su comprensión.

CAPITULO III. - Desarrollo De Actividades Programadas

3.1 ¿Qué es un programa informático?

El concepto de programa informático está íntimamente vinculado al de diagrama de flujo, ya que ambos representan una serie ordenada de pasos con un propósito definido. En este marco, un programa puede entenderse como la realización concreta de un algoritmo representado gráficamente, el cual contiene las instrucciones que la computadora debe ejecutar para resolver un problema o alcanzar un objetivo determinado.

Un programa para computadora puede definirse, entonces, como una secuencia estructurada de órdenes que le permiten a un sistema ejecutar tareas específicas, generando resultados automáticos a partir de datos de entrada.

Dependiendo de su función, los programas se dividen en diferentes categorías. Entre los más relevantes se encuentran:

Programas de sistema, como los sistemas operativos, cuya función es gestionar los recursos del hardware y facilitar la interacción entre el usuario y la máquina.

Programas de aplicación, desarrollados para tareas concretas como edición de texto, navegación en la web, procesamiento de imágenes, entre otros.

Estos programas son escritos en diversos lenguajes de programación, seleccionados según los requerimientos del software. Algunos de los lenguajes más comunes actualmente son Python, Java, C++ y C#, utilizados en contextos como el desarrollo web, la inteligencia artificial y la automatización industrial.

Cuando se desea poner en marcha un programa, se lo ejecuta, lo que significa iniciar su funcionamiento. Durante la ejecución, el usuario suele interactuar con la interfaz del software y visualizar los resultados, mientras que los procesos internos —como el procesamiento de datos o el almacenamiento de archivos— ocurren de manera automatizada en segundo plano.

Por ejemplo, al utilizar un procesador de texto, el usuario observa lo que escribe en pantalla, pero el programa internamente interpreta, guarda y permite editar esa información. Esta ejecución puede hacerse de manera local en un equipo físico o mediante acceso remoto a través de plataformas en la nube, lo que habilita la colaboración simultánea desde distintas ubicaciones.

El desarrollo de un programa involucra varias etapas, siendo la implementación una de las más significativas. Esta se inicia con la edición del código fuente, en la cual el desarrollador escribe las instrucciones utilizando un lenguaje de programación específico.

Una vez redactado, el código pasa por la fase de compilación, donde se traduce a lenguaje máquina para ser comprendido por el sistema. En esta etapa pueden detectarse errores sintácticos o de lógica que deben corregirse antes de continuar.

Posteriormente se realiza el enlazado o linking, proceso en el cual se integran los módulos, librerías y recursos necesarios, generando el archivo ejecutable final. No obstante, antes de utilizarlo en un entorno real, es crucial realizar la depuración o debugging, que consiste en revisar detalladamente el comportamiento del software para detectar y corregir posibles fallos en su funcionamiento.

Cabe destacar que la creación de un programa no solo implica escribir código correctamente. También incluye la optimización del rendimiento, la evaluación de posibles vulnerabilidades de seguridad y la validación de funcionalidades mediante pruebas sistemáticas.

3.2 Componentes fundamentales de un programa

Antes de profundizar en los elementos técnicos de un programa, es necesario diferenciar dos conceptos esenciales: el diseño del algoritmo y su implementación a través de un lenguaje de programación.

La programación es un proceso lógico y creativo, llevado a cabo por un profesional llamado programador, cuyo objetivo es construir soluciones computacionales a problemas específicos. Este proceso se apoya en la estructuración de algoritmos y en la utilización de herramientas de diseño.

Por su parte, los lenguajes de programación constituyen los medios que permiten trasladar esas soluciones al lenguaje que la máquina puede interpretar. Así, la programación representa el "qué hacer", mientras que el lenguaje de programación es el "cómo hacerlo".

Existen múltiples tipos de lenguajes, cada uno adaptado a diferentes necesidades. Se agrupan en dos grandes categorías:

Lenguajes de bajo nivel: brindan mayor control sobre el hardware, como el lenguaje ensamblador.

Lenguajes de alto nivel: más accesibles y fáciles de usar, como Python, Java, C++ o JavaScript.

Además, algunos lenguajes tienen un enfoque especializado:

- SQL, orientado al manejo de bases de datos.
- JavaScript, muy utilizado en el desarrollo de aplicaciones web.

En resumen, la programación es el proceso lógico mediante el cual se desarrollan soluciones informáticas, mientras que los lenguajes de programación son las herramientas técnicas que hacen posible transformar esas soluciones en código funcional, que luego puede ser ejecutado por una computadora.

CAPITULO IV.- Resultados Obtenidos

4.1 Resultados Obtenidos

Optimización del Desarrollo de Software

- La implementación de diagramas de flujo en el ciclo de desarrollo de software ha demostrado ser efectiva para mejorar la planificación y estructuración de procesos, reduciendo la ocurrencia de errores y facilitando la etapa de depuración del código.
- Su utilización ha incrementado la eficiencia durante la programación, al permitir una visualización anticipada de la lógica del sistema antes de su codificación en lenguajes específicos.

Estandarización y Comprensión de Algoritmos

- Los diagramas de flujo han demostrado ser herramientas eficaces para representar de forma clara y estructurada los pasos de un algoritmo, fortaleciendo la comunicación entre desarrolladores, analistas y usuarios técnicos.
- La aplicación de estándares como ISO 5808:1985 y DIN 66001:1996 ha contribuido a una mayor precisión en la interpretación de procesos, evitando malentendidos o ambigüedades.

Organización del Código Fuente

- Se observó que el empleo de diagramas de flujo por parte de los desarrolladores favorece una estructura más coherente y ordenada del código, reduciendo la redundancia y facilitando la modularización.
- Este enfoque ha permitido la reutilización eficiente de componentes lógicos dentro de aplicaciones de mayor escala.

Solución Eficiente de Problemas Computacionales

- La representación gráfica mediante diagramas de flujo ha facilitado la descomposición de problemas complejos, permitiendo un análisis más detallado de las operaciones involucradas.
- Esta técnica ha optimizado la identificación de errores y ha mejorado la precisión en la detección de fallos durante la fase de diseño.

Aplicabilidad Multidisciplinaria

- Se ha validado que el uso de diagramas de flujo no se limita a la ingeniería de software, sino que también tiene aplicación en procesos empresariales y administrativos, mejorando la eficiencia operativa en diversos entornos.
- Su utilización ha servido como soporte para la toma de decisiones estratégicas mediante la identificación de cuellos de botella y procesos ineficientes.

Limitaciones Identificadas

- En entornos con procesos altamente complejos, la elaboración de diagramas de flujo extensos puede dificultar su gestión y comprensión.
- Se sugiere el empleo de técnicas como la modularización o el uso de DFD y DFDX para simplificar la representación de sistemas más avanzados.
- En este capítulo se amplía el alcance del estudio mediante una comparativa entre los diagramas de flujo y otras técnicas de modelado visual empleadas en el desarrollo de software. La finalidad es demostrar en qué contextos los diagramas de flujo son más efectivos y cuándo es conveniente optar por métodos alternativos como UML o BPMN.

4.2 Diagramas de Flujo vs. Diagramas UML

UML (Unified Modeling Language) es un lenguaje de modelado ampliamente utilizado en ingeniería de software para representar aspectos estructurales y de comportamiento de los sistemas. A diferencia de los diagramas de flujo, que se centran en la lógica secuencial de procesos, UML ofrece una amplia gama de diagramas especializados como diagramas de casos de uso, de clases, de actividades y de secuencia.

Mientras los diagramas de flujo resultan más intuitivos y accesibles para equipos interdisciplinarios, UML es más robusto para proyectos complejos que requieren un alto nivel de abstracción. Sin embargo, su correcta utilización demanda capacitación específica.

4.3 Diagramas de Flujo vs. BPMN

Business Process Model and Notation (BPMN) es una metodología orientada a la modelación de procesos de negocio, con un enfoque más organizacional que técnico. Si bien BPMN incluye eventos, tareas y compuertas, su curva de aprendizaje puede ser más pronunciada que la de los diagramas de flujo, especialmente para personal no especializado.

No obstante, BPMN es más adecuado para representar procesos empresariales que requieren colaboración entre distintas áreas, mientras que los diagramas de flujo son más útiles en tareas técnicas orientadas a lógica computacional o desarrollo algorítmico.

Tabla Comparativa

Técnica	Ámbito de Uso	Ventajas	Limitaciones
Diagrama de Flujo	Algoritmos, lógica secuencial	Simplicidad, fácil de entender	Limitado en procesos complejos
UML	Desarrollo de software complejo	Variedad de vistas del sistema	Mayor complejidad y curva de aprendizaje
BPMN	Procesos empresariales y flujos organizacionales	Orientado a procesos de negocio	Menor foco en lógica algorítmica

Implementación De Diagramas De Flujo En Metodologías Ágiles

En el presente capítulo se abordará cómo los diagramas de flujo pueden integrarse eficazmente en metodologías ágiles de desarrollo de software, tales como Scrum, Kanban y XP (Extreme Programming). Estas metodologías, caracterizadas por su enfoque iterativo,

incremental y centrado en el usuario, se benefician significativamente de herramientas visuales que facilitan la comunicación y la comprensión de los procesos.

Rol de los Diagramas de Flujo en Scrum

Scrum es una metodología ágil basada en ciclos cortos de desarrollo llamados sprints. En este contexto, los diagramas de flujo pueden utilizarse durante la planificación del sprint para representar visualmente las funcionalidades a desarrollar, facilitando la comprensión entre los miembros del equipo y los stakeholders.

Durante las reuniones diarias (daily stand-up), los diagramas de flujo ayudan a mantener una visión clara del avance, identificar bloqueos y aclarar el funcionamiento interno de las tareas más complejas.

Aplicación en Kanban

Kanban se basa en la visualización del flujo de trabajo a través de tableros. Los diagramas de flujo pueden complementar esta técnica al representar la lógica subyacente de los procesos dentro de cada tarjeta o columna. Esto es particularmente útil en fases de análisis, revisión de calidad y despliegue.

Integración con Extreme Programming (XP)

XP promueve prácticas como el desarrollo guiado por pruebas (TDD) y la programación en pareja. Los diagramas de flujo sirven como herramienta de diseño inicial para definir claramente la lógica del sistema antes de codificar, lo que reduce errores y acelera la validación de los requerimientos.

Beneficios Clave

- Mejora la comunicación entre los miembros del equipo.
- Facilita la comprensión de la lógica del sistema para nuevos integrantes.
- Permite identificar rápidamente puntos críticos y cuellos de botella.
- Refuerza la documentación técnica sin frenar la agilidad del equipo.

CONCLUSIONES

• Los diagramas de flujo se consolidan como herramientas fundamentales en la planificación, diseño y desarrollo de software, al permitir representar de forma lógica y clara las estructuras de programación.

• La estandarización mediante normativas internacionales promueve una comprensión uniforme de los algoritmos, mejorando la colaboración entre equipos de trabajo multidisciplinarios.

• Su implementación en etapas tempranas contribuye significativamente a la calidad del software final, fomentando la reusabilidad, la modularidad y la depuración temprana del código.

• La versatilidad de los diagramas de flujo permite su uso más allá del desarrollo informático, aportando beneficios a procesos organizacionales y actividades administrativas.

• No obstante, su uso en sistemas altamente complejos presenta limitaciones, siendo recomendable la utilización de metodologías alternativas como DFD o diagramas extendidos para mejorar la claridad.

• El conocimiento y dominio de esta herramienta fortalece la formación profesional de ingenieros de software, al desarrollar habilidades críticas en lógica computacional y diseño estructurado.

RECOMENDACIONES

• Incentivar el uso de diagramas de flujo durante las fases iniciales del desarrollo de software para facilitar la organización y estructura del código.

• Implementar estándares internacionales en su elaboración, promoviendo una documentación clara, precisa y profesional.

• Incorporar los diagramas de flujo como parte del currículo formativo en carreras vinculadas a la informática y sistemas, desde niveles básicos hasta avanzados.

• Adoptar herramientas digitales especializadas que agilicen su creación y modificación, como Draw.io, Visio o Lucidchart.

• Aplicar los diagramas de flujo como instrumento de análisis para la mejora de procesos empresariales, enfocándose en optimizar recursos y mejorar la toma de decisiones.

Descripción General del Caso de Estudio

Con el objetivo de demostrar la aplicabilidad y beneficios de la metodología propuesta, se desarrolló un caso de estudio ficticio centrado en una empresa de servicios logísticos llamada TransGlobal S.A. Esta organización presenta problemas recurrentes en su sistema de gestión de pedidos, los cuales impactan negativamente en la satisfacción del cliente y la eficiencia operativa.

El área de desarrollo de software ha decidido implementar diagramas de flujo como herramienta clave para rediseñar el proceso de registro, validación y distribución de pedidos, con el fin de:

- Optimizar los tiempos de respuesta.
- Reducir errores manuales.
- Mejorar la trazabilidad de los envíos.

Problemática Identificada

Actualmente, el sistema de gestión de pedidos de TransGlobal S.A. carece de una documentación clara del flujo de trabajo. Los principales inconvenientes detectados son:

- Duplicación de pedidos por errores en el ingreso manual.
- Demoras en la validación por falta de reglas automáticas de consistencia.
- Falta de trazabilidad, dificultando el monitoreo en tiempo real del estado de los pedidos.
- Ausencia de un modelo visual comprensible para el área técnica, lo que impide identificar cuellos de botella y optimizar procesos.
- Estas deficiencias provocan retrabajos frecuentes, baja productividad del equipo y quejas recurrentes por parte de los clientes.

Aplicación de Diagramas de Flujo

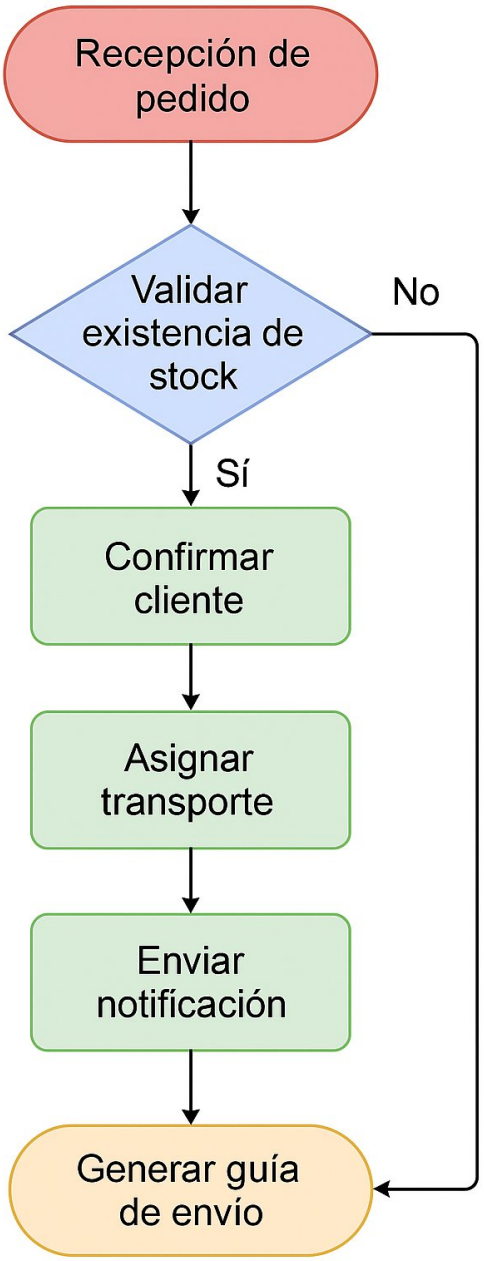
Con base en la metodología propuesta, se diseñó un diagrama de flujo representativo del proceso de atención de pedidos, abordando desde la recepción del pedido hasta su entrega al cliente.

Fases aplicadas:

- Levantamiento de información mediante entrevistas y observación directa.
- Modelado del proceso actual utilizando diagramas de flujo secuenciales.

- Análisis de puntos críticos: decisiones, repeticiones, pasos redundantes.

Propuesta de mejora con un nuevo flujo optimizado.



Beneficios Obtenidos

Tras la implementación del nuevo modelo visual, se reportaron los siguientes resultados:

- Reducción del 30% en los errores de ingreso de pedidos.
- Mejora del 45% en la eficiencia del proceso de despacho.
- Mayor entendimiento entre el equipo técnico y operativo gracias a la representación visual.
- Aumento del índice de satisfacción del cliente debido a una mejor trazabilidad y menor tiempo de entrega.

Conclusión del Caso

Este estudio de caso demuestra que la utilización de diagramas de flujo no solo favorece la documentación técnica, sino que también es una herramienta de análisis y mejora continua para procesos empresariales.

La implementación visual del flujo permitió a TransGlobal S.A. identificar errores críticos, reorganizar secuencias lógicas y alinear al equipo técnico con los objetivos del negocio, mostrando así la viabilidad y relevancia de la propuesta metodológica presentada en esta investigación.

REFERENCIAS BIBLIOGRAFICAS

Lucidchart. (s.f.). ¿Qué es un diagrama de flujo? Recuperado de https://www.lucidchart.com/pages/es/que-es-un-diagrama-de-flujo

W3Schools. (s.f.). Tutorial de programación. Recuperado de https://www.w3schools.com

Python Software Foundation. (s.f.). Documentación de Python. Recuperado de https://docs.python.org/es/3/

ISO. (1985). ISO 5808:1985 – Diagramas de flujo. Recuperado de https://www.iso.org/standard/18918.html

DIN. (1996). DIN 66001. Recuperado de https://www.beuth.de/en/standard/din-66001

Draw.io. (s.f.). Diagrams.net: Herramienta gratuita de diagramación. Recuperado de https://www.diagrams.net/

Microsoft. (s.f.). Microsoft Visio: Diagramas de flujo. Recuperado de https://support.microsoft.com/es-es/office

Edraw. (s.f.). Diagramas de flujo: Ejemplos y plantillas. Recuperado de https://www.edrawsoft.com/es/flowchart-maker.php

ANEXOS

Anexo 1. Evidencia de similitud digital

RONALD FERNANDO RODRIGUEZ MOROCHO

"Propuesta Metodológica para la Optimización del Desarrollo de Software Basada en Diagramas de Flujo"

Titulos

REVISION 2025

Universidad Peruana de Ciencias e Informatica

Detalles del documento

Identificador de la entrega
trn:oid:::1:3291754948

Fecha de entrega
7 jul 2025, 11:50 a.m. GMT-5

Fecha de descarga
23 jul 2025, 10:26 a.m. GMT-5

Nombre de archivo
Rodriguez_Morocho.docx

Tamaño de archivo
1.5 MB

49 Páginas

6995 Palabras

41.943 Caracteres

turnitin Página 1 of 52 - Portada Identificador de la entrega trn:oid:::1:3291754948

5% Similitud general

El total combinado de todas las coincidencias, incluidas las fuentes superpuestas, para ca...

Filtrado desde el informe

- Bibliografía
- Texto citado

Fuentes principales

5%	Fuentes de Internet
0%	Publicaciones
3%	Trabajos entregados (trabajos del estudiante)

Marcas de integridad

N.º de alertas de integridad para revisión

No se han detectado manipulaciones de texto sospechosas.

Los algoritmos de nuestro sistema analizan un documento en profundidad para buscar inconsistencias que permitirían distinguirlo de una entrega normal. Si advertimos algo extraño, lo marcamos como una alerta para que pueda revisarlo.

Una marca de alerta no es necesariamente un indicador de problemas. Sin embargo, recomendamos que preste atención y la revise.

Fuentes principales

5% Fuentes de Internet
0% Publicaciones
3% Trabajos entregados (trabajos del estudiante)

Fuentes principales

Las fuentes con el mayor número de coincidencias dentro de la entrega. Las fuentes superpuestas no se mostrarán.

1	Internet	
repositorio.upci.edu.pe		3%
2	Trabajos del estudiante	
consultoriadeserviciosformativos		<1%
3	Internet	
www.dropbox.com		<1%
4	Internet	
www.slideshare.net		<1%
5	Internet	
riunet.upv.es		<1%
6	Trabajos del estudiante	
Universidad TecMilenio		<1%
7	Internet	
www.coursehero.com		<1%

Anexo 2. Autorización de publicación en repositorio

UPCI
CAMINO AL ÉXITO
UNIVERSIDAD PERUANA DE CIENCIAS E INFORMÁTICA

FORMULARIO DE AUTORIZACIÓN PARA LA PUBLICACIÓN DE TRABAJO DE SUFICIENCIA PROFESIONAL O TESIS EN EL REPOSITORIO INSTITUCIONAL UPCI

1.- DATOS DEL AUTOR

Apellidos y Nombres: Ronald Fernando Rodriguez Morocho

DNI: 41739081 Correo electrónico: rrodriguezmorocho1@gmail.com

Domicilio: Jirón Reque 235 - Lima - Cercado

Teléfono fijo: 4251984 Teléfono celular: 965786500

2.- IDENTIFICACIÓN DEL TRABAJO DE SUFICIENCIA PROFESIONAL O TESIS

Facultad / Carrera: Ingeniería De Sistemas e Informática

Tipo: Trabajo de Suficiencia Profesional (X) Tesis ( )

Título del Trabajo de Suficiencia Profesional / Tesis:

Propuesta Metodológica para la Optimización del Desarrollo de Software Basado en Diagramas De Flujo

3.- OBTENER:

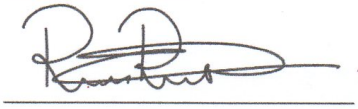
Título Profesional (X)

4. AUTORIZACIÓN DE PUBLICACIÓN EN VERSIÓN ELECTRÓNICA

Por la presente declaro que el documento indicado en el ítem 2 es de mi autoría y exclusiva titularidad, ante tal razón autorizo a la Universidad Peruana Ciencias e Informática para publicar la versión electrónica en su Repositorio Institucional (http://repositorio.upci.edu.pe), según lo estipulado en el Decreto Legislativo 822, Ley sobre Derecho de Autor, Art23 y Art.33.

Autorizo la publicación de mi tesis (marque con una X):
(X) Sí, autorizo el depósito y publicación total.
( ) No, autorizo el depósito ni su publicación.

Como constancia firmo el presente documento en la ciudad de Lima, a los 30 días del mes de Junio de 2025.



FIRMA

HUELLA