Diseño de interfaces web amigables desde la perspectiva de la semiótica organizacional*

[Web interfaces design from the perspective of the organizational semiotics]

Diego Samir MELO SOLARTE[1]

Recibo: 25.06.2012 - Aprobación: 26.10.2012

Resumen

Este artículo presenta una estrategia para el diseño de productos de software con interfaces web amigables, incorporando modelos de la Ingeniería de Software articulados con la Semiótica Organizacional, lo cual permite abstraer un problema desde diferentes perspectivas y plantear soluciones integrales donde se considere las características del usuario final. La estrategia planteada se basa en el diseño de prototipos no funcionales a nivel de mockups; los cuales permiten tener apreciación inicial del usuario a la vez que permite desenvolver libremente la creatividad del diseñador evitando limitarse por las competencias del personal encargado de la implementación. Por otra parte, en este artículo también se presenta dos modelos basados en inspección para la revisión y evaluación de interfaces, con los cuales se orienta a los encargados de un producto de software sobre los elementos relevantes que constituyen una interfaz usable y accesible.

Palabras clave: *Semiótica Organizacional, Interfaces Amigables, Método de Articulación de Problemas, Evaluación Heurística, GOMS, Elaboración de prototipos no funcionales.*

---

* Modelo para citación de este artículo de reflexión:
MELO SOLARTE, Diego Samir (2012). Diseño de interfaces web amigables desde la perspectiva de la Semiótica Organizacional. En: Ventana Informática. No. 27 (jul.-dic., 2012). Manizales (Colombia): Facultad de Ciencias e Ingeniería, Universidad de Manizales. p. 97-113. ISSN: 0123-9678

1 Ingeniero de Sistemas, Especialista en Telecomunicaciones, MSc. en Ciencias de la Computación. Docente investigador y Coordinador Unidad de Gestión Tecnológica CEDUM – Universidad de Manizales (Manizales, Caldas, Colombia). Correo electrónico: mdiego@umanizales.edu.co

Abstract

This article presents a strategy for the design of software products with friendly web interfaces including articulated Software engineering models with the Organizational Semiotic. This allows the abstraction of the problem from different perspectives and the planning of comprehensive solutions considering the final user's characteristics. The idea is based on the design of mockups. The mockups help to obtain the user's first appreciation and the designer's creativity fluency at the same time, avoiding the limits created by the competence of those people in charge of the implementation. On the other side, this article intends to show two models based on the inspection for revision and evaluation of interfaces to direct those in charge of the software product to outstanding elements that make a usable and accessible interface.

Keywords*: Organizational Semiotics, friendly interfaces, Problem Articulation Method, heuristic evaluation, GOMS, prototyping nonfunctional.*

Introducción

Con la masificación de las Tecnologías de la Información y la Comunicación (TIC), el desarrollo de productos de software ha adquirido una enorme relevancia en la sociedad, más aún, lograr que el producto cumpla con las expectativas y requerimientos planteados.

Construir un producto de software es una tarea dispendiosa y cada vez más exigente, para lo cual hoy en día se puede encontrar en el mercado diferentes metodologías basadas en la Ingeniería de software que facilitan modelos para lograr un proceso efectivo y en lo posible eficiente, sin embargo, en la vida real no siempre se logra el éxito en la construcción de software y muchas veces después de construir el producto, este no termina en producción debido a que los usuarios no se sienten confortables en el manejo o el producto final no cumple con los requerimientos planteados. Es allí que en este artículo se plantea una estrategia de desarrollo de software basada en la construcción de prototipos e interfaces, a la vez que se utiliza herramientas de la Semiótica Organizacional (SO) para analizar el problema o los requerimientos desde diferentes perspectivas, procurando abstraen posibles soluciones.

Por otra parte, en este artículo se plantea el diseño de interfaces considerando su usabilidad y accesibilidad, tratando de lograr interfaces amigable para el usuario final quién a lo largo de la construcción del

producto, se torna un agente activo, evaluando las propuestas y proponiendo nuevas ideas.

La elaboración de prototipos[2] no funcionales y el diseño de interfaces orientado al usuario, busca liberar la creatividad de los analistas, diseñadores y/o programadores, a fin de promover soluciones innovadoras, dejando de lado las competencias del equipo o las limitaciones tecnológicas. Si bien es cierto que los conocimientos del personal encargado del desarrollo y su experiencia juegan un papel importante, también es cierto que no se debe caer en soluciones cotidianas y facilistas.

Con la inclusión de la SO se busca innovar en las soluciones y agilizar los mecanismos de comunicación gráfica, tratando de construir interfaces intuitivas y pensadas en las características del usuario final.

Este artículo presenta en la primera unidad el referencial teórico base, en la unidad dos son presentadas algunas metodologías de evaluación de sistemas de información, la tercera unidad plantea una estrategia para el diseño y construcción de software de pequeña y mediana envergadura, la unidad cuatro presenta las conclusiones del trabajo.

1. Referencial teórico base

1.1 Semiótica organizacional

En muchas ocasiones se habla que el análisis de un sistema de información debe ser evaluado integralmente, sin embargo muchos modelos inclinan su estudio desde una perspectiva puramente técnica y operativa, dejando de lado al usuario como un agente activo dentro del proceso, y donde este agente está expuesto a una serie de influencias que puede afectar la interacción y la vez los resultados del sistema; es de esta manera que la Semiótica Organizacional busca crear un nuevo panorama del análisis de los sistemas de información basados en la comunicación como eje central de la interacción y donde las interpretaciones de los signos toman relevancia, de acuerdo con González (2008) y Gutiérrez (2012).

La SO se interpreta como una disciplina que estudia los procesos organizacionales, teniendo como base la naturaleza, características, funciones y efectos de la información y la comunicación en un contexto organizacional, según Stamper (1994), Stamper, Althaus & Backhouse (1988), Liu (2000) y Neris & Baranauskas (2010). Desde la perspectiva de la SO, una organización puede ver a la información como un sistema

2 En el texto original, el autor utiliza la expresión *prototipado*, término no aceptado por la RAE, por lo que se sustituye en todos los casos por *elaboración de prototipos* (Nota del editor).

donde los agentes emplean los signos para realizar unas acciones o propósitos específicos, afirman Neris & Baranauskas (2010, 14).

La figura 1 presenta las capas que hacen parte de la SO y los elementos que en cada una de ellas se pretende analizar sobre cualquier sistema de información, de acuerdo con Liu (2000, 29).

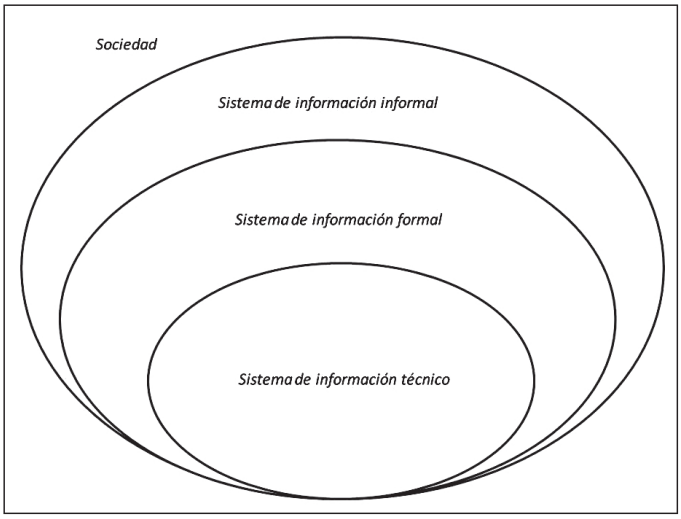

Figura 1. Cebolla de la Semiótica Organizacional (Liu, 2000, 29)

La capa interior está orientada al sistema de información técnico, donde se evalúan el software del sistema; la capa intermedia contempla el sistema de información formal evalúa los formularios y la reglas; la capa externa contempla el sistema de información informal que se encarga de evaluar métricas, intensiones, responsabilidades, creencias, entre otras. De esta manera el análisis de un sistema de información contempla diferentes perspectivas de evaluación a fin de lograr una visión integral del sistema.

Stamper (1994) y Stamper, Althaus & Backhouse (1988) han propuesto un conjunto de métodos para apoyar el análisis de un sistema de información desde la perspectiva de la SO llamados MEASUR (*Methods for Eliciting, Analysing and Specifying Users Requirements*), presentado por Liu (2000, 7-8). Entre los cuales se tiene el Método de Articulación de Problemas, que permite evaluar un sistema de información como un problema desde sus estados iniciales de entendimiento, cuando sus fronteras aún no son claras y la solución debe considerar los aspectos sociales, pragmáticos y semánticos involucrados, además de la infraestructura necesaria (aspectos físicos, empíricos y sintácticos).

1.1.1 Método de Articulación de Problemas (PAM) La articulación compartida de problemas es transversal a todo el proceso, está inspirada en métodos y artefactos de la Semiótica Organizacional, adaptados para su uso en el contexto colaborativo a distancia, según Melo-Solarte & Baranauskas (2009, 22). La SO es considerada una disciplina de la Semiótica particularmente relacionada con el tratamiento de la información en sus varios niveles sígnicos en los grupos sociales y de esta forma se pretende darle otra mirada a los sistemas de información, tratando de profundizar en su interpretación y la construcción de soluciones.

• **Análisis de Partes Interesadas.** Ayuda al grupo de participantes a entender la situación real del problema y los requisitos de la solución pretendida, por medio de la discusión y levantamiento de las partes que directa o indirectamente influyen o sufren influencia de la solución; aplicar este artefacto (Figura 2) ha permitido generar una lluvia de ideas colectiva que vislumbra posibles caminos a seguir y posibles alternativas de solución, pues es evidente que solo el planteamiento de un problema deja a los estudiantes desorientados y sin rumbo claro para comenzar a trabajar.

A continuación se presenta brevemente una descripción de las capas que hacen parte del análisis de Partes Interesadas:

- Operación: indica el sistema o problema que está siendo sometido a evaluación.

- Contribución: está orientado a determinar los actores y/o responsables que contribuyen o son afectados directa o indirectamente por el sistema o problema.

- Fuente: permite identificar clientes, proveedores y todos aquellos actores que hacen uso de los datos y las informaciones del problema o de la solución.

- Mercado: permite identificar cuáles pueden ser los posibles aliados o la competencia que hace parte del mercado asociado al problema o solución.

- Comunidad: está orientado a identificar los espectadores, legisladores y todos los elementos que representan los elementos de la comunidad que influencian o son influenciados por el problema o solución en un determinado contexto social.

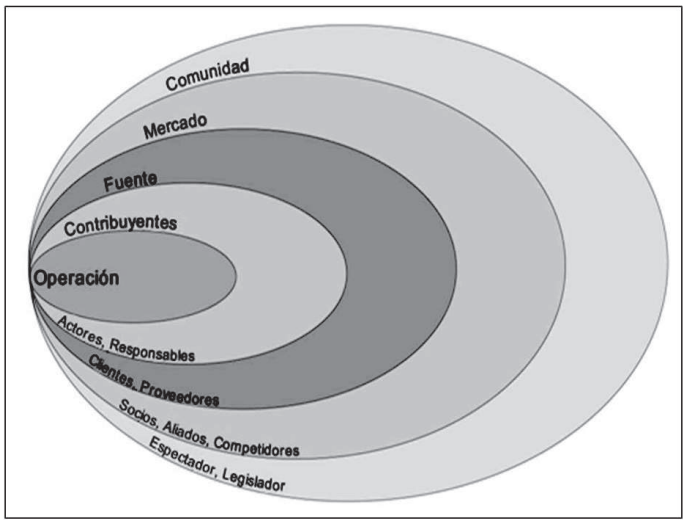

Figura 1. Artefacto - Partes Interesadas

• **Cuadro de evaluación.** Permite al grupo de estudiantes identificar intereses, dificultades, restricciones y principales problemas para cada parte interesada, tratando de discutir posible soluciones que impactarán en los requisitos del problema y en la solución general del mismo. Este artefacto ayuda a la comprensión del problema en sus estados iniciales de búsqueda de soluciones, apoyándose en el intercambio, así como a compartir significados entre los participantes.

El Cuadro de Evaluación permite identificar para cada una de las partes interesadas, cuáles son sus principales intereses, preguntas o problemas, para discutir posibles ideas o soluciones, de las cuales resultarán algunos requisitos de usuario con impacto en la propuesta final de la solución. El Cuadro de evaluación está compuesto por tres componentes, el primero representa las categorías sobre las cuales están siendo clasificadas las Partes Interesadas; el segundo elemento permite registrar preguntas o problemas identificados en cada una de las Partes Interesadas; y el tercer elemento permite registrar las posibles ideas o soluciones sobre los problemas levantados en este análisis.

• **Escala Semiótica.** Desde la perspectiva semiótica varios niveles de significados deben ser consideradas en la discusión de un problema. El Framework Semiótico de Stamper (1994) o escalera semiótica, está compuesto por seis capas o escalas, a saber:

- Física: esta capa tiene el objetivo de identificar los requisitos asociados al contexto físico, como son todos los componentes que serán usados

para crear la solución, sus propiedades, su tamaño, su formato, los recursos, el hardware requerido, etc.

- Empírica: esta capa tiene por objetivo ayudar a identificar los requisitos asociados a la gestión de datos sin interpretación de significado, como es el canal de comunicación, el ruido, la redundancia, la eficiencia, etc.

Sintáctica: permite analizar las estructuras de lenguaje complejas, sin considerar sus significados, permite levantar los requisitos del problema asociados a la estructura, forma, datos, patrones, etc.

- Semántica: a partir del concepto de *significado comportamental* (Stamper, 1994), esta capa permite levantar requisitos asociados a los significados, suposiciones e validez del problema y sus posibles soluciones.

- Pragmática: esta capa está enfocada en identificar los requisitos asociados al comportamiento, comunicación y relaciones entre las personas y el problema y sus posibles soluciones.

- Mundo social: en esta capa se pretende identificar los requisitos asociados al problema o sus posibles soluciones en un contexto social, como son las leyes, los contratos, las creencias, etc.

1.2 Metodologías para desarrollo de software

Una metodología de desarrollo de software se puede interpretar como un conjunto de pasos formales que se deben seguir para lograr la correcta construcción de un producto de software, entre dichos pasos están: dividir un producto en etapas, determinar las tareas a realizarse en cada etapa, evaluar requerimientos, analizar y diseñar posibles soluciones, determinar las herramientas a usar, clasificar el equipo de trabajo y determinar sus responsabilidades, pactar tiempos de producción, documentar el proceso, entre otros. Estos formalismos vienen de la Ingeniería de Software con el fin de garantizar el correcto desarrollo de un producto (Pressman, 2005); sin embargo, paulatinamente se ha identificado que proyectos de software de pequeña o mediana envergadura no tienen que estar sujetos a un cúmulo de formalismos y actividades desgastante, ya que en ellos impera el resultado final, claro está, sin descuidar la calidad.

De esta forma y atendiendo las necesidades actuales, se han abierto puertas para nuevos modelos que buscan agilizar la producción ejecutando y regulando aquellas actividades que sean realmente necesarias y útiles en el proceso, dando origen a las denominadas *metodologías ágiles* de desarrollo de software, algunas de las cuales se describen a continuación.

1.2.1 Metodologías ágiles. Los métodos ágiles son una respuesta a la gran demanda por métodos y modelos de procesos que sean flexibles a

los constantes cambios en el desarrollo de un determinado proyecto de software, y a la entrega rápida del mismo, sin dejar de lado la calidad.

En el manifiesto para el desarrollo ágil de software, Beck et al. (2001), define las prioridades para los procesos basados en métodos ágiles: «*Individuos e interacciones antes que procesos y herramientas, Software en funcionamiento antes que grandes documentos, Colaboración con el cliente antes que negociación de contratos, Responder a cambios antes que seguir un plan*».

En la actualidad se encuentra una gran cantidad de métodos ágiles disponibles a la orden del día, estos métodos pueden variar de manera significativa, por tal razón es necesario escoger cuidadosamente y ceñido a las características del proyecto de software, algunos de los principales ejemplos de métodos ágiles y más populares según Koch (2005):

- Adaptive Software Development (ASD): los equipos del proyecto de software a su interior son vistos como un sistema adaptativo complejo, compuesto por agentes, el entorno con su incidencia y los resultados emergentes.

- Dynamic System Development Method (DSDM): los detalles sobre la creación documental del software son dejados relativamente indefinidos y el método se enfoca principalmente en fortalecer el proceso de desarrollo de software, es decir, la documentación pasa a un segundo plano.

- eXtreme Programming (XP): este método está orientado a la entrega de pequeños desarrollos con el fin de promover las respuestas rápidas a los cambios sobre los requerimientos. Es uno de los métodos más conocidos y su esquema de trabajo se ha tornado referente para el desarrollo de muchos proyectos ágiles.

- Scrum: es una metodología para el gerenciamiento del desarrollo de un producto que puede ser adecuado a cualquier tecnología específica. Lo que plantea *scrum* es seccionar un producto en pequeños fragmentos para llegar a determinar el trabajo hasta un nivel de control.

- AIPM: un modelo de proceso ágil desarrollado en el contexto del proyecto *e-Cidadania*, este modelo busca integrar las metodologías ágiles y las técnicas de la interacción humano computador, de la semiótica organizacional, el diseño universal, el trabajo colaborativo, entre otros.

Cabe anotar, que en la Ingeniería de Software tanto en sus metodologías tradicionales como en las metodologías ágiles, prima el aspecto funcional de un sistema de información sobre las estrategias de comunicación e interacción, es por eso que de cierta forma se descuida el pensamiento y conocimiento que puede tener un usuario final a la hora de interactuar con un sistema de información.

2. Evaluación de usabilidad y accesibilidad de las interfaces

La evaluación de la usabilidad y accesibilidad de una interfaz, es un aspecto fundamental que se debe tener en cuenta en los sistemas interactivos, aseguran Juristo et al. (2007, 543). La evaluación permite estudiar, valorar y validar un sistema a través de la aplicación de diferentes métodos que pueden ser aplicados en diferentes momentos.

La evaluación de un sistema de información a través de sus interfaces de usuario, de acuerdo con Lorés, Sendín & Agost (2001, 6), busca identificar la calidad de la interacción y la funcionalidad que conlleva el uso de un determinado aplicativo desde el punto de vista de quien será la persona que lo usará, afirman Almeida & Baranauskas (2008).

De esta manera, se trata de identificar que el sistema cumpla con los objetivos planteados pero además se busca que dicho sistema pueda ser usado por un gran número de usuarios lo que lleva al concepto de diseño universal interpretado como:**«***el proceso de diseñar productos que sean usables por el rango más amplio de personas, funcionando en el rango más amplio de situaciones y que es comercialmente practicable»* Abascal & Valero (2001, 5) Si se considera la evaluación de las interfaces desde el comienzo del diseño y donde se plantea como objetivo central la manera como el usuario final interactuará, es posible referirse, a un esquema de producción y construcción denominado *diseño centrado en el usuario* el cual se guía por tres pilares que son: la ingeniería de software, el diseño y elaboración de prototipos, finalmente está la evaluación del producto.

Entre las metodologías más comúnmente usadas para evaluar interfaces están aquellas denominadas metodologías de evaluación por inspección entre las cuales se tienen: la evaluación heurística y la evaluación conocida como GOMS que se orienta especialmente a medir el rendimiento y desempeño contemplando metas, operadores, métodos y reglas de un sistema.

2.1 Evaluación heurística

La evaluación heurística propuesta por Nielsen & Molich (1990, 252), consiste en evaluar las interfaces de un sistema de información desde el punto de vista de los principios de usabilidad, utilizando la técnica de inspección realizada por evaluadores expertos, quienes se destacan especialmente por su experiencia y conocimientos para detectar problemas de usabilidad y accesibilidad.

Teniendo en cuenta que todas las metodologías son susceptibles de errores, el uso de varios evaluadores para aplicar este esquema, trata de evitar que un evaluador genere conceptos sesgados a la vez que se busca producir discusiones entre evaluadores que permitan generar ideas y retroalimentar a los diseñadores y desarrolladores para mejorar las interfaces propuestas; sin embargo por datos encontrados en la literatura, se ha encontrado que se recomienda usar como mínimo tres evaluadores y un máximo de cinco, incrementar este número de evaluadores no garantiza la obtención de mejores resultados.

En la evaluación heurística, cada evaluador revisa y analiza las interfaces de un sistema de información basados en diez principios heurísticos de usabilidad que plantean los focos de análisis y que a través de un conjunto de preguntas buscan orientar el proceso de evaluación. Cabe anotar que los principios heurísticos y las preguntas realizadas en cada uno de ellos, permiten tener una guía sobre los factores que se deben tener en cuenta, sin embargo no se pretende lograr un resultado cuantitativo de cada principio sino un resultado cualitativo con soporte y evidencia.

A continuación son presentados los principios heurísticos de Nielsen & Molich (1990, 252) basados en el análisis de usabilidad de proyectos de diseño centrados en el usuario. La evaluación heurística se lleva a cabo realizando por parte de cada evaluador una revisión de la interfaz. Cuando se han terminado todas las evaluaciones se permite a los evaluadores comunicar los resultados y sintetizarlos.

2.1.1 Principios heurísticos de usabilidad. De acuerdo con González, Pascual & Lorés (2001, 8-9), consisten en:

- Visibilidad del estado del sistema. El sistema siempre deberá mantener informados a los usuarios sobre el estado del sistema, presentando un esquema de *feedback* o retroalimentación adecuada en todo momento y ante toda interacción del usuario.
- Utilizar el lenguaje de los usuarios. Se debe buscar que el sistema use un lenguaje fácil de interpretar por los usuarios, libre de tecnicismos, textos en otros idiomas o incluso expresiones regionales que no todos los usuarios podrán entender, además se sugiere utilizar convenciones del mundo real, fáciles de interpretar y que contengan un orden lógico.
- Control y libertad para el usuario. El usuario es quién tiene el control sobre el accionar del sistema, de esta manera él siempre deberá tener alternativas de ejecutar algunos procesos o incluso cancelarlos retornando a su estado previo.

- Consistencia y estándares. Las interfaces deben evitar la posibilidad de inconsistencias entre la información textual, gráfica y funcional, de esta manera se debe construir interfaces que aprovechen la intuición y el conocimiento previo del usuario antes que forzar un nuevo aprendizaje.
- Prevención de errores. Las interfaces deben validar y guiar las diferentes acciones de los usuarios, tratando en lo posible de evitar que el usuario cometa errores involuntarios.
- Minimizar la carga de la memoria del usuario. El diseño de una interfaz no debería hacer que un usuario tenga que memorizar un elemento o un dialogo previo para poder ejecutar una acción. De esta manera es conveniente mantener siempre visible los objetos y los textos para que el usuario pueda convalidar sus acciones.
- Flexibilidad y eficiencia de uso. Se debe mantener siempre visible la información relacionada con el uso del sistema a la vez que este tipo de información debe ser de fácil accesibilidad; se trata de guiar al usuario inexperto para que tenga igual desempeño que el usuario experto.
- Los diálogos estéticos y diseño minimalista. Las interfaces no deben presentar información innecesaria o que sea usada raras veces, siempre se debe resaltar la información relevante guardando equilibrio con la información complementaria.
- Ayudar a los usuarios a reconocer y recuperarse de los errores. Los errores deben presentarse con un lenguaje claro e indicar con exactitud el problema y en lo posible las circunstancias que lo ocasionaron.
- Ayuda y documentación. El sistema debe tener disponible la documentación necesaria que permita guiar al usuario en cuanto al manejo e identificación de las funciones disponibles, varios autores recomiendan que la documentación debe ser muy concreta y no muy extensa.

Estos principios de evaluación heurística son guiados a través de un cúmulo de preguntas, como la lista de comprobación de Pierotti (2004), que permiten caracterizar cada heurística tornando más objetiva la apreciación de un experto, a la vez que agrega tres principios heurísticos más: habilidades, interacción placentera y respetuosa con el usuario y finalmente el principio de privacidad.

2.2 Evaluación GOMS

La metodología de evaluación GOMS (*Goals, Operators, Methods and Selection rules*) está orientada a evaluar el trabajo que implica

desarrollar determinada acción a través de una interfaz considerando los parámetros de eficiencia y eficacia que ello implica; la esencia del modelo GOMS radica en contabilizar los tiempos que requiere un usuario para ejecutar una determinada función posibilitada por las interfaces.

La estructura GOMS, de acuerdo con Kieras et al. (1995), consiste en:

- Objetivos: planteamiento de las metas que se pretende lograr con el usuario;
- Operadores: acciones básicas que se tienen que ejecutar para cumplir con los objetivos planteados;
- Métodos: indica el mecanismo usado para ejecutar las acciones ya sea teclado, mouse o pantalla táctil;
- Reglas de selección: selección de posibles alternativas que permiten el alcance del objetivo.

En términos generales, GOMS consiste en plantear un objetivo para que los usuarios lo puedan cumplir y a partir de allí el equipo evaluador se debe encargar de supervisar todas las actividades que desarrolla un usuario y los caminos que elije para cumplir con el objetivo planteado, Claros & Collazos (2006); los tiempos propuestos para evaluar la usabilidad de una interfaz son Haunold & Kuhn (1994):

- K = *Keying* (tecleando): este parámetro contabiliza el tiempo que le toma a un usuario para oprimir una tecla o un conjunto de teclas que desencadenan una acción;
- P = *Pointing* (apuntando): este parámetro contabiliza el tiempo que le toma a un usuario para posicionar el cursor en un determinado lugar de la interfaz;
- H = *Homing* (mover la mano): calcular el tiempo que le toma a un usuario para mover la mano y posicionarla sobre un dispositivo que habilita la acción;
- M = *Mentally preparing* (preparación mental): calcular el tiempo que le toma a un usuario para prepararse para el siguiente paso;
- R= Responding (respondiendo): El tiempo que un usuario tiene que esperar para obtener un *feedback* del sistema.

Los tiempos promedio logrados por los usuarios para cumplir un objetivo permiten determinar la complejidad de uso que puede tener una interfaz, de esta manera se logra hacer una evaluación de interfaces aproximándose al uso real del sistema, sin embargo este tipo de evaluación en interfaces complejas resulta un proceso dispendioso.

3. Estrategia y herramientas para construcción de interfaces amigables

Si bien es cierto que no hay una receta mágica para garantizar el correcto desarrollo de un producto de software, existen metodologías basadas en la ingeniería de software y/o la interacción humano computador que orientan el desarrollo mediante directrices y modelos, sin embargo, en esta unidad se plantea una estrategia de desarrollo de software basada en la práctica y experiencia para proyectos de pequeña o mediana envergadura, que puede de cierta forma, ayudar a los nuevos desarrolladores en el análisis de un problema y el construcción de propuestas consistentes.

Los formalismos planteados por la ingeniería de software para la producción de software son totalmente válidos y no es objetivo de esta unidad ir en contra de ellos, pero si tomar como base a las metodologías ágiles y el diseño basado en prototipos, a partir de allí reestructurar la abstracción de un problema tratando de tener una visión basada en el usuario final, considerando sus necesidades, sus capacidades y sus limitaciones; en definitiva es el usuario final quien va a interactuar directamente con el sistema y por lo tanto su apreciación es alta relevancia.

A continuación se presentan las estrategias con cada uno de los elementos sugeridos para estudiar adecuadamente un problema y tratar de proponer soluciones prácticas, pero que cumplan satisfactoriamente las expectativas de un cliente:

- Evaluación y especificación de requerimientos: uno de los principales inconvenientes o errores que suele cometer un analista o desarrollador, es dar por entendido un requerimiento y no profundizar lo suficiente para lograr interpretarlo en detalle. Es importante en esta etapa levantar los requerimientos, detallarlos en minucia y luego evaluar su viabilidad.
- Análisis del sistema: el análisis del sistema es la etapa donde se estudia las diferentes características técnicas y funcionales de un sistema o de un problema planteado, típicamente son usados los modelos de datos y funcionales; sin embargo en esta propuesta se plantea el uso de la SO y el PAM como un mecanismo para tener una perspectiva complementaria, considerando la semántica, la pragmática y las implicaciones sociales de un sistema. La SO y el PAM cuentan con un conjunto de herramientas que ayudan a los analistas el levantamiento de los datos y la interpretación de los mismos, esperando que con ello se logre incidir en la solución.

- Diseño y elaboración de prototipos no funcionales: en esta etapa se comienza el proceso de construir una solución, sin embargo se plantea la construcción de una solución gráfica inicial, basada en la arquitectura del sistema que determinará los diferentes elementos que intervendrán en la solución y en la construcción de las interfaces que se podría considerar como un prototipo no funcional, con el cual se trata de describir las funcionalidades del sistema a la vez que se estructura el esquema de comunicación e interacción. Las interfaces pueden ser planteadas a nivel de *mockups*, que son diseños no perfeccionados pero que ya conllevan una distribución de los elementos a nivel de interfaz y como aquellos elementos pueden ser accionados, algunas herramientas de utilidad para esta actividad pueden ser *Balsamiq* o *Cacoo* que contienen un pool de elementos gráficos que pueden agilizar el diseño. Si se desea lograr una mayor evolución de las interfaces, se puede requerir la participación de un diseñador gráfico quién será el encargado de migrar un *mockup* a un diseño más realista, pero conservando las características planteadas funcionales o distributivas del *mockup*.
- Evaluación del prototipo no funcional: en este momento se puede hablar del primer encuentro del cliente con lo que posteriormente será el sistema, la primera impresión del cliente permitirá lograr una apreciación positiva sobre la propuesta o descartarla antes de generar un mayor desgaste. La evaluación del prototipo es básicamente un análisis general de los requerimientos y como ellos están plasmados en las interfaces propuestas, para que en este momento el cliente valide la propuesta, plantee los ajustes necesarios o descarte completamente la propuesta; una primera evaluación puede ser de tipo *inspección* guiada por la evaluación heurística de usabilidad.
- Implementación: una vez aprobado el prototipo no funcional se puede pasar a la codificación para tornar reales y funcionales dichas interfaces, es de anotar que al desarrollador le cabe una alta responsabilidad de construir un sistema a partir de unos planos iniciales, los cuales serán guía y argumento del producto de software. El analizar un problema desde diferentes perspectivas y plantear una solución gráfica a través de interfaces pretende dejar en libertad la parte creativa de los diseñadores, pensando en una solución innovadora y no en una solución limitada por las tecnologías, siendo así, es obligación de programador construir lo diseñado y no plantear nuevas soluciones acordes a su conocimiento. Entre las tecnologías

que pueden contribuir a una buena implementación de interfaces web se tienen: los lenguajes basados en etiquetas HTML y XML, las hojas de estilo CSS, el lenguaje de programación *Javascript*, el *framework JQuery*, entre otros.

- Evaluación, validación y verificación: una vez construido el producto de software es necesario pasar por diferente procesos que permitan garantizar la calidad del mismo, teniendo en cuenta que se debe garantizar el cumplimiento de los requerimientos planteados al comienzo del proyecto y que el producto debe estar acorde al prototipo diseñado. En esta etapa se puede usar también la evaluación heurística para determinar la usabilidad del sistema a la vez que se puede utilizar GOMS para evaluar el desempeño del sistema.
- Puesta en producción: un producto de software después de haber pasado la etapa de verificación y validación, puede entrar en producción, teniendo en cuenta que se requiere hacer un seguimiento continuo durante cierto tiempo para determinar si se requerirán ajustes. Por otra parte, aunque desde el principio se pensó en la construcción de un producto de software basado en el usuario, utilizando interfaces amigables, es necesario realizar un acompañamiento continuo hasta que el usuario logre las competencias necesarias para dominar el sistema.

4. Conclusiones

- El éxito de un sistema de información en gran parte, depende de la calidad de sus interfaces medidas a partir de su usabilidad y accesibilidad, sin descuidar la estética que torna a una interfaz visualmente atractiva a la vez que busca ser intuitiva.
- Existe un gran reto para los analistas, diseñadores y desarrolladores de software, el reto está enfocado en complementar la parte técnica que implica la construcción de una interfaz con otros elementos que pueden ser considerados para enriquecer un determinado diseño como son: aspectos sociales, culturales, cognitivos, económicos, legales, entre otros. La semiótica organizacional puede ser de gran ayuda para analizar y construir interfaces que consideren dichos elementos.
- El diseño centrado en el usuario final busca considerar diferentes cualidades físicas, cognitivas, culturales y sociales, para la construcción de interfaces inclusivas, usables y accesibles; aunque no se puede pensar en una solución plena y perfecta, si se puede lograr una interfaz efectiva, eficiente e incluyente.

- La elaboración de prototipos de productos de software es una buena alternativa para contextualizar a los usuarios o clientes sobre la manera como podrán interactuar con el sistema de información, a la vez que evalúan la consistencia de las funcionalidades planteadas.

Bibliografía

ABASCAL, J. & VALERO, P. (2001). Accesibilidad. En: LORÉS, J. (ed.). La interacción persona-ordenador [en línea]. Lleida (España): Asociación Interacción Persona-Ordenador. Licencia de Creative Commons. <http://www.aipo.es/libro/pdf/07Accesi.pdf> 14 p. [consulta: 17/09/2012]

ALMEIDA, L.D.A. & BARANAUSKAS, M.C.C. (2008). Awareness em sistemas colaborativos: Conceitos e desafios [em línea]. Em: Relatório Técnico IC-08-23 (sep). Campinas (Brasil): Instituto de Computação, Universidade de Campinas. 17 p. <http://www.ic.unicamp.br/~reltech/2008/08-23.pdf> [12/09/2012]

BECK, K.; BEEDLE, M.; VAN BENNEKUM, A.; COCKBURN, A.; CUNNINGHAM, W.; MARTIN FOWLER, M.; GRENNING, J.; HIGHSMITH, J.; HUNT, A.; JEFFRIES, R.; KERN, J.; MARICK, B.; MARTIN, R.C.; MELLOR, S.; SCHWABER, K.; SUTHERLAND, J. & THOMAS, D. (2001). Manifesto for Agile Software Development [on line]. Portland (Oregon, USA): Q7 Enterprises, Inc. <http://agilemanifesto.org/> [12/07/2012]

CLAROS, I.D. & COLLAZOS, C.A. (2006). Propuesta Metodológica para la Evaluación de la Usabilidad en Sitios Web: Experiencia Colombiana [en línea]. Popayán (Colombia): Universidad del Cauca, Facultad de Ingeniería Electrónica y telecomunicaciones. <http://artemisa.unicauca.edu.co/~iclaros/usabilidad/descargas/paperInteraccion2006.pdf> [consulta: 20/07/2012].

GONZÁLEZ PÉREZ, C.F. (2008). Aportes de la Semiótica Cognitiva a los estudios de la Comunicación Organizacional [en línea]. En: XII Jornadas Nacionales de Investigadores en Comunicación: Nuevos escenarios y lenguajes convergentes. Rosario (Santa Fe, Argentina): Red Nacional de Investigadores en Comunicación. Memorias de las Jornadas Nacionales de Investigadores en Comunicación No. 12. ISSN: 1852-0308 <http://www.redcomunicacion.org/memorias/p_jornadas_p.php?id=397&idj=4> [consulta: 10/07/2012]

GONZÁLEZ, M.P.; PASCUAL, A. & LORÉS J. (2001). Evaluación heurística. En: LORÉS, J. (ed.). La interacción persona-ordenador [en línea]. Lleida (España): Asociación Interacción Persona-Ordenador. Licencia de Creative Commons. <http://www.aipo.es/libro/pdf/15-Evaluacion-Heuristica.pdf> 39 p. [consulta: 17/09/2012]

GUTIÉRREZ, L. M. (2012). Identidad e Imagen Corporativas: Construcciones Simbólicas Para la Comunicación. En: Boletín Red Iberoamericana de Pedagogía REDIPE, Boletín 806, (ene. 30). Cali (Colombia): Red Colombiana de Pedagogía. 19 p. ISSN: 2256-1536. <http://www.rediberoamericanadepedagogia.com/index.php/articulos/boletin-806/identidad-e-imagen-corporativaspdf/download> {consulta: 15/09/2012]

HAUNOLD, P. & KUHN, W. (1994). A Keystroke Level Analysis of a Graphics Application: Manual Map Digitizing. In: Conference on Human factors in computing systems celebrating interdependence, CHI'94 (24-28/04/1994), Boston (Massachusetts, USA): ACM. ADELSON, B.; DUMAIS, S. & OLSON, J.S. (Eds.). Proceedings of the CHI'94, New York (NY, USA): ACM, p. 337-343. ISBN: 0-89791-650-6. <http://ifgi.uni-muenster.de/~kuhn/research/publications/pdfs/refereed%20conferences/CHI%201994.pdf> [consult: 19/09/2012]

JURISTO, N.; MORENO, A.; SÁNCHEZ-SEGURA, M. I. & BARANAUSKAS, M. C. C. (2007). A Glass Box Design: Making the Impact of Usability on Software Development Visible [on line]. Lecture Notes in Computer Science, Vol. 4663, No. 2, Berlin (Germany): Springer, p. 541-554. ISSN: 0302-9743. <http://www.springerlink.com/content/t7527u5703885745/fulltext.pdf> [consult: 20/09/2012]

KIERAS, D.E; WOOD, S.D; ABOTEL, K. & HORNOF, A. (1995) GLEAN: A Computer-based tool for rapid GOMS model usability evaluation of user interface designs. In: 8th annual ACM symposium on User interface and software technology, UIST'95 (14-17/11/1995), Pittsburgh

(PA, USA): ACM. Proceedings of the UIST`95. New York (NY, USA): ACM. p. 91-100. ISBN: 0-89791-709-X

KOCH, A.S. (2005). Agile Software Development: Evaluating the Methods for Your Organization. Norwood (USA): Artech House. 280 p. ISBN: 978-1580538428

LIU, K. (2000). Semiotics in Information Systems Engineering. Cambridge (UK): Cambridge University Press. 232 p. ISBN: 0-521-59335-2

LORÉS, J.; SENDÍN, M. & AGOST. J (ed.) (2001). Evaluación [en línea]. En: LORÉS, J. (ed.). La interacción persona-ordenador [en línea]. Lleida (España): Asociación Interacción Persona-Ordenador. Licencia de Creative Commons. <http://www.aipo.es/libro/pdf/04Evalua.pdf> 31 p. [consulta: 10/08/2012]

MELO SOLARTE D. S. & BARANAUSKAS M. C. (2009). Aprendizagem Colaborativa Baseada em Problemas – ACBP: Modelo conceitual e ferramentas, Dissertação de mestrado (Mestre em Ciência da Computação). Campinas (Brasil): Instituto de computação, Universidade de Campinas.

NERIS, V.P. & BARANAUSKAS, M.C.C. (2010). Estudo e Proposta de um Framework para Design de Interfaces de Usuário Ajustáveis, Tese doutorado (Doutor em Ciência da computação), Campinas (Brasil): Instituto de computação, Universidade de Campinas.

NIELSEN, J. & MOLICH, R. (1990). Heuristic evaluation of user interfaces. In: conference on Human factors in computing systems: Empowering people CHI'90. (01-05/04/1990), Seattle (WA, USA): ACM. CARRASCO, J & WHITESIDE, J. (eds) Proceedings of the CHI '90. New York (NY, USA): ACM, p. 249-256. ISBN: 0-201-50932-6.

PIEROTTI, D. (2004) Heuristic Evaluation: a System Checklist [on line]. Fairfax (VA, USA): Society for Technical Communication. <http://www.stcsig.org/usability/topics/articles/he-checklist.html> [12/07/2012]

PRESSMAN R. S. (2005). Software Engineering; a Practitioner Approach, 6 ed. New York (NY, USA): McGraw-Hill International. 880 p. ISBN: 978-0073019338

STAMPER, R. (1994). Social norms in requirements analysis: an outline of MEASUR. In: JIROTKA, M. & GOGUEN, J. (1994). Requirements Engineering: Social and Technical Issues. San Diego (CA, USA): Academic Press Professional. p. 107-139. ISBN: 0-12-385335-4

STAMPER, R.K.; ALTHAUS, K.E. & BACKHOUSE, J. (1988). MEASUR: Method for Eliciting, Analizing and Specifying User Requirements. In: Working Conference on Computerized Assistance During the Information Systems Life Cycle, CRIS 88 IFIP WG 8.1 (19-22/09/1988), Egham (UK): International Federation For Information Processing, IFIP. OLLE, T.W.; VERRIJN-STUART, A.A. & BHABUTS, L. (eds). Proceedings of the IFIP WG 8.1. North-Holland (Holland): Elsevier Science Publishers. ISBN 0-444-70512-0