Tecnicatura Universitaria en Programación (TUP-2024)

Metodología en Sistemas I
Profesora Petrelli Nadia

Trabajo Práctico N°3
Tema: "El ciclo de vida del proyecto"
Grupo 3

Integrantes:

- Díaz Medina Tomás
- Díaz Rossini Juan José
- Gallo Genaro
- Hinojosa Alexis
- Liacoplo Gerónimo Emiliano
- Navarro Victor Leandro

Consigna

1. Mencione dos sinónimos de metodología.

2. **¿Cuál es la diferencia entre una herramienta, como se utiliza en este libro, y una metodología?**

3. **¿Cuáles son los tres principales propósitos del ciclo de vida de un proyecto?**

4. **Proyecto de investigación:** Encuentre el precio de tres productos para metodología comerciales ofrecidos por proveedores de software o empresas consultoras

5. **¿Por qué normalmente las organizaciones pequeñas de proceso de datos no usan metodologías formales?**

6. **¿Por qué es útil para los administradores nuevos una metodología?**

7. **¿Por qué es importante tener una metodología en una organización en la que se estén llevando a cabo muchos proyectos diferentes?**

8. **¿Cómo es que una metodología es útil para controlar proyectos?**

9. **¿Cuáles son las principales características que distinguen el ciclo de vida clásico?**

10. **¿Qué significa la expresión implantación ascendente?**

11. **¿Cuáles son las cuatro principales dificultades con la estrategia de implantación ascendente?**

12. **¿Qué tipo de ambiente es el adecuado para un enfoque de implantación ascendente?**

13. **¿Por qué es importante que "nada está hecho hasta que todo esté hecho", que además es lo que caracteriza al enfoque ascendente?**

14. **¿Por qué debieran encontrarse los errores triviales primeramente en la fase de prueba de un proyecto?**

15. **¿Qué diferencia hay entre prueba y eliminación de errores?**

16. **¿Por qué es difícil la eliminación de errores en una implantación ascendente?**

17. **¿Qué se entiende por la frase "progresión secuencial" cuando se describe el ciclo de vida de un proyecto?**

18. **¿Cuáles son los dos principales problemas de la progresión secuencial?**

19. **¿Cuáles son las principales diferencias entre el ciclo de vida semiestructurado y el clásico?**

20. **¿Cuáles son las dos principales consecuencias del enfoque de la implantación descendente?**

21. **¿Por qué, en el ciclo de vida semiestructurado, el diseño a menudo involucra trabajo redundante?**

22. **¿Cuáles son las principales diferencias entre los ciclos de vida semi estructurado y estructurado?**

23. Nombre las nueve actividades del ciclo de vida estructurado del proyecto.

24. **¿Quiénes son los tres tipos de personas que proveen de entradas primarias al ciclo de vida del proyecto?**

25. **¿Cuáles son los cinco principales objetivos de la actividad de la encuesta?**

26. **¿Qué es un diagrama de contexto?**

27. **¿Cuál es el principal propósito de la actividad de análisis?**

28. **¿Cuáles son los tipos de modelos producidos por la actividad de análisis?**

29. **¿Cuál es el propósito de la actividad de diseño?**

Desarrollo

1. **Dos sinónimos de metodología:**
   - **Procedimiento**
   - **Enfoque**

2. **¿Cuál es la diferencia entre una herramienta, como se utiliza en este libro, y una metodología?:**
   La herramienta es lo que usamos para hacer el trabajo. Por ejemplo, puede ser un diagrama, un software, una técnica de modelado, etc. Es algo más concreto y puntual.
   La metodología, en cambio, es el modo en que organizamos el trabajo. Es un conjunto de pasos, reglas o prácticas que seguimos para desarrollar el sistema. Podés pensarla como el "plan general" o "forma de trabajo" que guía todo el proyecto.
   Una no reemplaza a la otra, sino que se complementan: la metodología te dice cómo avanzar, y las herramientas te ayudan a ejecutar cada parte de ese avance.

3. **¿Cuáles son los tres principales propósitos del ciclo de vida de un proyecto?:**

El ciclo de vida de un proyecto tiene tres objetivos principales:

- **Definir claramente las actividades del proyecto,** para que todos sepan qué se debe hacer, especialmente útil en organizaciones grandes donde puede haber personal nuevo. Ayuda a que cada quien entienda cómo encajan sus tareas en el todo.
- **Lograr coherencia entre muchos proyectos dentro de una misma organización,** usando un mismo marco de trabajo. Esto facilita el seguimiento y control desde niveles altos de la empresa, evitando confusiones si cada proyecto se hace de forma distinta.
- **Proporcionar puntos de control y revisión,** que permitan evaluar si seguir o no con el proyecto. Esto es clave en proyectos grandes donde no se puede esperar hasta el final para ver si todo salió bien. Así se pueden detectar problemas a tiempo, reasignar recursos o incluso detener un proyecto si ya no tiene sentido.

Aunque el ciclo de vida ordena y da estructura, no reemplaza al administrador del proyecto. La toma de decisiones, la negociación con usuarios o la motivación del equipo siguen siendo tareas humanas y difíciles que ningún ciclo puede evitar.

4. **Precio de tres productos para metodología comerciales ofrecidos por proveedores de software o empresas consultoras:**

Según el libro, existen herramientas comerciales que ayudan a implementar metodologías de desarrollo de sistemas, en la actualidad hay diversas opciones.

1. **Spectrum ERP (Force Intellect)**

Este es un ERP diseñado para pymes manufactureras, ofreciendo módulos como finanzas, inventario, CRM, RRHH, producción y más.

- **Precio:** USD 7,000 por usuario, pago único.
- **Licencia:** Perpetua, sin versión gratuita ni prueba disponible.

2. **NDS ERP Solution (NDS Systems)**

Una solución ERP para distribución, con funcionalidades como procesamiento de pedidos, gestión de inventario y contabilidad integrada.

- **Precio:** USD 2,500 por usuario, pago único.
- **Licencia:** Perpetua.

3. **SAP ERP (SAP)**

Un ERP integral que permite a las organizaciones gestionar y automatizar sus procesos empresariales centrales en un sistema unificado.

- **Precio:** No especificado públicamente; se recomienda contactar al proveedor para obtener una cotización personalizada.
- **Licencia:** Varía según las necesidades del cliente.

El libro menciona que estos productos pueden tener un costo significativo, aunque no indica precios específicos. Se trata de herramientas orientadas a empresas grandes, y su implementación puede implicar también servicios de consultoría y capacitación.

Estas herramientas son ejemplos de cómo una metodología puede ser formalizada a través de software y documentación técnica, en lugar de depender exclusivamente de procesos internos o caseros.

5. **¿Por qué normalmente las organizaciones pequeñas de proceso de datos no usan metodologías formales?**
   No usan metodologías formales por varias razones, principalmente relacionadas con su tamaño, recursos y la naturaleza de sus operaciones:
   - **Implementación costosa:** Las metodologías formales a menudo requieren inversión en software especializado, herramientas de modelado y capacitación del personal. Las pequeñas organizaciones pueden no tener el presupuesto para esto.
   - **Sobrecarga administrativa:** Metodologías detalladas implican una documentación exhaustiva, reuniones frecuentes y procesos rigurosos. Ésto en una organización pequeña podría ser una carga administrativa para el equipo.
   - **Beneficios no evidentes:** Las pequeñas organizaciones pueden no percibir los beneficios inmediatos de invertir tiempo y recursos en la implementación de metodologías formales, especialmente si sus procesos actuales parecen funcionar adecuadamente.

6. **¿Por qué es útil para los administradores nuevos una metodología?**
   Para los administradores nuevos, una metodología resulta útil por varias razones fundamentales.
- **Proporciona una Estructura y Guía Clara:**
  Al ingresar a un rol de liderazgo, los nuevos administradores a menudo se enfrentan a la complejidad de gestionar equipos, proyectos y responsabilidades. Una metodología ofrece un marco de trabajo definido, con pasos, procesos y herramientas específicas , esto establece un camino a seguir, en lugar de tener que inventar la rueda, una metodología probada proporciona un mapa para abordar tareas y desafíos comunes.
- **Facilita el Aprendizaje y el Desarrollo:**
  Las metodologías a menudo vienen acompañadas de documentación, capacitación y mejores prácticas. Esto permite a los nuevos administradores adquirir conocimientos y habilidades esenciales de manera más rápida y eficiente.

7. **¿Por qué es importante tener una metodología en una organización en la que se estén llevando a cabo muchos proyectos diferentes?**
   Es importante tener una metodología bien definida en una organización que gestiona múltiples proyectos diferentes por varias razones.
- **Lenguaje Común:** Una metodología establece un lenguaje y unos procesos comunes para todos los proyectos. Esto facilita la comunicación entre los equipos, los stakeholders y la dirección, evitando malentendidos y confusiones.

- **Optimización de Recursos:** Una metodología ayuda a identificar las mejores prácticas para la asignación y gestión de recursos (humanos, financieros, materiales) en diferentes tipos de proyectos.
- **Identificación Temprana:** Una metodología suele incluir procesos para la identificación y evaluación sistemática de riesgos en cada proyecto.

8. **¿Cómo es que una metodología es útil para controlar proyectos?**

Una metodología es útil para controlar proyectos porque proporciona puntos de control y revisión administrativos que permiten supervisar el avance del proyecto en etapas clave. Esto hace posible detener, revisar o redirigir un proyecto en caso de que no se esté cumpliendo con los objetivos establecidos o si surge algún problema grave.
Además, al definir de forma clara las actividades a realizar, su secuencia y responsables, permite que cada miembro del equipo entienda qué hacer y cuándo, lo cual facilita la coordinación general. Esto es especialmente importante en organizaciones grandes, donde puede haber múltiples proyectos en paralelo o personal nuevo.
El uso de un ciclo de vida estructurado también organiza las actividades del administrador, aumentando la probabilidad de que se aborden los problemas pertinentes en el momento adecuado, y proporciona una base para la toma de decisiones informadas.

9. **¿Cuáles son las principales características que distinguen el ciclo de vida clásico?**

   Las principales características del ciclo de vida clásico son su naturaleza secuencial, su enfoque en la planificación y la documentación exhaustiva desde el inicio, y su limitada flexibilidad para incorporar cambios una vez que el proyecto ha avanzado.
   Nada está hecho hasta que todo esté terminado. Por eso, si el proyecto se atrasa y la fecha límite cae precisamente en medio del proceso de prueba del sistema, no habrá nada que mostrarle al usuario más que una enorme pila de listados de programas, los cuales, vistos en su totalidad, no le ofrecen nada de valor.
   Las fallas más triviales se encuentran al comienzo del periodo de prueba y las más graves al final. Esto es casi una tautología: las pruebas modulares dejan al descubierto fallas relativamente simples dentro de los módulos individuales. Las pruebas del sistema, por otra parte, descubren errores grandes de interfaz entre subsistemas. La cuestión es que los errores de interfaz no son lo que el programador desea descubrir al final de un proyecto de desarrollo; tales fallas pueden obligar a la recodificación de un gran número de módulos, y pueden tener un impacto devastador sobre el calendario, justo en un momento en el cual es probable que todo el mundo esté algo cansado y molesto tras haber trabajado duro durante tantos meses.
   La eliminación de fallas suele ser extremadamente difícil durante las últimas etapas de prueba del sistema. Nótese que se puede distinguir entre pruebas y eliminación de fallas. La eliminación de

fallas es el arte de descubrir dónde está la falla (y subsecuentemente cómo arreglaría) después de que el proceso de prueba ha determinado que la falla de hecho existe. Cuando la falla se descubre durante la fase de prueba del sistema en un proyecto ascendente, a menudo suele ser extremadamente difícil determinar cuál módulo la contiene; pudiera tratarse de cualquiera de los cientos (o miles) de módulos que se han combinado por primera vez. Localizar una falla a menudo es como hallar una aguja en un pajar. La necesidad de prueba con la computadora aumenta exponencialmente durante las etapas finales de prueba. Para ser más específicos, el administrador del proyecto a menudo descubre que necesita una gran cantidad de horas-máquina para probar el sistema; tal vez 12 horas de labor ininterrumpida diaria. Dado que suele ser difícil obtener tanto tiempo de uso de la computadora, el proyecto suele retrasarse mucho.

10. ¿Qué significa la expresión implantación ascendente?

La implantación ascendente es una característica del ciclo de vida clásico de desarrollo de sistemas, donde el sistema se construye y prueba desde los módulos más pequeños hacia los más grandes. **Primero se realizan pruebas modulares, luego pruebas de subsistemas, y finalmente pruebas del sistema completo.** Este enfoque también se conoce como **"modelo en cascada"**, y se inspira en procesos de fabricación como las líneas de montaje.

11. ¿Cuáles son las cuatro principales dificultades con la estrategia de implantación ascendente?

Podrían ser:

- **Descubrimiento tardío de errores graves:**
  Las fallas más simples se detectan en las primeras pruebas modulares, pero los errores grandes —especialmente los de interfaz entre subsistemas— suelen aparecer recién al final del proyecto, cuando es más difícil y costoso corregirlos.
- **Dificultad para localizar las fallas:**
  Cuando aparece un error en la fase final, ya se han integrado cientos o miles de módulos por primera vez. Encontrar cuál de todos causa la falla puede ser como buscar una aguja en un pajar, lo que complica enormemente el trabajo.
- **Demanda excesiva de tiempo de prueba:**
  Durante las etapas finales, la cantidad de horas-máquina necesarias para probar el sistema aumenta de forma exponencial. Como es difícil conseguir tanto tiempo de uso de la computadora, esto provoca importantes retrasos.
- **Nada está listo hasta que todo esté terminado:**
  En este enfoque, el sistema completo solo puede presentarse al final. Si el proyecto se atrasa, no hay una versión funcional para mostrar al usuario, solo una pila de listados de código que no ofrecen valor práctico hasta que todo esté integrado y probado.

12. ¿Qué tipo de ambiente es el adecuado para un enfoque de implantación ascendente?

El enfoque de implantación ascendente es más adecuado en entornos donde el sistema que se está desarrollando **ya está bien definido y probado**, como por ejemplo en procesos repetitivos tipo fábricas o producción en serie.

No es ideal para proyectos nuevos o con muchos cambios, porque en esos casos pueden aparecer errores de integración que arruinan todo al final.

"La implantación ascendente es buena para el montaje de automóviles en línea, ¡pero sólo después de que el prototipo esté completamente libre de fallas!"

13. ¿Por qué es importante que 'nada está hecho hasta que todo esté hecho', que además es lo que caracteriza al enfoque ascendente?

Porque en este tipo de enfoque, **no hay entregables parciales que funcionen**. El sistema solo empieza a ser útil cuando **todas las piezas están terminadas y ensambladas correctamente**. Eso quiere decir que, si el proyecto se atrasa, el usuario no va a ver nada tangible ni funcional hasta el final. Es como construir un auto y recién poder probarlo cuando le pusiste hasta la última tuerca.

Esta forma de trabajo es riesgosa porque **si algo falla al final, puede arruinar todo lo anterior**, y muchas veces no hay margen para resolverlo sin grandes retrabajos o atrasos importantes.

"Nada está hecho hasta que todo esté terminado. Por eso, si el proyecto se atrasa y la fecha límite cae precisamente en medio del proceso de prueba del sistema, no habrá nada que mostrarle al usuario más que una enorme pila de listados de programas"

Así que esa frase resume perfecto lo que lo hace tan limitante: **no podés mostrar progreso real hasta el final**, aunque hayas trabajado un montón.

14. Por qué debieran encontrarse los errores triviales primeramente en la fase de prueba de un proyecto?

Porque al inicio del proceso de pruebas se testean los **módulos individuales**, que son partes más pequeñas del sistema. Ahí es donde suelen aparecer los errores simples, como variables mal usadas, cálculos erróneos o lógica básica mal implementada. Son cosas fáciles de detectar porque se analiza cada módulo por separado.

En cambio, los errores más graves, como los que tienen que ver con la **comunicación entre módulos** o el funcionamiento completo del sistema, **recién aparecen al final**, cuando todo está integrado. Por eso es clave que los errores más simples se identifiquen primero, para que no se acumulen y compliquen el análisis final.

> *"Las fallas más triviales se encuentran al comienzo del periodo de prueba y las más graves al final. Las pruebas modulares dejan al descubierto fallas relativamente simples dentro de los módulos individuales."*

Entonces, encontrar primero los errores chicos **ahorra tiempo y esfuerzo más adelante**, porque el equipo puede enfocarse en los problemas grandes

sin tener que perder tiempo arreglando bugs básicos que deberían haber estado resueltos desde el principio.

15. ¿Qué diferencia hay entre prueba y eliminación de errores?

La prueba es el proceso de **verificar si algo está funcionando como se espera**, es decir, ejecutar el sistema (o una parte) y observar si hay errores. Si aparece un fallo, lo único que se sabe es que **algo anda mal**, pero **no se sabe dónde ni por qué**.
En cambio, la eliminación de errores es lo que viene después: implica **buscar el origen exacto del problema**, entenderlo y corregirlo. Y eso puede llevar mucho tiempo, sobre todo si el sistema es grande y complejo. Muchas veces, **detectar el error es fácil**, pero **eliminarlo es un lío bárbaro**, especialmente si el módulo donde está el error está conectado con otros cien.

> *"La eliminación de fallas es el arte de descubrir dónde está la falla (y subsecuentemente cómo arreglarla) después de que el proceso de prueba ha determinado que la falla de hecho existe."*

O sea, hacer pruebas es como notar que algo no anda; eliminar errores es **descubrir el por qué y arreglarlo**, que muchas veces es un dolor de cabeza.

16. ¿Por qué es difícil la eliminación de errores en una implantación ascendente?

En una **implantación ascendente**, la eliminación de errores es difícil por varias razones:

1. **Los errores más graves se detectan al final del proceso:**
   Las pruebas modulares detectan fallas simples en módulos individuales, pero las pruebas del sistema revelan errores de **interfaz entre subsistemas**, que son más complejos y críticos. Encontrarlos al final implica que ya se ha invertido mucho esfuerzo, y corregirlos puede afectar a distintas partes del sistema.
2. **Localizar la falla es como buscar una aguja en un pajar:**
   Cuando el sistema está integrado, hay miles de módulos funcionando juntos. Si aparece un error, **es muy difícil identificar en cuál módulo específico se encuentra**.
3. **Se requiere mucho tiempo de máquina para las pruebas finales:**
   La necesidad de prueba con la computadora **aumenta exponencialmente** en las etapas finales. Es posible que se necesiten muchas horas continuas de uso de la computadora para probar todo el sistema, lo cual puede provocar retrasos.

17. ¿Qué se entiende por la frase "progresión secuencial" cuando se describe el ciclo de vida de un proyecto?

Cuando hablamos del ciclo de vida de un proyecto, entendemos por “progresión secuencial” que cada fase o etapa del proyecto debe ser completada antes de avanzar a la siguiente. Es una forma ordenada y lineal de avanzar con un proyecto.

18. ¿Cuáles son los dos principales problemas de la progresión secuencial?

Los dos principales problemas de la progresión secuencial son:

1. **No se adapta a la realidad:** Se asume que todo saldrá bien desde el principio, pero en la práctica es muy distinto, ya que las personas cometen errores, los usuarios pueden cambiar de opinión, y el entorno puede sufrir cambios.
2. **Produce una falsa sensación de avance:** Muchas veces se pretende que al terminar una fase ya no habrá que volver atrás, llegando a "congelar" los documentos para que no se modifiquen más. Esto es poco realista, porque pueden surgir errores o descubrirse nuevas necesidades más adelante, complicando el modificar algo que ya se dio por completado.

19. ¿Cuáles son las principales diferencias entre el ciclo de vida semiestructurado y el clásico?

Las principales diferencias entre el ciclo de vida semiestructurado y el clásico son:

1. **Dirección de la implantación**

- **Clásico:** usa implantación ascendente (de abajo hacia arriba).
  Primero se codifican y prueban los módulos más pequeños y luego se van uniendo en subsistemas y finalmente en el sistema completo
- **Semiestructurado:** usa implantación descendente (de arriba hacia abajo).
  Primero se codifican y prueban los módulos de alto nivel, y luego los más detallados. Esto permite retroalimentación más temprana y detecta errores antes.

2. **Técnicas utilizadas**

- **Clásico:** tiende a ignorar técnicas modernas como análisis estructurado o programación estructurada.
- **Semiestructurado:** incorpora esas técnicas modernas, como el diseño estructurado y la programación estructurada, lo que mejora la claridad y la calidad del sistema.

3. **Flujo del trabajo**

- **Clásico:** sigue una secuencia estrictamente lineal, una fase termina antes de empezar la siguiente.
- **Semiestructurado:** permite más flexibilidad y retroalimentación entre actividades. Algunas tareas se pueden hacer en paralelo o solaparse.

4. **Interacción con el usuario**

- **Clásico:** el usuario participa poco después de las primeras etapas, lo que puede llevar a malentendidos.
- **Semiestructurado:** aunque no lo hace perfecto, favorece mayor retroalimentación del usuario, sobre todo durante la implantación.

5. **Corrección de errores**

- **Clásico:** al detectar errores tarde, suele ser más costoso y difícil corregirlos.
- **Semiestructurado:** al probar módulos de alto nivel primero, es más fácil detectar y corregir errores temprano.

20. ¿Cuáles son las dos principales consecuencias del enfoque de la implantación descendente?

Las dos principales consecuencias del enfoque de la implantación descendente son:

1. **Retroalimentación entre etapas del desarrollo:**
   A diferencia del enfoque clásico (ascendente), la implantación descendente permite que actividades como codificación, pruebas y eliminación de fallas se realicen en paralelo, lo cual genera una retroalimentación más fluida entre estas etapas. Esta retroalimentación puede incluso hacer que se revisen decisiones tomadas previamente en el análisis o diseño.
2. **Tentación de modificar especificaciones sin control formal:**
   Al permitir que los desarrolladores interactúen directamente con los usuarios mientras aún están codificando, se crea la posibilidad de hacer cambios a la especificación sin que pasen por un proceso de revisión o autorización formal. Esto puede ser bueno en términos de flexibilidad y adaptación, pero también puede ser perjudicial si no se maneja adecuadamente, ya que el administrador del proyecto podría ni enterarse de las modificaciones realizadas.

21. ¿Por qué, en el ciclo de vida semiestructurado, el diseño a menudo involucra trabajo redundante?

En el ciclo de vida semiestructurado, el diseño a menudo involucra trabajo redundante porque gran parte del trabajo del diseño estructurado se dedica a corregir o reinterpretar especificaciones erróneas, ambiguas o incompletas que se hicieron en etapas anteriores, especialmente durante el análisis.

¿Por qué pasa esto?

- Las especificaciones suelen estar escritas de forma narrativa, con ambigüedades, redundancias y falta de estructura formal.
- Quienes realizan el diseño muchas veces no tuvieron contacto directo con los analistas que redactaron esas especificaciones, ni con los usuarios del sistema.
- Por lo tanto, los diseñadores tienen que traducir ese documento poco claro en un conjunto de herramientas técnicas útiles: diagramas de flujo de datos, diccionarios de datos, diagramas entidad-relación, etc.

22. ¿Cuáles son las principales diferencias entre los ciclos de vida semi estructurado y estructurado?

El ciclo de vida semi estructurado reemplaza la codificación y pruebas ascendentes por una implantación descendente. Comienza a introducir conceptos como la programación estructurada y el diseño estructurado, lo cual mejora la organización del desarrollo, pero sigue siendo un proceso principalmente secuencial.

Por otro lado, el ciclo de vida estructurado va un paso más allá al permitir que varias actividades puedan realizarse en paralelo (flujo no secuencial), y se apoya en modelos formales como los diagramas de flujo de datos y los diagramas entidad-relación. Además, establece una retroalimentación constante entre las fases, permitiendo mejorar el

sistema mientras se desarrolla. Define nueve actividades específicas que guían todo el proceso, desde la encuesta inicial hasta la instalación.

23. Las nueve actividades del ciclo de vida estructurado del proyecto:

1) **Encuesta**
2) **Análisis de sistemas**
3) **Diseño**
4) **Implantación**
5) **Generación de pruebas de aceptación**
6) **Garantía de calidad**
7) **Descripción del procedimiento**
8) **Conversión de bases de datos**
9) **Instalación**

24. ¿Quiénes son los tres tipos de personas que proveen de entradas primarias al ciclo de vida del proyecto?

Los tres tipos principales de personas que participan en el ciclo de vida son:

- **Usuarios**
- **Administradores**
- **Personal de operaciones**

Estos grupos se consideran **"terminadores"**, y son las fuentes externas de información que alimentan al sistema y participan activamente en las decisiones, requerimientos y validación del mismo.

25. ¿Cuáles son los cinco principales objetivos de la actividad de la encuesta?

- **Identificar a los usuarios responsables y crear un campo de actividad inicial.**
- **Identificar las deficiencias actuales en el ambiente del usuario.**
- **Establecer metas y objetivos para un nuevo sistema.**
- **Determinar si es factible automatizar el sistema y sugerir escenarios aceptables.**
- **Preparar un esquema general para guiar el resto del proyecto.**

26. ¿Qué es un diagrama de contexto?

Es un diagrama de flujo de datos simplificado que representa todo el sistema como un único proceso, y muestra su relación con los terminadores (usuarios, otras unidades organizacionales, u otros sistemas). Su propósito principal es definir los límites del sistema, es decir, qué incluye y qué queda fuera de su alcance.

27. ¿Cuál es el principal propósito de la actividad de análisis?

Transformar las necesidades del usuario y el esquema general del proyecto en una especificación formal y estructurada que describa de forma precisa lo que el nuevo sistema debe hacer, sin importar la tecnología que se utilizará para implementarlo. Esta especificación es el puente entre lo que se necesita y lo que se va a construir.

28. ¿Cuáles son los tipos de modelos producidos por la actividad de análisis?

- **Modelo ambiental:** describe el ambiente actual del usuario, mostrando cómo funciona el sistema existente.
- **Modelo de comportamiento:** representa cómo debería comportarse el nuevo sistema.

Ambos modelos se combinan para formar el **modelo esencial**, que refleja la esencia del sistema a desarrollar, sin condicionamientos tecnológicos.

29. ¿Cuál es el propósito de la actividad de diseño?

- **Asignar partes de la especificación a diferentes procesadores (humanos o máquinas)**
- **Definir la jerarquía de módulos**
- **Diseñar las interfaces entre ellos**
- **Transformar los modelos de datos en un diseño de base de datos.**

Además, se construye el **modelo de implantación del usuario**, que permite visualizar cómo se integrará el sistema en el entorno real.