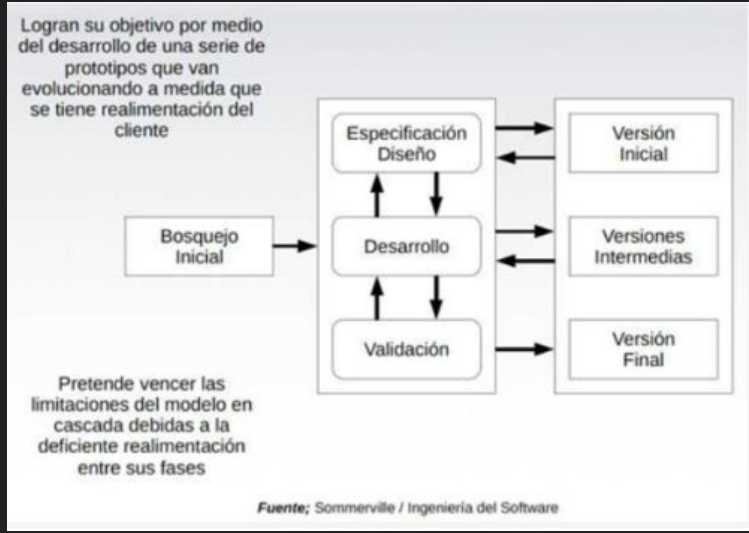
# Desarrollo Evolutivo

# Desarrollo evolutivo

El desarrollo evolutivo consta del desarrollo de una versión inicial que luego de exponerse se va refinando de acuerdo a los comentarios o nuevos requerimientos por parte del cliente o del usuario final. Las fases de especificación, desarrollo y validación se entrelazan en vez de separarse.

Está enfocado a la adaptación y refinamiento del desarrollo de un sistema y el **desconocimiento de la problemática específica**.



Logran su objetivo por medio del desarrollo de una serie de prototipos que van evolucionando a medida que se tiene realimentación del cliente


Pretende vencer las limitaciones del modelo en cascada debidas a la deficiente realimentación entre sus fases

***Fuente;*** Sommerville / Ingeniería del Software

# Tipos de desarrollo evolutivo

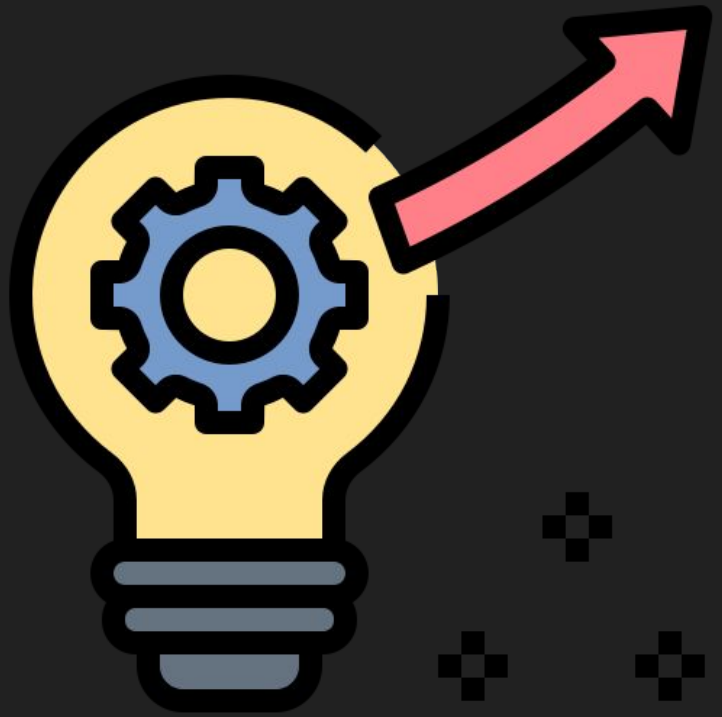
Existen dos tipos de desarrollo evolutivo:

- **Desarrollo exploratorio**, donde el objetivo del proceso es trabajar con el cliente para explorar sus requerimientos y entregar un sistema final. El desarrollo empieza con las partes del sistema que se comprenden mejor. El sistema evoluciona agregando nuevos atributos propuestos por el cliente.
- **Prototipos desechables**, donde el objetivo del proceso de desarrollo evolutivo es comprender los requerimientos del cliente y entonces desarrollar una definición mejorada de los requerimientos para el sistema. El prototipo se centra en experimentar con los requerimientos del cliente que no se comprenden del todo.

# Desarrollo evolutivo

Desde el punto de vista de desarrollo de sistema el enfoque evolutivo suele traer más ventajas en comparación con un enfoque en pesado o rígido ya que el sistema se va ajustando a las necesidades del cliente, a la vez que él mismo entiende mejor sus propios requerimientos.

Sin embargo, el enfoque evolutivo desde una perspectiva de ingeniería y gestión suele tener dos grandes problemas



# Problemas con desarrollo evolutivo

El desarrollo evolutivo es una metodología de desarrollo de software muy relacionada con, pero claramente distinta de, desarrollo por prototipos. El énfasis está puesto sobre la importancia de obtener un sistema de producción flexible y expandible. Así, si los requerimientos cambian durante el desarrollo del sistema, entonces con un mínimo de esfuerzo y tiempo se puede desarrollar un sistema de trabajo flexible.

La diferencia fundamental entre desarrollo evolutivo y prototipos de software es que el desarrollo evolutivo busca reemplazar el viejo sistema con uno nuevo que tendría la propiedad de satisfacer los nuevos requerimientos lo más rápido posible.

En contraste, prototipos usa un enfoque iterativo solo para determinar los requerimientos organizacionales. Por lo tanto, el tiempo tomado entre cada “iteración” puede ser mucho más extenso para el desarrollo evolutivo.

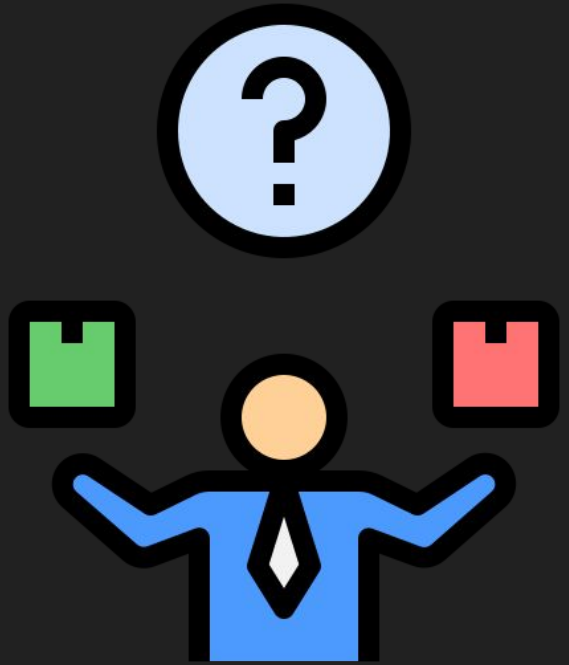
# Problemas con desarrollo evolutivo

El desarrollo evolutivo asume que los requerimientos de un proyecto están sujetos a cambios continuos, por lo cual es necesario definir una estrategia de desarrollo que refleje esta situación. En cambio, el desarrollo orientado a prototipos, así como los anteriores, asume que los requerimientos "reales" existen y se vale de las iteraciones del prototipo para establecerlos y modelarlos.

La idea entonces de la metodología de desarrollo evolutivo es estar liberando constantemente una nueva versión del sistema que sea completamente funcional; así, cada sistema producto de las iteraciones sucesivas del método tendría incorporado los nuevos requerimientos que ha sido posible identificar y que no estarían considerados en la anterior versión.

# Diferencias/similitudes entre desarrollo evolutivo e iterativo

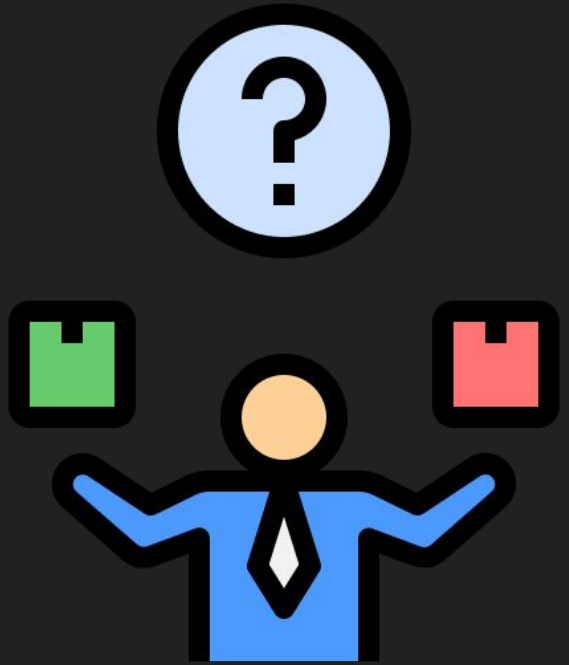
El desarrollo evolutivo se centra en la construcción gradual de un sistema, mientras que el desarrollo iterativo se enfoca en la repetición de ciclos de desarrollo para mejorar y adaptar el sistema a medida que avanza. Las diferencias principales incluyen:



- Enfoque de construcción:
  - Evolutivo: Construcción gradual y continua del sistema en etapas sucesivas.
  - Iterativo: Ciclos repetidos de diseño, desarrollo, prueba y evaluación.
- Manejo de cambios en requisitos:
  - Iterativo: Funciona mejor con requisitos relativamente estables, ya que los cambios pueden ser costosos.
  - Evolutivo: Altamente adaptable a cambios en los requisitos, con ajustes en cada iteración.

# Diferencias/similitudes entre desarrollo evolutivo e iterativo

- Flexibilidad
  - Iterativo: Menos flexible para cambios significativos, ya que las etapas posteriores dependen de las anteriores.
  - Evolutivo: Mayor flexibilidad para acomodar cambios (significativos) en cualquier momento.
- Entrega de versiones funcionales
  - Evolutivo: Se entregan versiones funcionales del sistema al final de cada etapa.
  - Iterativo: Entrega versiones funcionales del sistema al final de cada iteración.
- Riesgo de entrega final
  - Evolutivo: Mayor riesgo en la entrega final debido a la acumulación de problemas no detectados.
  - Iterativo: Menos riesgo en la entrega final, ya que los problemas se descubren y resuelven en etapas tempranas.



# Ventajas y desventajas

Ventajas:

Es interactivo: Con cada incremento, el cliente recibe un producto operacional.

Es robusto a la hora de gestionar recursos técnicos. (puede responder a los cambios en herramientas, tecnologías, y recursos humanos disponibles gracias a su naturaleza iterativa).

Desventajas:

Puede ser difícil ajustar los requisitos a los incrementos del sistema.

Se usa en:

Proyectos donde el entorno de aplicación es cambiante (mercado, programa científico, contexto, entidad donde se aplique).

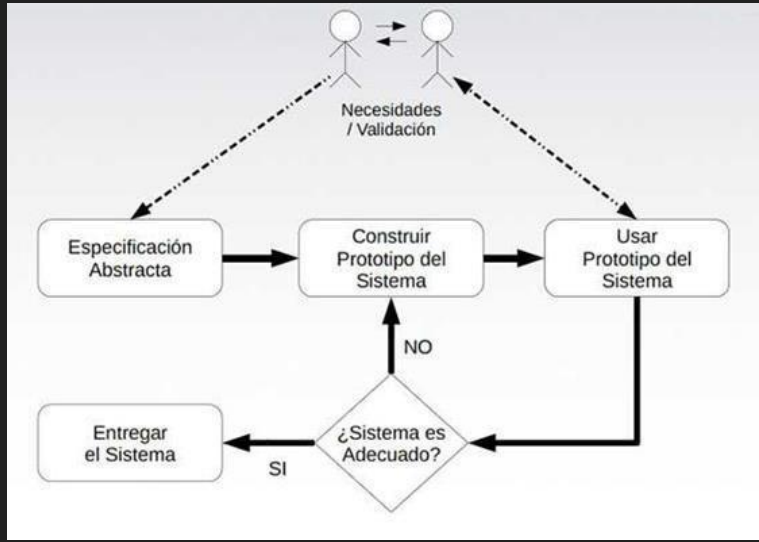
# Desarrollo por prototipos

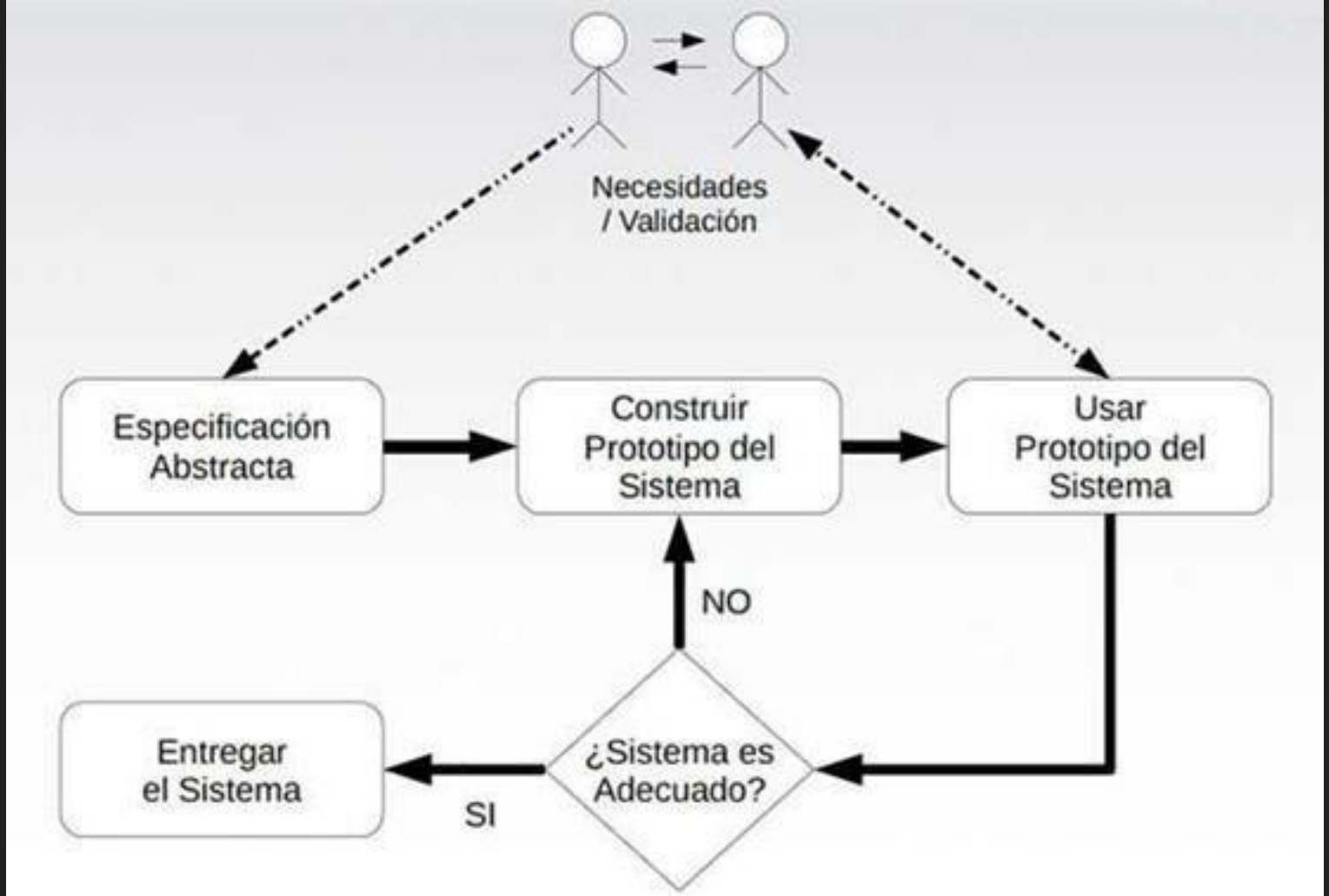
# Desarrollo por prototipos

El desarrollo evolutivo asume que los requerimientos de un proyecto están sujetos a cambios continuos, por lo cual es necesario definir una estrategia de desarrollo que refleje esta situación.

En cambio, el desarrollo orientado a prototipos, así como los anteriores, asume que los requerimientos "reales" existen y se vale de las iteraciones del prototipo para establecerlos y modelarlos.

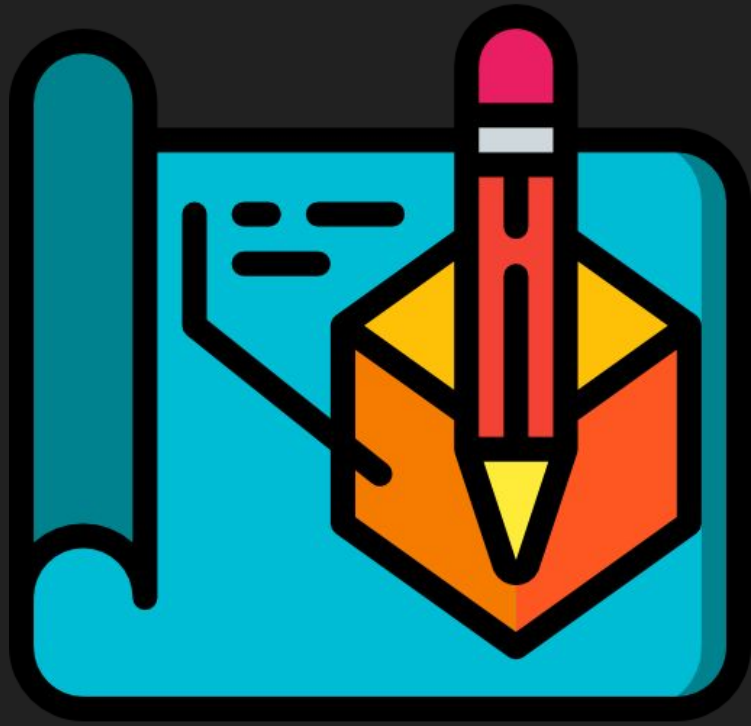
Un enfoque de desarrollo orientado a prototipos está orientado a: reducción de la incertidumbre y del riesgo, reducción de tiempo y de costos, incrementos en la aceptación del nuevo sistema, mejoras en la administración de proyectos, mejoras en la comunicación entre desarrolladores y clientes, etc.




Necesidades
/ Validación
Especificación
Abstracta
Construir
Prototipo del
Sistema
Usar
Prototipo del
Sistema
NO
Entregar
el Sistema
¿Sistema es
Adecuado?
SI

# Desarrollo por prototipos

No se puede desconocer que la fase de definición de requerimientos se ha perfeccionado en dos aspectos importantes: primero se ha aproximado las visiones del usuario y el desarrollador, lo cual representa el beneficio de establecer una base común de comunicación; también, el hacer explícito la posibilidad de iterar sobre estos dominios permitiría que la convergencia de los mismos sea una posibilidad cierta.



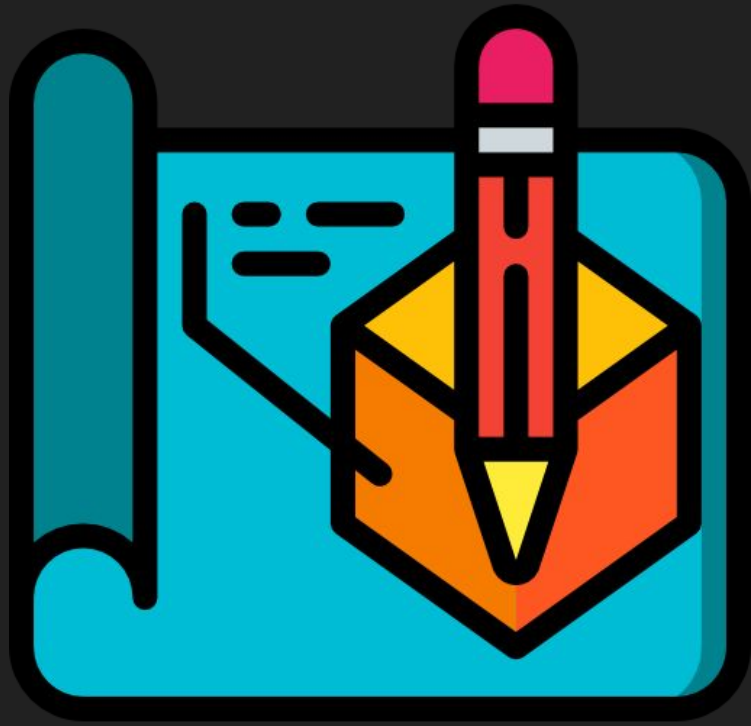
Pero lo anterior no asegura el éxito, por ejemplo, qué certeza existe en que esta iteración sea en la dirección correcta y lleve a la convergencia de los dominios; no se puede desconocer las diferencias que existen entre la prueba de un prototipo de software en la fase de definición de requerimientos y el uso del mismo ya como un producto terminado. Para explicar esto, podemos hablar de **dos dominios en el usuario**, uno que es el que se establece cuando se **prueba el prototipo** y otro, distinto, por cierto, el que ocurre cuando el usuario hace **uso del software en ambiente de explotación**.

# Desarrollo por prototipos

Por último, el proceso de iteración, para que sea efectivo, debería ser infinito, lo que lo hace poco efectivo. Es decir, mediante este método acercamos la problemática de los usuarios a los dominios de los desarrolladores y viceversa, pero no sería posible lograr un apareamiento uno a uno entre estos dominios, lo que sería el ideal.

Es así que este modelo subraya las fuentes de requisitos para el producto, puntos decisivos de continuar/detenerse, y el uso de prototipos.

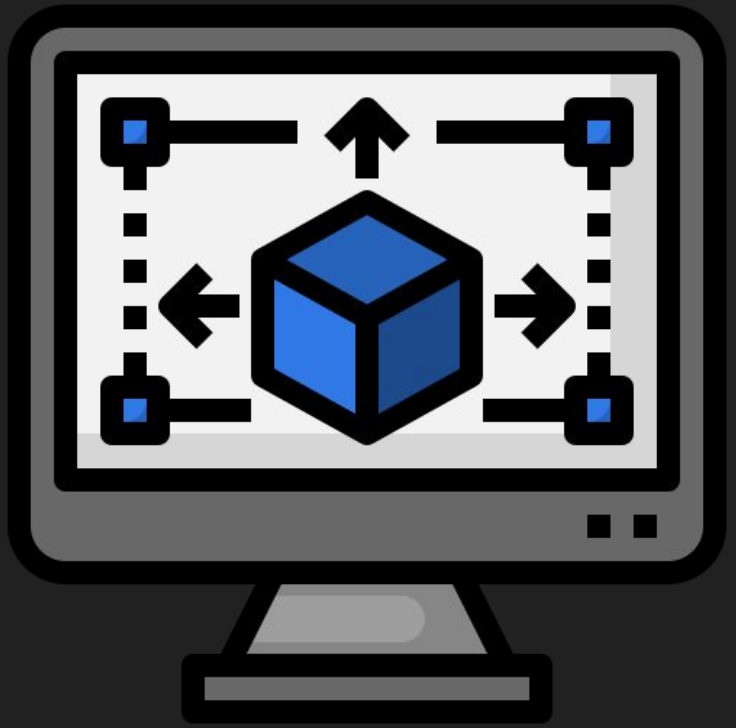
Un prototipo es una representación o modelo del producto de programación que, a diferencia de un modelo de simulación, incorpora componentes del producto real. Por lo regular, un prototipo tiene un funcionamiento limitado en cuanta a capacidades, confiabilidad o eficiencia.



# ¿Por qué desarrollar un prototipo?

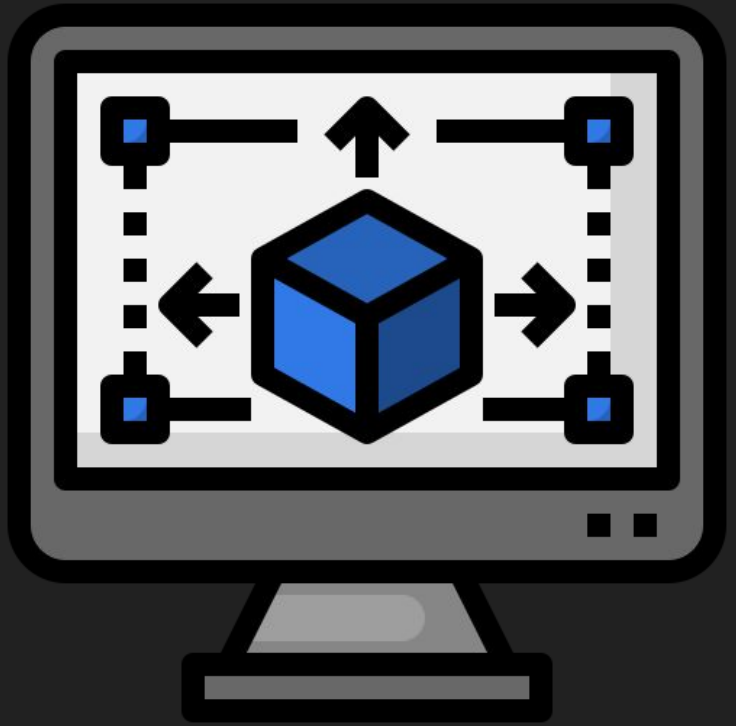
Hay varias razones para desarrollar un prototipo; una de ellas es ilustrar los formatos de datos de entrada, mensajes, informes y diálogos al cliente, este es un mecanismo adecuado para explicar opciones de procesamiento y tener un mejor entendimiento de las necesidades de él.

La segunda razón es para explorar aspectos técnicos del producto propuesto. Con frecuencia, una decisión importante del diseño dependerá., por ejemplo, del tiempo de respuesta del controlador de un dispositivo o de la eficiencia de un algoritmo de clasificación; en tales casos, un prototipo puede ser la mejor o única manera de resolver el problema.



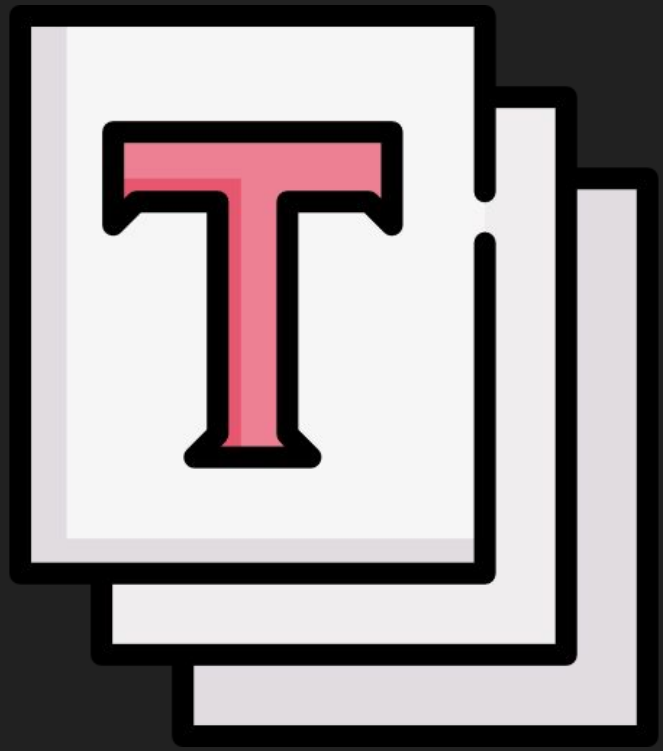
# ¿Por qué desarrollar un prototipo?

La tercera razón para desarrollar un prototipo se da cuando el modelo de fases análisis -> diseño -> instrumentación es inapropiado. El modelo de fases se aplica cuando se puede redactar un conjunto razonablemente completo de especificaciones al inicio del ciclo de vida. Algunas veces no es posible definir el producto sin un desarrollo exploratorio, y en ocasiones no es claro cómo proceder a la mejora del sistema hasta que no se instrumenta y evalúa una versión. El desarrollo exploratorio se utiliza para desarrollar algoritmos para, por ejemplo, jugar ajedrez, para resolver problemas confusos, y para llevar a cabo tareas que requieren la simulación del comportamiento humano; sin embargo, esta técnica no se limita a estas situaciones.



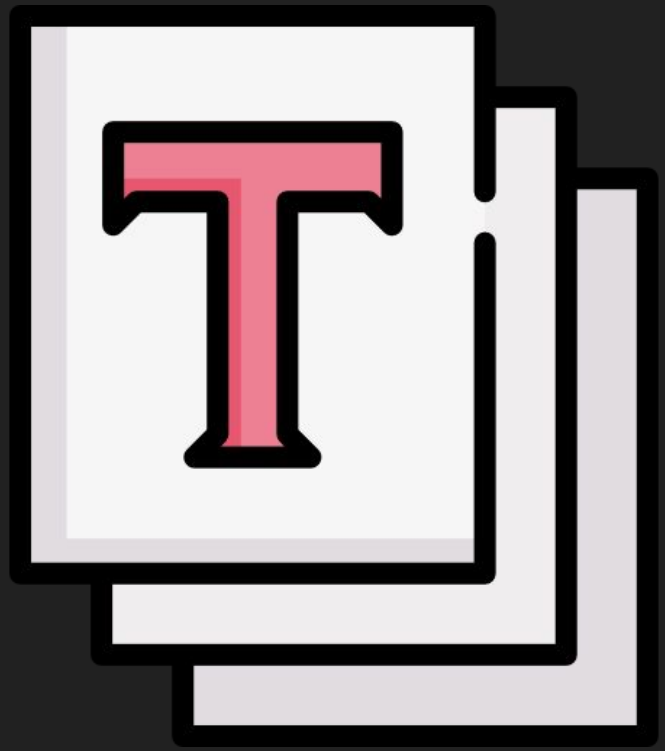
# Tipos de modelo de prototipo

- **Prototipo rápido:** Metodología de diseño que desarrolla rápidamente nuevos diseños, los evalúa y prescinde del prototipo cuando el próximo diseño es desarrollado mediante un nuevo prototipo.
- **Prototipo reutilizable:** También conocido como "Evolutionary Prototyping"; no se pierde el esfuerzo efectuado en la construcción del prototipo pues sus partes o el conjunto pueden ser utilizados para construir el producto real. Mayormente es utilizado en el desarrollo de software, si bien determinados productos de hardware pueden hacer uso del prototipo como la base del diseño de moldes en la fabricación con plásticos o en el diseño de carrocerías de automóviles.
- **Prototipo Modular:** También conocido como Prototipado Incremental (Incremental prototyping); se añaden nuevos elementos sobre el prototipo a medida que el ciclo de diseño progresa.



# Tipos de modelo de prototipo

- **Prototipo Horizontal**: El prototipo cubre un amplio número de aspectos y funciones, pero la mayoría no son operativas. Resulta muy útil para evaluar el alcance del producto, pero no su uso real.
- **Prototipo Vertical:** El prototipo cubre sólo un pequeño número de funciones operativas. Resulta muy útil para evaluar el uso real sobre una pequeña parte del producto.
- **Prototipos de Baja-fidelidad:** El prototipo se implementa con papel y lápiz, emulando la función del producto real sin mostrar el aspecto real del mismo. Resulta muy útil para realizar test baratos.
- **Prototipos de Alta-fidelidad:** El prototipo se implementa de la forma más cercana posible al diseño real en términos de aspecto, impresiones, interacción y tiempo.

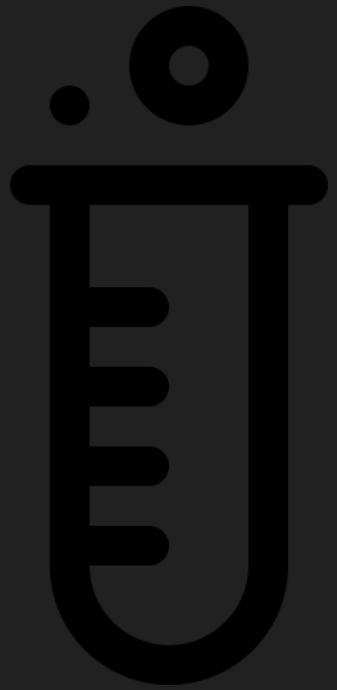


# Claves para que el prototipo sea efectivo

- Debe ser un sistema con el que se pueda experimentar
- Debe ser comparativamente barato
- Debe desarrollarse rápidamente
- Énfasis en la interfaz de usuario
- Equipo de desarrollo reducido
- Herramientas y lenguajes adecuadas

# Sistema con el que se pueda experimentar

La clave para que un prototipo sea efectivo en el sistema por prototipos es que debe ser un sistema con el que se pueda experimentar porque permite a los usuarios y las partes interesadas interactuar directamente con el prototipo, lo que facilita la identificación de problemas, la validación de conceptos y la alineación de expectativas de manera más efectiva.



# Comparativamente barato

la inversión en la construcción del prototipo inicial debe ser proporcionalmente menor en términos de tiempo y recursos financieros en comparación con el desarrollo completo del sistema. Esto permite que las organizaciones experimenten con ideas y conceptos de manera asequible, sin incurrir en costos significativos antes de validar la viabilidad de un proyecto.

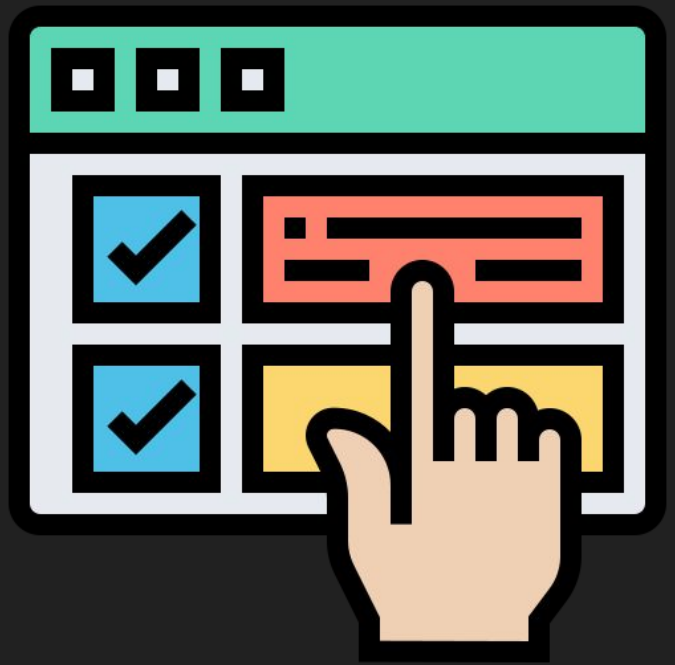
# Rápido desarrollo

La velocidad en la creación del prototipo es esencial para obtener resultados ágiles y oportunidades de retroalimentación temprana. Un prototipo que se desarrolla rápidamente permite a los diseñadores, desarrolladores y usuarios explorar conceptos y funcionalidades de manera oportuna, lo que acelera el proceso de toma de decisiones y ajustes en el diseño del sistema final.

La falta de desarrollo rápido en un prototipo puede llevar a retrasos, costos adicionales, problemas de implementación y desalineación de expectativas. La velocidad en la creación del prototipo es esencial para aprovechar al máximo las oportunidades de retroalimentación temprana y mantener un proceso de desarrollo ágil y eficiente.

# Énfasis en la interfaz de usuario

Al centrarse en la interfaz de usuario desde el principio, se asegura que esta sea intuitiva, eficiente y satisfactoria para los usuarios finales. Esto facilita la validación temprana de la usabilidad y la identificación de posibles problemas de diseño antes de que se desarrolle el sistema completo.



# Equipos de desarrollo reducido

Un equipo más pequeño permite una comunicación más directa y eficiente entre los miembros. Esto facilita la toma de decisiones rápida y la implementación ágil de cambios en el prototipo. Un equipo reducido también minimiza la burocracia y la complejidad, lo que acelera el proceso de desarrollo y permite un enfoque más centrado en la experimentación y la retroalimentación.

# Herramientas y lenguajes adecuadas

Deben utilizarse herramientas y lenguajes adecuados porque la elección correcta de estas herramientas permite una construcción eficiente y rápida del prototipo, lo que a su vez facilita la iteración y la adaptación continua del diseño. Las herramientas y lenguajes adecuados ayudan a los desarrolladores a implementar funcionalidades de manera efectiva, lo que permite que el prototipo se asemeje de cerca al sistema final y proporcione una experiencia realista para los usuarios y las partes interesadas. Esto, a su vez, mejora la calidad de la retroalimentación y la toma de decisiones durante el proceso de desarrollo del sistema final, es decir, más de lo mismo.

# Ventajas del modelo de prototipo

Utilizar prototipos puede mejorar la calidad de los requisitos y especificaciones proporcionados a los desarrolladores. Dado que los cambios cuestan exponencialmente más en su implementación a medida que se detectan más tarde en el desarrollo, la determinación temprana de lo que realmente desea el usuario puede resultar en un software más rápido y menos costoso.

El modelo de prototipos requiere la participación del usuario y les permite a los mismos ver e interactuar con un prototipo, lo que les permite brindar una retroalimentación y especificaciones mejores y más completas. El prototipo examinado por el usuario previene muchas confusiones y malentendidos que ocurren cuando cada parte “cree que la otra entiende lo que dijo”. Dado que los usuarios conocen el dominio del problema mejor que cualquier miembro del equipo de desarrollo, una mayor interacción puede resultar en un producto final con una mayor calidad (tangible e intangible). El producto final tiene más probabilidades de satisfacer el deseo del usuario en términos de apariencia, sensación y rendimiento.

# Desventajas del modelo de prototipo

**Análisis insuficiente**: El enfoque en un prototipo limitado puede distraer a los desarrolladores de analizar adecuadamente el proyecto completo. Esto puede llevar a pasar por alto soluciones mejores, la preparación de especificaciones incompletas, o la conversión de prototipos limitados en proyectos finales mal diseñados que son difíciles de mantener. Además, dado que un prototipo tiene funcionalidad limitada, es posible que no se escale bien si se utiliza como base para un entregable final, lo que puede pasar desapercibido si los desarrolladores están demasiado centrados en construir un prototipo como modelo.

# Desventajas del modelo de prototipo

**Confusión del usuario entre el prototipo y el sistema final**: Los usuarios pueden empezar a pensar que un prototipo, que está destinado a ser descartado, es en realidad un sistema final que solo necesita ser terminado o pulido. (Muchas veces, desconocen el esfuerzo necesario para agregar funciones de verificación de errores y seguridad que un prototipo puede no tener). Esto puede llevarlos a esperar que el prototipo modele con precisión el rendimiento del sistema final, cuando esa no es la intención de los desarrolladores. Los usuarios también pueden apegarse a características que se incluyeron en un prototipo para su consideración y luego se eliminaron de la especificación para el sistema final. Si los usuarios pueden exigir que se incluyan todas las características propuestas en el sistema final, esto puede dar lugar a conflictos.

# Desventajas del modelo de prototipo

**Incomprensión del desarrollador sobre los objetivos del usuario**: Los desarrolladores pueden asumir que los usuarios comparten sus objetivos (por ejemplo, entregar la funcionalidad principal a tiempo y dentro del presupuesto), sin comprender cuestiones comerciales más amplias. Por ejemplo, los representantes de los usuarios que asisten a eventos de software empresarial (por ejemplo, PeopleSoft) pueden haber visto demostraciones de "auditoría de transacciones" (donde se registran y muestran los cambios en una vista de cuadrícula de diferencias) sin que se les informe que esta función requiere codificación adicional y aparte, necesita más hardware para manejar los accesos adicionales a la base de datos. Los usuarios podrían creer que pueden exigir auditoría en cada campo, mientras que los desarrolladores podrían pensar que esto es un aumento no planificado en las funciones porque han hecho suposiciones sobre el alcance de los requisitos del usuario. Si el desarrollador ha comprometido la entrega antes de que se revisen los requisitos del usuario, los desarrolladores se encuentran en una situación complicada, especialmente si la dirección del usuario obtiene alguna ventaja de su incapacidad para implementar los requisitos.

# Desventajas del modelo de prototipo

**Apegamiento del desarrollador al prototipo**: Los desarrolladores también pueden apegarse a los prototipos en los que han invertido una gran cantidad de esfuerzo en su creación; esto puede dar lugar a problemas, como intentar convertir un prototipo limitado en un sistema final cuando no tiene una arquitectura subyacente adecuada. (Esto puede sugerir que se debe utilizar la prototipación desechable en lugar de la prototipación evolutiva, por lo que muchas veces capaz en realidad lo que debe replantearse es simplemente la utilización de una prototipación distinta, y ya está).

# Desventajas del modelo de prototipo

**Exceso de tiempo de desarrollo del prototipo**: Una propiedad clave de la prototipación es que se supone que se realiza de manera rápida. Si los desarrolladores pierden de vista este hecho, es posible que intenten desarrollar un prototipo demasiado complejo. Cuando se desecha el prototipo, los requisitos precisamente desarrollados que proporciona pueden no generar un aumento suficiente en la productividad que compense el tiempo invertido en desarrollar el prototipo. Los usuarios pueden quedar atrapados en debates sobre los detalles del prototipo, reteniendo al equipo de desarrollo y retrasando el producto final.

# Conclusiones del modelo de prototipo

Finalmente, hay que tener en cuenta que el cliente y el desarrollador se deben poner de acuerdo en:

- Que el prototipo se construya y sirva como un mecanismo para la definición de requisitos.
- Que el prototipo se descarte, al menos en parte.
- Que después se desarrolle el software real con un enfoque hacia la calidad.

Este modelo se va a utilizar en proyectos donde el cliente conoce los objetivos generales para el software, pero no cuenta con mayor detalle específico.