UNIVERSIDAD POLITÉCNICA DE PUEBLA

Ingeniería
Tecnologías de la Información

Investigación sobre Arquitecturas y Patrones de diseño

Ingeniería de Requisitos
7A ITI

Diana Aylin González Romero
Laura Felipa Reyes Medel
Johanna Rocha Santiago

¿Qué es diseño?

Definición de diseño

El diseño de software agrupa el conjunto de principios, conceptos y prácticas que llevan al desarrollo de un sistema o producto de alta calidad. Los principios de diseño establecen una filosofía general que guía el trabajo de diseño que debe ejecutarse.

Es el lugar en el que las reglas de la creatividad los **requerimientos de los participantes**, las **necesidades del negocio** y las **consideraciones técnicas** se unen para formular un producto o sistema. El diseño crea una representación o modelo del software, pero, a diferencia del modelo de los requerimientos (que se centra en describir los datos que se necesitan, la función y el comportamiento), el modelo de diseño proporciona detalles sobre arquitectura del software, estructuras de datos, interfaces y componentes que se necesitan para implementar el sistema

El diseño se basa en algunas características las cuales son:

1. La fabricación y el uso de herramientas
2. Comunicación
3. Reutilización de ideas
4. Creación de cosas nuevas
5. Importancia de las funciones:

- Fiabilidad
- Eficacia
- Buenas prácticas
- Claridad

Pressman, R. S. (2010). Ingeniería del software un enfoque práctico. En R. S. Pressman, Ingeniería del software un enfoque práctico (pág. 777). México, D. F.: Marcela I. Rocha Martínez.

Budgen, D. (2003). Software design (2nd Ed.). Pearson Education Limited.

¿Cuál es exactamente el propósito del diseño?

El propósito del diseño es simplemente producir una solución a un problema.

El problema suele resumirse en algún tipo de especificación de requisitos

La tarea del diseñador consiste en describir cómo se va a satisfacer ese requisito. Por lo tanto, el diseño es esencialmente una tarea de resolución de problemas, y los ejemplos dados muestran

Budgen, D. (2003). Software design (2nd Ed.). Pearson Education Limited.

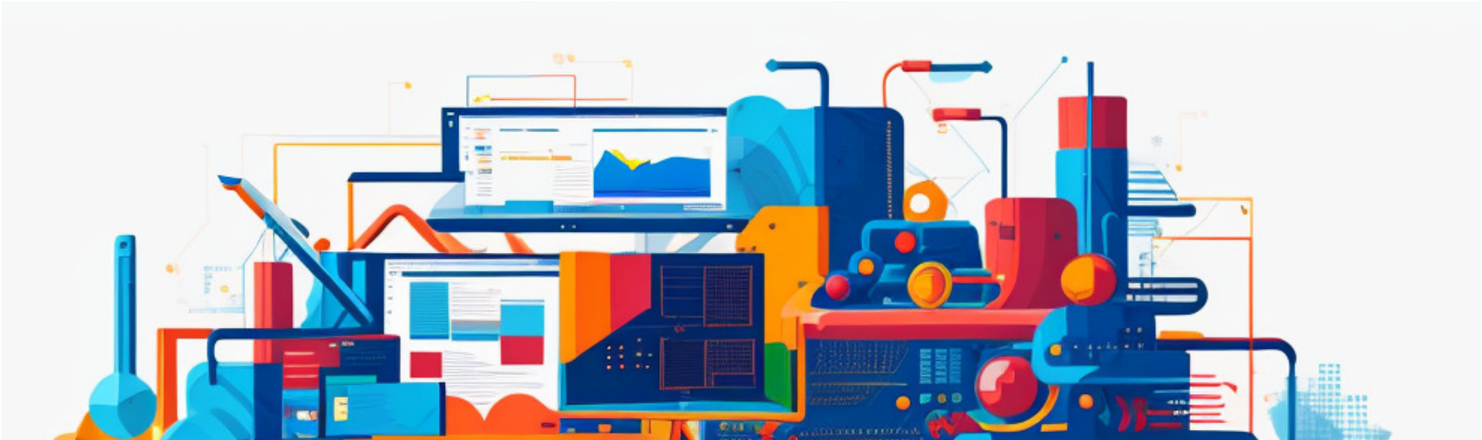
¿Por qué es importante modelar el contexto de un sistema que se desarrollará?

El diseño permite modelar el sistema o producto que se va a construir. Este modelo se evalúa respecto de la calidad y su mejora antes de generar código; después, se efectúan pruebas y se involucra a muchos usuarios finales. El diseño es el lugar en el que se establece la calidad del software.

Pressman, R. S. (2010). Ingeniería del software un enfoque práctico. En R. S. Pressman, Ingeniería del software un enfoque práctico (pág. 777). México, D. F.: Marcela I. Rocha Martínez.

El modelado de sistemas es el proceso para desarrollar modelos abstractos de un sistema, donde cada modelo presenta una visión o perspectiva diferente de dicho sistema.

Sommerville, I. (2011). Ingeniería del software. Pearson Educación.



Los modelos de contexto, incluye varios sistemas automatizados.
En una primera etapa en la especificación de un sistema, debe decidir sobre las fronteras del sistema. Esto implica trabajar con los participantes del sistema para determinar cuál funcionalidad se incluirá en el sistema y cuál la ofrece el entorno del sistema.

Por consiguiente, los modelos de contexto simples se usan junto con otros modelos, como los modelos de proceso empresarial. Éstos describen procesos humanos y automatizados que se usan en sistemas particulares de software.

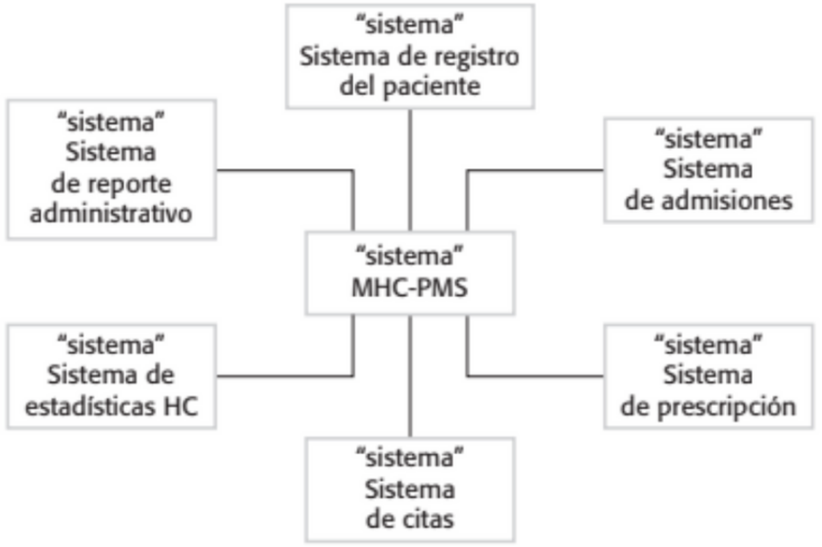
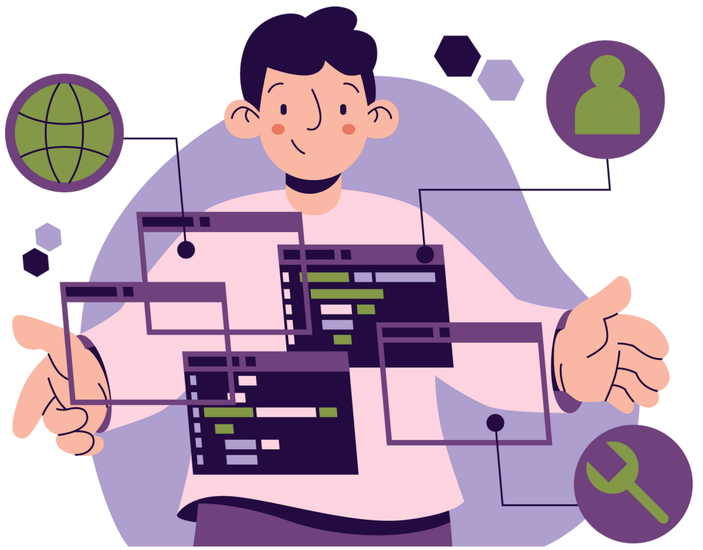


Figura 3. Modelo de contexto simple.

Sommerville, I. (2011). Ingeniería del software. Pearson Educación.

Al no entender el contexto del sistema pueden pasar diversas situaciones que terminen en errores de software como:

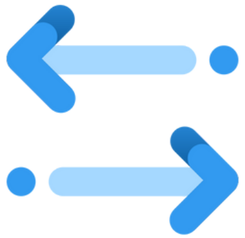
- Diseño de sistema.
- Relaciones del sistema.
- Comprensión.
- Cambio constante.
- Conformidad.



Complejidad. Se considera una propiedad esencial del software, en el que no hay dos partes iguales y un sistema puede tener muchos estados durante su ejecución.
dos partes son iguales y un sistema puede poseer muchos estados durante su ejecución.
Esta complejidad también es arbitraria, ya que depende del diseñador y no del problema.
del problema.

Conformidad. Al ser "flexible", se espera que el software se ajuste a las normas impuestas por otros componentes, como el hardware, por organismos externos o por el software existente.

Cambiabilidad. El software sufre una necesidad constante de cambio, en parte debido a la aparente facilidad para hacer cambios (y las técnicas relativamente pobres para calcular su coste).



Invisibilidad. La invisibilidad del software dificulta su representación visual y comprensión, limitando la comunicación en su desarrollo. A diferencia de un plano de edificio, no hay vínculo visual directo con las características del software.

Budgen, D. (2003). Software design (2nd Ed.). Pearson Education Limited.

NOTA:

Todas estas relaciones llegan a afectar los requerimientos y el diseño del sistema a definir, por lo que deben tomarse en cuenta.

Gracias por su atención