SECRETARÍA DE EDUCACIÓN PÚBLICA

DIRECCIÓN GENERAL DE EDUCACIÓN SUPERIOR TECNOLÓGICA

INSTITUTO TECNOLÓGICO DE VERACRUZ

Carrera:
Ingeniería Mecatrónica

Materia:
Simulación

Titular de la materia:
Dr. José Antonio Garrido Natarén

Tema:
"Herramientas de programación de modelos de simulación numérica – Ciclo de Vida de Software"

H. Veracruz, Ver. Octubre de 2014

Índice

Ciclo de vida de un software

1. Introducción

El término **ciclo de vida del software** describe el desarrollo de software, desde la fase inicial hasta la fase final.

El propósito de este proceso es definir las distintas fases intermedias que se requieren para **validar** el desarrollo de la aplicación, es decir, para garantizar que el software cumpla los requisitos para la aplicación y **verificación** de los procedimientos de desarrollo: se asegura de que los métodos utilizados son apropiados.

El ciclo de vida permite que los errores se detecten lo antes posible y por lo tanto, permite a los desarrolladores concentrarse en la calidad del software, en los plazos de implementación y en los costos asociados.

Las fases que podemos encontrar en el ciclo de vida son:

1.- Definición de objetivos: definir el resultado del proyecto y su papel en la estrategia global.

2.- Análisis de los requisitos y su viabilidad: recopilar, examinar y formular los requisitos del cliente y examinar cualquier restricción que se pueda aplicar.

3.- Diseño general: requisitos generales de la arquitectura de la aplicación.

4.- Diseño en detalle: definición precisa de cada subconjunto de la aplicación.

5.- Programación (programación e implementación): es la implementación de un lenguaje de programación para crear las funciones definidas durante la etapa de diseño.

6.- Prueba de unidad: prueba individual de cada subconjunto de la aplicación para garantizar que se implementaron de acuerdo con las especificaciones.

7.- Integración: para garantizar que los diferentes módulos se integren con la aplicación. Éste es el propósito de la prueba de integración que está cuidadosamente documentada.

8.- Prueba beta (o validación), para garantizar que el software cumple con las especificaciones originales.

9.- Documentación: sirve para documentar información necesaria para los usuarios del software y para desarrollos futuros.

10.- Implementación

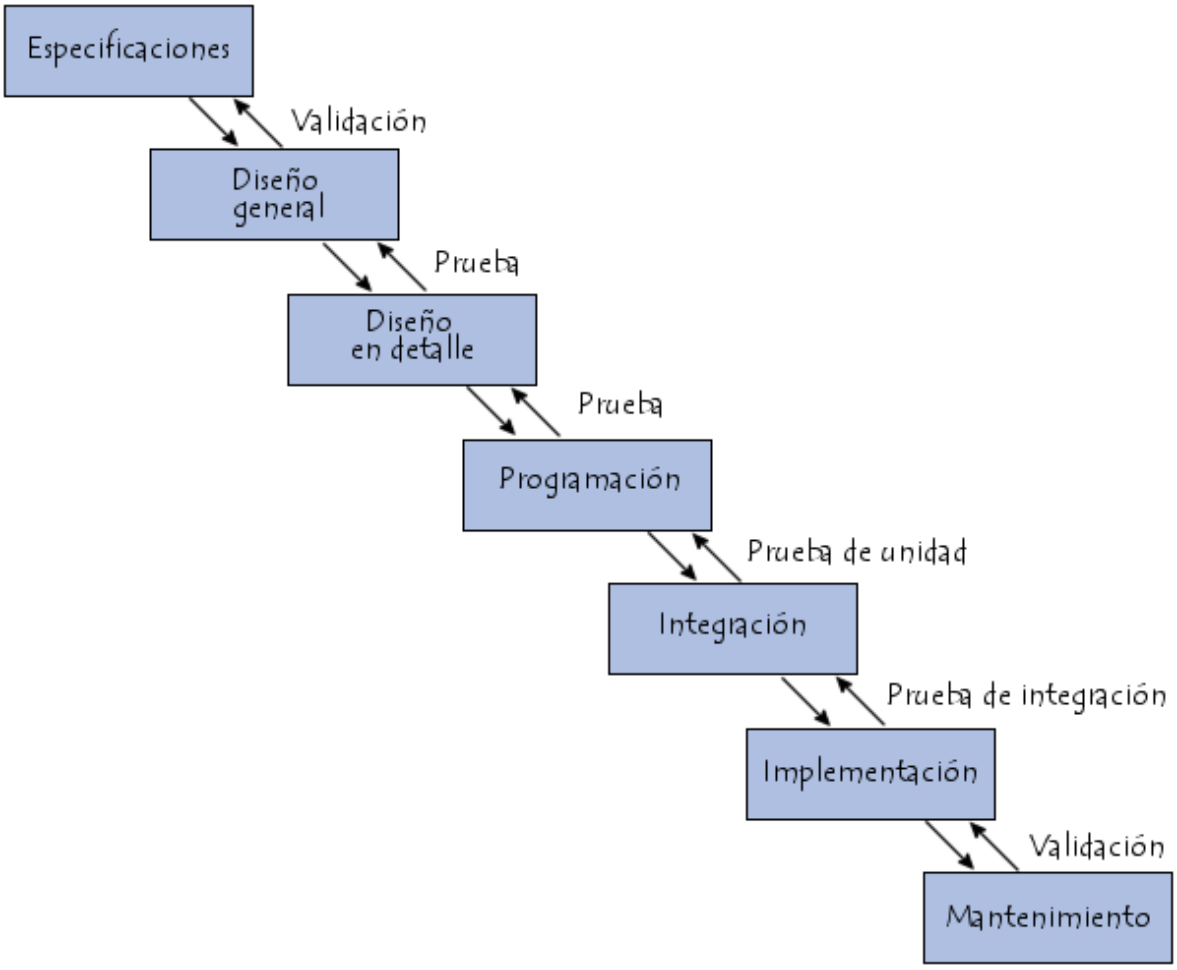
11.- Mantenimiento: para todos los procedimientos correctivos (mantenimiento correctivo) y las actualizaciones secundarias del software (mantenimiento continuo).

1.1 Modelos de ciclo de vida

1.1.1 Modelo Cascada

El modelo de ciclo de vida cascada, captura algunos principios básicos:

- Planear un proyecto antes de embarcarse en él.
- Definir el comportamiento externo deseado del sistema antes de diseñar su arquitectura interna.
- Documentar los resultados de cada actividad.
- Diseñar un sistema antes de codificarlo.
- Testear un sistema después de construirlo

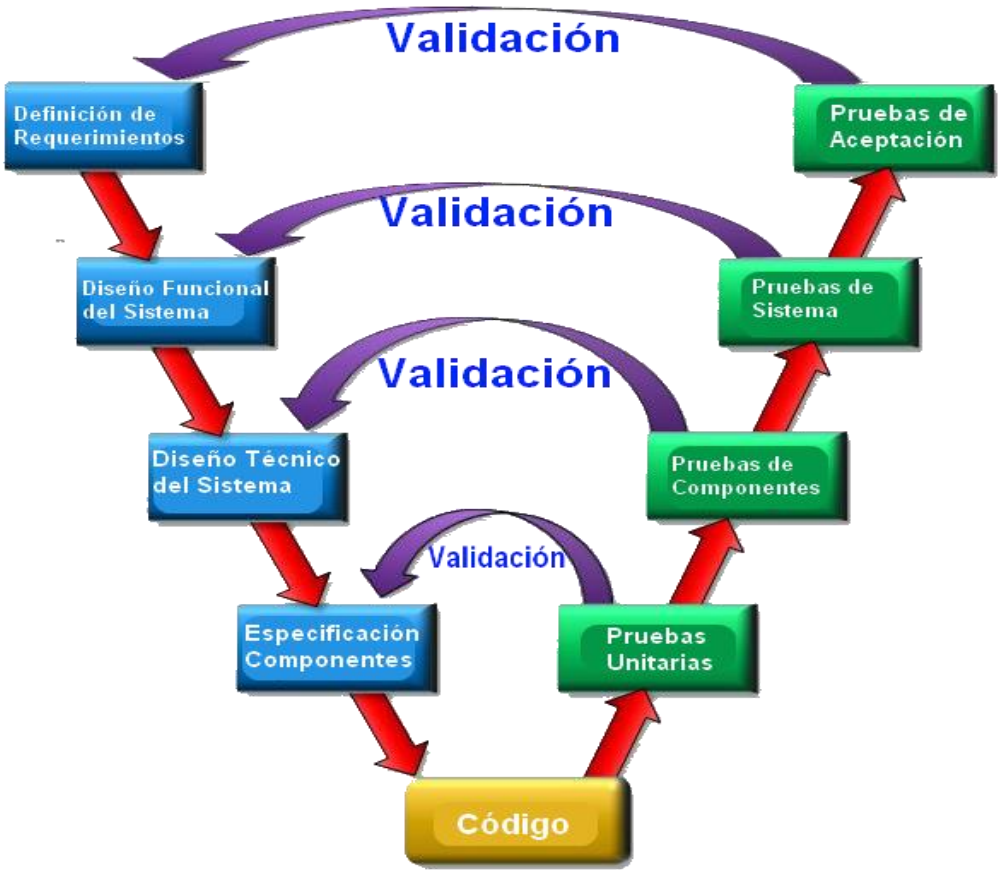


Ventajas	Inconvenientes
Las ventajas que se pueden destacar de este modelo son las siguientes: • En cada una de las fases hay entregables específicos	Entre los inconvenientes y las críticas que se le hacen a este modelo están las siguientes: • Es un modelo muy rígido, como el modelo en V

• Tiene una alta oportunidad de éxito sobre el modelo en cascada debido al desarrollo de planes de prueba en etapas tempranas del ciclo de vida. • Es un modelo que suele funcionar bien para proyectos pequeños donde los requisitos son entendidos fácilmente.	• Tiene poca flexibilidad y ajustar el alcance es difícil y caro (lo cual nunca es deseable) • El software se desarrolla durante la fase de implementación, por lo que no se producen prototipos del software

1.1.2 Modelo V

En la práctica, este modelo puede tener más o menos niveles dependiendo del proyecto y el producto que se está desarrollando. Se creó como respuesta a los distintos problemas que ocurrían en el desarrollo en cascada (Waterfall Model). El Modelo V, propone realizar testing desde el comienzo del proyecto, realizando Testing Estático en cada una de las fases de desarrollo, preparando las pruebas para esos niveles y luego ejecutando las mismas en forma ascendente se tiene siempre un control sobre cada una de las etapas.



Ventajas	Inconvenientes
Las ventajas que se pueden destacar de este modelo son las siguientes: • En cada una de las fases hay entregables específicos • Tiene una alta oportunidad de éxito sobre el modelo en cascada debido al desarrollo de planes de prueba en etapas tempranas del ciclo de vida. • El modelo en V hace más explícita la tarea parte de la iteración de las actividades del proceso • Las pruebas de cada fase ayudaran a corregir posibles errores sin tener que esperar a que sean rectificados en la etapa final del proceso. • Con las pruebas unitarias y de integración se consigue obtener exactitud en los programas..	Entre los inconvenientes y las críticas que se le hacen a este modelo están las siguientes: • Es un modelo muy rígido, como el modelo en cascada • Tiene poca flexibilidad y ajustar el alcance es difícil • El software se desarrolla durante la fase de implementación, por lo que no se producen prototipos del software

1.1.3 Alternativas de Modelos de Ciclo de Vida

Existen otros modelos de ciclo de vida, en éste apartado a continuación se irá enumerarán las alternativas, con sus ventajas y desventajas; incluyendo la del ciclo en V y la del modelo cascada.

Algunas de las alternativas de modelos con las que contamos para el ciclo de vida son:

1. Modelo Iterativo
2. Modelo de Incremental
3. Modelo Espiral
4. Modelo de Desarrollo Evolutivo
5. Modelo Concurrente

1.1.3.1 Modelo Iterativo (prueba y error)

Ventajas	Inconvenientes
• Una de las principales ventajas que ofrece este modelo es que no hace falta que los requisitos estén totalmente definidos al inicio del desarrollo, sino que se pueden ir refinando en cada una de las iteraciones. • Igual que otros modelos similares tiene las ventajas propias de realizar el desarrollo en pequeños ciclos, lo que permite gestionar mejor los riesgos, gestionar mejor las entregas	La primera de las ventajas que ofrece este modelo, el no ser necesario tener los requisitos definidos desde el principio, puede verse también como un inconveniente ya que pueden surgir problemas relacionados con la arquitectura.

1.1.3.2 Modelo de Incremental

Ventajas	Inconvenientes
Entre las ventajas que puede proporcionar un modelo de este tipo encontramos las siguientes: • Mediante este modelo se genera software operativo de forma rápida y en etapas tempranas del ciclo de vida del software. • Es un modelo más flexible, por lo que se reduce el coste en el cambio de alcance y requisitos. • Es más fácil probar y depurar en una iteración más pequeña. • Es más fácil gestionar riesgos. • Cada iteración es un hito gestionado fácilmente	Entre los inconvenientes que aparecen en el uso de este modelo podemos destacar los siguientes: • Cada fase de una iteración es rígida y no se superponen con otras. • Pueden surgir problemas referidos a la arquitectura del sistema porque no todos los requisitos se han reunido, ya que se supone que todos ellos se han definido al inicio.

4.4.2.5 Modelo en Espiral

Ventajas	Inconvenientes
El análisis de riesgos se hace de forma explícita y clara. Une los mejores elementos de los restantes modelos. Entre ellos: • Reduce riesgos del proyecto • Incorpora objetivos de calidad • Integra el desarrollo con el mantenimiento Además es posible tener en cuenta mejoras y nuevos requerimientos sin romper con el modelo, ya que el ciclo de vida no es rígido ni estático. Mediante este modelo se produce software en etapas tempranas del ciclo de vida y suele ser adecuado para proyectos largos de misión crítica.	Es un modelo que genera mucho trabajo adicional. Al ser el análisis de riesgos una de las tareas principales exige un alto nivel de experiencia y cierta habilidad en los analistas de riesgos (es bastante difícil). Esto puede llevar a que sea un modelo costoso. Además, no es un modelo que funcione bien para proyectos pequeños.

4.5 Ergonomía y técnicas de presentación de resultados

Teniendo claro los lenguajes de simulación que existen, los algoritmos usados, los tipos de programación (declarativa, imperativa, estructurada y objeto) y el ciclo de vida de un software; ahora nos toca abordar la ergonomía y las técnicas de presentación de resultados, ya que ¿de qué nos serviría hacer una óptima selección del lenguaje, si nuestra interfaz no es ergonómica o si no contamos con una presentación adecuada? Obviamente el programa resultaría confuso para el usuario o simplemente no se entenderían los resultados.

Al ser las simulaciones en la actualidad completamente gráficas, la representación adecuada de los resultados juega un papel fundamental al momento de realizar el programa de simulación cualquiera que sea su fin; además de que éstas deben de ser amigables y fáciles de entender para el usuario final, por lo que también hay que considerar los aspectos ergonómicos al momento de realizar la interfaz; es por eso que éste tópico se incluye en la unidad de herramientas de programación de modelos de simulación numérica.

4.5.1 Ergonomía

Antes de adentrarnos a los aspectos ergonómicos de un software primero debemos de tener un concepto claro de lo que es la ergonomía. La Real Academia Española, define ergonomía como:

"Estudio de las condiciones de adaptación de un lugar de trabajo, una máquina, un vehículo; a las características físicas y psicológicas del trabajador o usuario."

Es decir, es una rama del saber que tiene como finalidad adaptar un entorno a las características o requerimientos del usuario.

Aplicándolo al diseño de software ya sea de simulación o con otros fines, hay que tener en mente tres puntos:

2. Focalizan sus estudios en los aspectos físico y mental de los interfaces entre el usuario y los programas.
3. Intentan diseñar procedimientos de diálogo y formatos de presentación que sean efectivos y fáciles de usar.
4. Buscan que las aplicaciones sean fáciles de entender y aprender y que potencien los conocimientos de quienes los usan.

Las razones principales por las cuales hay que tener en cuenta éste factor al momento de programar radican en tres aspectos primordiales, los cuales son:

1) Un software fácil de usar, fácil de aprender y entretenido en su manejo; reducen los costes de formación y el tiempo de aprendizaje a la vez que incrementan la productividad, el uso de los ordenadores y la motivación para trabajar confortablemente en el puesto de trabajo. Un ejemplo sería cuando se compara una aplicación de ventanas, basada en un entorno Microsoft Windows, con la típica aplicación basada en el tratamiento de caracteres, se pueden llegar a reducir los períodos de aprendizaje hasta un 50 por ciento, dependiendo de la experiencia previa del usuario en entornos Windows. Para ciertas empresas, este ahorro puede significar mil
2) Los aspectos ergonómicos son hoy en día uno de los principales argumentos de venta. La principal diferencia entre distintos productos o casas de software conciernen a los aspectos ergonómicos.
3) Dentro de ISO, la norma 9241, partes 8 a 17, cubre también aspectos relacionados con el software y obliga a los desarrolladores a tener en cuenta sus indicaciones. Muchos países usarán las normas ISO para crear su propia legislación local acerca de los programas y aplicaciones software.

Las normas ISO aplicables al diseño ergonómico son varias y abarcan desde la realización de guías de usuario hasta la confección de los cuadros de diálogo, como más adelante se verá.

4.5.1.1 Aspectos ergonómicos a considerar

Cuando alguien está trabajando con un producto software, espera que la interacción o diálogo con el ordenador sea fácil y eficiente. Las técnicas de diálogo incluyen menús, comandos, manipulaciones directas y formatos de entrada de datos.

Un producto software que quiera ser catalogado como ergonómico, deberá seguir los siguientes siete principios que rigen el diseño de los diálogos ergonómicos:

1) Adecuado para el trabajo al que se destina: dotar a los productos de herramientas adecuadas, es decir, una clara definición de lo que realizan las diferentes opciones del software, además de ocultar la complejidad de los procesos internos que el programa y/o el ordenador esté realizando.

2) Autodescriptivo: la aplicación debe de ser fácil y rápidamente comprendida por el usuario, en otras palabras el usuario no tendrá que verificar en el manual qué es lo que se puede esperar de un determinado menú o qué significado tienen los diferentes términos y palabras que aparezcan en pantalla. El sistema deberá ofrecer al usuario un sistema de diálogo claro, simple y conciso, apoyado por un mecanismo de pantallas de ayuda de fácil acceso, que contengan explicaciones concretas.

3) Controlable: el consumidor tendrá en todo momento la posibilidad de cancelar acciones que haya emprendido, deshacer los últimos comandos que haya ordenado y gobernar sus dispositivos de entrada y salida de datos.

4) Conforme a las expectaciones que genera: que cumpla con las cualidades con las que ha sido publicitado y que todos los componentes cumplan con la función que deben de desempeñar.

5) Tolerante con los errores que el usuario pueda cometer: es importante que no permita a los usuarios el ejecutar tareas que puedan provocar un error irrecuperable. No sólo deben detectar y avisar al usuario de los errores, sino que deben tratar de prevenir al mismo de lo que puede suceder. Cuando se produzca un error, el usuario podrá ser capaz de salir del mismo, comprender qué es lo que ha sucedido y tener a su elección una serie de opciones de salida del proceso.

6) Personalizable por el usuario: cualquier usuario independientemente de su nivel de conocimientos, podrá personalizar su área de trabajo según le convenga y según conciba que puede aumentar la efectividad del uso de los programas. Podrá definir colores, formas, agrupamiento de iconos, creación y agrupamiento de macros, entre muchas otras cualidades.

7) Documentado suficientemente para facilitar su aprendizaje: contarán con explicaciones coherentes (tanto en pantalla como en manuales) que vayan encaminadas a facilitar la labor de aprendizaje.

4.5.1.2 Normas aplicables a la ergonomía del software

Como lo señala Fenoulière (2002), “la normalización por lo general sólo es visto por las empresas a través de su resultado, la certificación, un paso obligado para ser reconocido por sus pares, e incluso una aprobación indispensable para estar presente en un mercado determinado”.

Esta certificación generalmente se refiere a las normas relativas a la gestión de la calidad (ISO 9000). A pesar de que la certificación no sea un tema de actualidad en la ergonomía de software, las cosas avanzan y al aumentar cada vez más los usuarios de software, es necesario presentar normas que regulen su creación y entorno.

Las normas que regulan la creación, desarrollo, entorno, entre muchas otras cualidades del software son:

- ISO 13407 proceso de diseño centrado en el usuario para sistemas interactivos

Esta parte proporciona recomendaciones relativas a procesos de diseño centrados en el usuario a través de toda la vida útil de los sistemas interactivos informáticos. Esta norma está dirigida a los responsables de los procesos de diseño (los jefes de proyecto) y proporciona una guía de fuentes de información y normas que tratan del enfoque centrado en el usuario. Por lo tanto, esta norma tiene que ver con la planificación y la gestión del diseño centrada en el usuario.

Indica al jefe del proyecto cómo evaluar la conformidad del proceso de diseño que ha puesto en práctica, con la norma 13407. Éste debe probar que los objetivos de usabilidad han sido objeto de test y que han sido realizados utilizando métodos válidos, y además que una cantidad apropiada de usuarios ha participado en él y son representativos de los futuros usuarios, y que los datos producto de estos test han sido tratados de manera apropiada.

- ISO/TR 16982: Métodos de usabilidad que soportan diseño centrado en el usuario

Este documento presenta una lista de métodos ergonómicos que pueden ser aplicados a las diferentes etapas del ciclo de diseño, precisando sus ventajas y desventajas. La observación, la medida del desempeño, la técnica de los incidentes críticos, los cuestionarios, las entrevistas, las técnicas de diseño y evaluación participativa.

- ISO 9241-10: Principios para diálogos

Esta parte describe principios generales de ergonomía juzgados importantes para el diseño y evaluación de diálogos entre el usuario y los sistemas de información (adaptación a la tarea, carácter auto descriptivo, control por parte del usuario, conformidad con las expectativas del usuario, tolerancia a errores, aptitud a la individualización, facilidad de aprendizaje).

- ISO 9241-11: Guía de especificaciones y medidas de usabilidad

Esta parte define la usabilidad y explica cómo identificar la información a tomar en cuenta para especificar o evaluar la usabilidad, en términos de desempeño y satisfacción del usuario. Proporciona directrices para la descripción del contexto de usabilidad del software y las medidas pertinentes relativas a la usabilidad (medida de la eficacia y de la eficiencia).

- ISO 9241-12: Presentación de la información

Proporciona recomendaciones ergonómicas relativas a la presentación y a las propiedades particulares de la información presentada en pantallas de visualización. Las recomendaciones proporcionadas tienen como objetivo permitir al usuario ejecutar tareas de percepción de manera eficaz y satisfactoria; desde la organización de la información, hasta las técnicas de decodificación de la información.

- ISO 9241-13: Guía del usuario

Las recomendaciones presentadas en esta parte deberían facilitar la interacción de un usuario con un programa, favoreciendo el uso eficaz del programa, evitando la carga de trabajo mental inútil, proporcionando a los usuarios un medio de gestión de errores y un asistente a los usuarios con niveles de conocimiento diferente.

- ISO 9241-14: Diálogos de menús

Esta parte proporciona recomendaciones para el diseño ergonómico de los menús, es decir tipos de interacción en el que se presentan opciones a los usuarios bajo diferentes formas (ventanas de dialogo con casillas a marcar, botones, campos, etc.). En esta parte, numerosas recomendaciones son condicionales, es decir que sólo deberían ser aplicadas en contextos específicos (ej. Tipo particular de usuario, de tarea, de entorno, de tecnología, etc.).

- ISO 9241-15: Diálogos de tipo lenguaje de órdenes

Este documento aborda la estructura y la sintaxis del lenguaje de órdenes, la representación de los comandos (nombres, abreviación, teclas de función, etc.), los aspectos relativos a los modos de entrada y salida, el feedback y la ayuda.

- ISO 9241-16: Diálogos de manipulación directa

Esta parte aborda las metáforas graficas, la apariencia de los objetos utilizados en la manipulación directa, el feedback, los dispositivos de entrada de datos, la manipulación de objetos, el punteo y la selección, el dimensionamiento, la manipulación directa de las ventanas y los iconos, etc.

- ISO 9241-17: Diálogos por cumplimentación de formularios

Las recomendaciones dadas en esta parte tienen que ver con la estructura de los formularios, los campos y etiquetas, las entradas (textuales alfanuméricas, de opción, los controles, las validaciones, etc.), el feedback y la navegación en el formulario.

- ISO 14915: Ergonomía del software para interfaces de usuario multimedia

Proporciona recomendaciones acerca del diseño de controles y la navegación (ej. controles de audio, funciones como “play”, “stop”, “pausa”, etc.), sobre medios específicos y sobre su articulación, y dominios de aplicación específicos como la formación asistida por ordenador, los bornes interactivos, etc.

- SO/TS 16071: La accesibilidad a interfaces

El objetivo de este documento es el de proporcionar recomendaciones a los desarrolladores que les permita garantizar la accesibilidad a las interfaces a los usuarios, es decir favorecer la eficacia y satisfacción de los usuarios con necesidades específicas a causa de deficiencias motoras, sensoriales o cognitivas (ej. Personas incapacitadas o invidentes, etc.). Por otra parte las recomendaciones propuestas deberán disminuir el uso de herramientas de asistencia de software y hardware como las síntesis de habla, el Braille, o aun favorecer su uso.

4.5.2. Técnicas de presentación de resultados

Una presentación es una forma de ofrecer y mostrar información de datos y resultados de un proceso. Con la presentación se dispone de un contenido multimedia (es decir cualquier apoyo visual o auditivo) que de una referencia sobre el tema y ayude a explicar los datos obtenidos en el proceso.

En lo que concierne al mundo de la simulación, la representación de resultados de manera general, puede ser de dos maneras gráfica o textual.

En un principio, la representación textual era la predilecta de los programas de simulación, consistía en la representación alfanumérica línea por línea de los resultados obtenidos durante la simulación. Éste suele ser cansado y tedioso, ya que la pantalla del ordenador se llena de líneas de números, dejándole todo el trabajo de interpretación y presentación al usuario.

No obstante, con el desarrollo de nueva tecnología se fue desarrollando poco a poco sistemas de gráficos que permitieron pasar de pantallas llenas de líneas con números a gráficas y animaciones; agilizando el análisis de los resultados obtenidos y reduciendo el tiempo empleado en su interpretación.

En la actualidad la representación de los datos admiten sistemas completamente gráficos o numéricos, sin embargo los más utilizados son los híbridos, donde tablas y gráficos se utilizan para la exposición de resultados.

En lo que corresponde a la representación textual, hay que tener en cuenta el tipo de variable que vayamos a representar si es cualitativa, cuantitativa, discreta o continua.

Si adoptamos un sistema gráfico, tenemos varias opciones para poder representar nuestros resultados, no obstante, por convención se utilizan principalmente cuatro tipos de gráficos:

1) Diagramas de líneas

Utilizados para la representación de relaciones entre variables, haciendo ahínco a las correspondencias de incremento o decremento de los datos.

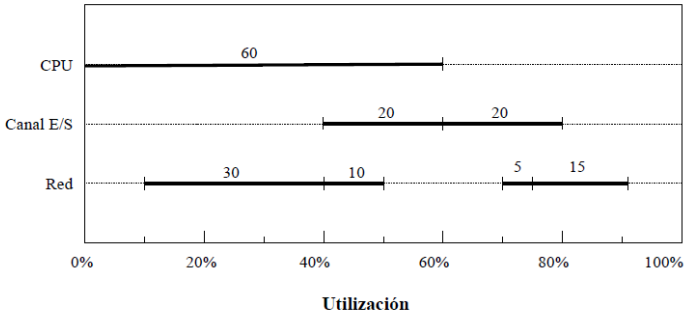
2) Diagramas de barras o histogramas

Se usan cuando lo que se quiere expresar es el incremento/decremento de varios recursos, donde un recurso no está estrechamente relacionado con el anterior o posterior, pero de una manera general si lo hacen. Por ejemplo, se puede relacionar los accidentes ocurridos al mes durante un año con un diagrama de barras; donde cada mes (recurso) estará relacionado con los otros porque pertenecen al mismo año (grupo de datos), más la incidencia de accidentes de cada uno, no tiene ninguna relación con la de los otros meses.

3) Diagramas de Gantt

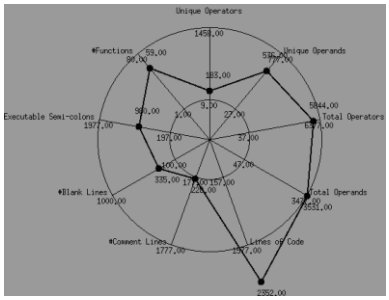
Muestra la duración relativa de condiciones o estados, partiendo de las primicias que un recurso con una utilización alta puede generar un cuello de botella y degradar el rendimiento y que uno con utilización baja representa una ineficiencia. Pretende la óptima utilización de recursos.

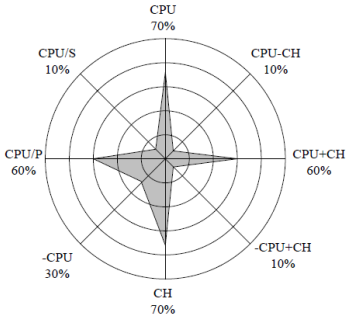
En ellos cada condición es representada por un conjunto de segmentos de línea, lo cual permite mostrar varios recursos; más **no** muestra dependencias.



4) Gráficas de Kiviat

Nos permite representar 3 o más índices de rendimiento, de una manera clara y precisa.

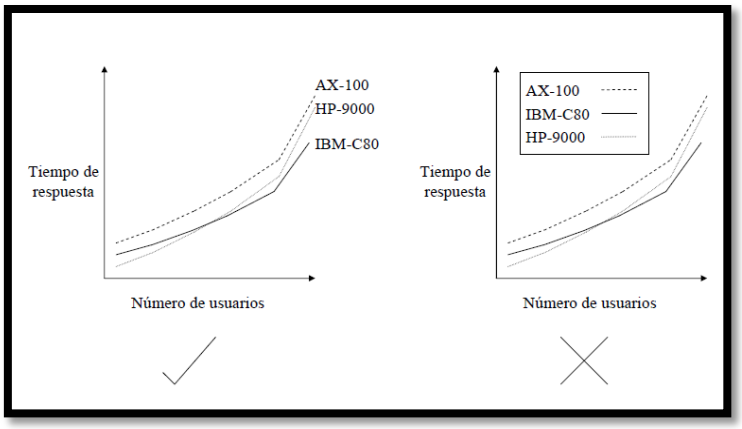




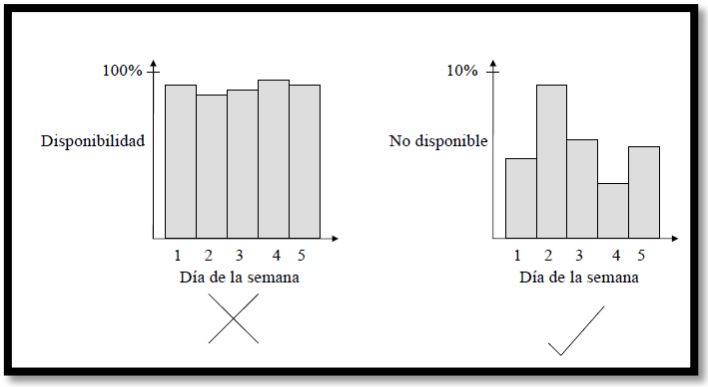
4.5.2.1 Criterios prácticos a la hora de presentar resultados

Al momento de presentar nuestros resultados, debemos de tener en cuenta algunos aspectos los cuales se podrían reducir a los siguientes puntos:

» Mínimo esfuerzo y carga gráfica mínima: Tratar de que la gráfica a emplear, sea lo más minimalista posible, mostrando los datos relevantes.



» Maximizar la información ofrecida: Es decir, mostrar la información más relevante o las que pueden causar un mayor impacto al usuario. En la imagen que se presenta abajo, se ejemplifica lo dicho, ya que será de más importancia e impactante ver la no disponibilidad que la disponibilidad de cierto artículo durante una semana, ya que éste es el que nos muestra mayor variabilidad.



» Minimizar la carga del gráfico.

» Aplicar prácticas comúnmente aceptadas

- Variable independiente en el eje x.
- Origen representa (0,0).
- Escalas incrementan de izquierda a derecha y de abajo hacia arriba.

» Evitar ambigüedad.

4.5.2.2 Errores comunes

Para poder evitarlos es necesario conocerlos, por lo que a continuación los enunciaremos:

» Presentar muchas alternativas en un diagrama: Si tiene muchas variantes, mostrarlas en varios gráficos, especificando las condiciones de cada variante.

» Presentar muchas variables independientes: En algunos casos, si tenemos muchas variables que representar; es más fácil realizarlos en varios gráficos que sólo en uno. Esto se hace con la finalidad de mejorar la productividad al momento de analizar gráficos, llegando a las conclusiones más rápido.

» Uso excesivo de símbolos: Ya que un símbolo puede significar muchas cosas aún en un tema en específico es necesario que solamente se incluyan, cuando el uso de los mismos ayude a la comprensión de los datos.

» Introducción de señales innecesarias (guías): Es decir, incluir indicaciones que no se han pedido o no son relevantes para el entendimiento del diagrama.

» Selección de escalas inapropiadas: Hay que tener en cuenta éste punto, ya que si no se escoge bien la escala a utilizar, la variabilidad entre los resultados podría ser difícil de apreciar.

» Uso de gráficas de línea en lugar de gráficas de bloque: Ya que como se especificó, las gráficas de línea expresan una relación entre las variables y las de bloque no.

Bibliografía

Ciclo de vida de un software

- http://www.dynadata.com/var/www/vhosts/dynadata.com/httpdocs/avondov.ru/count29.php

Ergonomía y presentación de resultados

- http://www.computerworld.es/archive/ergonomia-del-software-mas-alla-del-interfaz-grafico
- http://www.insht.es/InshtWeb/Contenidos/Documentacion/TextosOnline/Guias_Ev_Riesgos/normastecnicaspvd.pdf
- http://es.kioskea.net/faq/1632-las-normas-en-ergonomia-de-software
- *ISO 13407 proceso de diseño centrado en el usuario para sistemas interactivos*
- http://webdiis.unizar.es/asignaturas/IPO/wp-content/uploads/2013/02/UNE-EN_ISO_134072000.pdf
- *ISO/TR 16982: Métodos de usabilidad que soportan diseño centrado en el usuario*
- http://www.iso.org/iso/catalogue_detail?csnumber=31176
- *ISO 9241-10: Principios para diálogos*
- http://www.iso.org/iso/catalogue_detail.htm?csnumber=16882
- *ISO 9241-11: Guía de especificaciones y medidas de usabilidad*
- http://www.iso.org/iso/catalogue_detail.htm?csnumber=16883
- *ISO 9241-12: Presentación de la información*
- http://www.iso.org/iso/catalogue_detail.htm?csnumber=16884
- *ISO 9241-13: Guía del usuario*
- http://www.iso.org/iso/catalogue_detail.htm?csnumber=16885
- *ISO 9241-14: Diálogos de menús*

- http://www.iso.org/iso/catalogue_detail.htm?csnumber=16886
- *ISO 9241-15: Diálogos de tipo lenguaje de órdenes*
- http://www.iso.org/iso/catalogue_detail.htm?csnumber=16887
- *ISO 9241-16: Diálogos de manipulación directa*
- http://www.iso.org/iso/catalogue_detail.htm?csnumber=16888
- *ISO 9241-17: Diálogos por cumplimentación de formularios*
- http://www.iso.org/iso/catalogue_detail.htm?csnumber=16889
- *ISO 14915: Ergonomía del software para interfaces de usuario multimedia*
- http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=25578
- *ISO/TS 16071: La accesibilidad a interfaces*
- http://www.iso.org/iso/catalogue_detail.htm?csnumber=30858