Step 3 - Requirenments Engineering and Management

Daniela Ramírez Zuluaga COD 1143845172

Gabriel Felipe Roa Pinzón COD 1024559306

Oscar Perez Jiménez COD 77181989

GRUPO

301309_47

PRESENTADO A

MOISES DE JESUS RODRIGUEZ

Tutor

Universidad Nacional Abierta y a Distancia

Diseño de sistemas

Noviembre

INTRODUCCION

Por medio de este se pretende dar respuesta a las preguntas formuladas dentro de la actividad teniendo en cuenta las referencias dadas y la investigación realizada por cada estudiante referente al diseño de software, calidad del diseño de software, patrones, estilos, características, implementación, Diseño de la interfaz de usuario, Diseño basado en patrones, Diseño de WebApps y demás temas relacionados con la manera en la que se realiza un diseño de software y como se implemente.

OBJETIVOS

- Interiorizar la teoría sobre arquitectura del software, estructuras de datos, interfaces y componentes que se necesitan para implementar el sistema
- Desarrollar el diseño para un sistema de información o producto informático de alta calidad
- Identificar las distintas representaciones de un software tomando como características la resistencia, la funcionalidad y la belleza.

DESARROLLO DE ACTIVIDADES

1. Enuncie los principios de diseño que "den el control al usuario".

Mandel [Man97] define cierto número de principios de diseño que permiten que **el usuario tenga el control:**

Definir modos de interacción de manera que no se obligue al usuario a realizar acciones Innecesarias o no deseadas. Un modo de interacción es el estado actual de la interfaz.

Por ejemplo, si en el menú de un procesador de textos se selecciona revisar ortografía, el software pasa al modo de revisión de la ortografía. No hay razón para obligar al usuario a permanecer en este modo si acaso desea hacer una pequeña edición del texto. El usuario debe poder entrar y salir del modo con poco o ningún esfuerzo.

Dar una interacción flexible. Debido a que diferentes usuarios tienen distintas preferencias para la interacción, debe darse la posibilidad de elegir. Por ejemplo, el software debe permitir que el usuario interactúe por medio de comandos introducidos con el teclado, el ratón, una pluma digitalizadora, una pantalla sensible al tacto o un mecanismo de reconocimiento de voz.
Pero no todas las acciones son accesibles a través de cualquier mecanismo de interacción. Por ejemplo, piénsese en la dificultad de usar comandos del teclado (o entradas con la voz) para hacer un dibujo complicado.

2. Enuncie los principios de diseño "reduzcan la necesidad de memorización por parte del usuario".

Mandel [Man97] define los siguientes principios de diseño que permiten que una interfaz reduzca la **necesidad de que el usuario memorice:**

Reducir la demanda de memoria de corto plazo. Cuando los usuarios se involucran en tareas complejas, la demanda de memoria de corto plazo es significativa. La interfaz debe diseñarse para disminuir la necesidad de recordar acciones, entradas y resultados del pasado. Esto se logra dando claves visuales que permitan al usuario reconocer acciones anteriores, en lugar de que tenga que recordarlas.

Hacer que lo preestablecido sea significativo. Lo que al principio se dé por preestablecido debe tener sentido para el usuario promedio, pero éste debería poder especificar sus preferencias individuales. Sin embargo, debe disponerse de la opción de "reiniciar" para restablecer los valores originales.

Definir atajos que sean intuitivos. Cuando se utilice nemotecnia para ejecutar una función del sistema (como la secuencia Ctrl-B para invocar la función de buscar), debe estar ligada con la

acción, de modo que sea fácil de recordar (por ejemplo, con la primera letra de la tarea que se va a realizar).

La distribución visual de la interfaz debe basarse en una metáfora del mundo real. Por ejemplo, un sistema de pagos debe usar una metáfora de chequera y talonario que guíe al usuario a través del proceso de pago. Esto permite que el usuario se base en claves visuales que comprende bien, en vez de tener que memorizar una secuencia críptica de interacciones.

Revelar información de manera progresiva. La interfaz debe estar organizada de manera jerárquica. Es decir, la información acerca de una tarea, objeto o comportamiento debe presentarse primero en un nivel de generalización elevado. Después de que con el ratón el usuario manifieste interés, deben darse más detalles. Un ejemplo, común para muchas aplicaciones de procesamiento de textos, es la función de subrayar. La función en sí es una de varias en el menú *estilo del texto*. No obstante, no se enlista cada una de las herramientas para subrayar. El usuario debe hacer *clic* en la opción de subrayar; después se presentan todas las opciones para esta función (una raya, doble raya, línea punteada, etcétera).

3. Enuncie principios de diseño que "hagan consistente a la interfaz".

La interfaz debe presentar y obtener información en forma consistente. Esto implica: 1) que toda la información se organice de acuerdo con reglas de diseño que se respeten en todas las pantallas desplegadas, 2) que los mecanismos de entrada se limiten a un conjunto pequeño usado en forma consistente en toda la aplicación, y 3) que los mecanismos para pasar de una tarea a otra se definan e implementen de modo consistente.

Mandel [Man97] define varios principios de diseño que ayudan a que la interfaz tenga consistencia:

Permita que el usuario coloque la tarea en curso en un contexto significativo. Muchas interfaces implementan capas complejas de interacciones con decenas de imágenes en la pantalla. Es importante dar indicadores (títulos en las ventanas, iconos gráficos, código de colores consistente, etc.) que permitan al usuario conocer el contexto del trabajo en curso. Además, debe poder determinar de dónde viene y qué alternativas hay para hacer la transición a una nueva tarea.

Mantener la consistencia en toda la familia de aplicaciones. Todas las aplicaciones (o productos) que hay en un grupo deben implementar las mismas reglas de diseño a fin de que se mantenga la consistencia en toda la interacción.

Si los modelos interactivos anteriores han creado expectativas en el usuario, no haga cambios a menos de que haya una razón ineludible para ello. Una vez que una secuencia interactiva en particular se ha convertido en un estándar (como el uso de alt-G para guardar un archivo), el usuario la espera en toda aplicación que emplea. Un cambio (como utilizar alt-G para invocar la función de modificar la escala) generará confusión.

Los principios de diseño de la interfaz analizados en esta sección y en las anteriores dan una guía básica. En las que siguen, el lector aprenderá acerca del proceso de diseño de la interfaz en sí.

CONCLUSIONES

REFERENCIAS

- Design Patterns. Elements of Reusable Object-Oriented Software - Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides - Addison Wesley (GoF- Gang of Four)
- A System of Patterns - Buschmann, Meunier, Rohnert, Sommerlad, Stal – Wiley
- En la última década se han escrito muchos libros sobre el diseño de patrones destinados a los ingenieros de software. Gamma et al. [Gam95] escribieron un libro fundamental sobre dicho tema. Las contribuciones más recientes incluyen los textos de Lasater (Design Patterns, Wordware Publishing, Inc., 2007), Holzner (Design Patterns for Dummies, For Dummies, 2006), Freeman et al. (Head First Design Patterns, O'Reilly Media, Inc., 2005) y Shalloway y Trott (Design Patterns Explained, 2a. ed., Addison-Wesley, 2004).