Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación

Lecturas en Ciencias de la Computación
ISSN 1316-6239

AgilUs: un método ágil de desarrollo de software que incorpora la usabilidad

Alecia Eleonora Acosta

RT 2011-02

Centro de Ingeniería de Software y Sistemas
ISYS - UCV
Caracas, Septiembre, 2011.

AgilUs: un método ágil de desarrollo de software que incorpora la usabilidad

Alecia Eleonora Acosta

Centro de Ingeniería de Software y Sistemas (ISYS)

Escuela de Computación - Facultad de Ciencias - Universidad Central de Venezuela, Venezuela

eleonora.acosta@ciens.ucv.ve

Resumen

En la actualidad, el desarrollo de aplicaciones útiles, usables y agradables a los usuarios constituye uno de los grandes retos de la Ingeniería de Software y la Interacción Humano Computador, y conlleva a grandes beneficios tales como el incremento de la productividad, la satisfacción y la reducción de costos de entrenamiento y soporte del software. En este trabajo se propone un método de desarrollo ágil que permite la construcción de la usabilidad desde las primeras etapas del ciclo de vida del software, y se presenta dos casos de estudio que ha permitido validar este método, a través de las evaluaciones de usabilidad realizadas durante el ciclo de vida del software. Las técnicas de evaluación de usabilidad propuestas en el método no son costosas ni requieren de infraestructura tecnológica compleja. Cabe señalar que este método está orientado a desarrollos de aplicaciones que involucren un alto grado de interacción con los usuarios.

Palabras Clave: Método ágil de desarrollo de Software, usabilidad, evaluación de usabilidad.

1 Introducción

En los tiempos actuales, es cada vez más importante el desarrollo de las interfaces de usuarios usables, lo cual hace que surjan nuevos tópicos de investigación en el área de la Interacción Humano Computador, que involucran grupos multidisciplinarios de desarrollo, y donde se requiere producir lo más rápido posible debido a que hay mucha competencia en el mercado. En este sentido, el diseño de una interfaz de usuario usable se ha vuelto un aspecto fundamental para las aplicaciones interactivas, así como la entrega rápida de nuevos desarrollos de software. El desarrollo de aplicaciones útiles, usables y agradables a los usuarios conlleva a grandes beneficios tales como el incremento de la productividad, la satisfacción y la reducción de costos de entrenamiento y soporte del software.

Se platean métodos de desarrollo de software que buscan agilizar la producción de software, tal como el enfoque ágil. Sin embargo, no se incluye en forma explícita la evaluación de la usabilidad de los productos de software a ser desarrollados. Algunos autores, como Nielsen [1] y compañías como ThoughtWorks [2], han descrito experiencias en cuanto a la incorporación de la usabilidad en el proceso de desarrollo de software, dando algunas recomendaciones para tal fin; pero sería conveniente precisar algunas técnicas a utilizar en cada etapa del desarrollo.

Son muchos los métodos propuestos para el desarrollo de software y son diversas las herramientas y técnicas con que cuentan los ingenieros de software para realizar su trabajo. Quizás no es el mismo caso de los especialistas en Interacción Humano-Computador, entendiendo que esta disciplina no tiene la misma madurez que la anterior; sin embargo, se tienen técnicas, guías y recomendaciones, entre otros mecanismos que permiten realizar las evaluaciones a fin de desarrollar software usable. El objetivo de esta investigación es proponer algún mecanismo de integración entre estas dos disciplinas que permita la producción de sistemas usables. Se propone un método que provea un conjunto de "buenas prácticas" para el desarrollo de software desde una perspectiva ágil, con la finalidad de obtener un producto usable. Se plantea el desarrollo de un sistema partiendo de un prototipo de la interfaz de usuario, y que incorpora la aplicación de diversas técnicas de evaluación de usabilidad desde el inicio del ciclo de vida de la aplicación.

Considerando que la esencia de los enfoques ágiles es el intercambio continuo y constante con los usuarios, y que ese también es el principio del diseño centrado en el usuario, el método propuesto hace énfasis la obtención de ese *feedback* desde etapas tempranas del desarrollo, con la aplicación de tormentas de ideas, encuestas y prototipos en papel, entre otras, a lo largo de la evolución del software hasta la obtener el producto final, incluida la entrega del producto. Así los resultados de estas evaluaciones permitirán asegurar la obtención de un producto usable.

La sección 1expone los conceptos fundamentales de esta investigación: usabilidad, diseño centrado en el usuario y evaluación de usabilidad. La sección 2 presenta un resumen de los aspectos básicos acerca de los métodos ágiles de desarrollo de software. La sección 3 describe la propuesta concreta del método AgilUs y la sección 4 muestra los resultados obtenidos en dos casos de estudio en cuyos desarrollos se aplicó el método propuesto.

2 Usabilidad

En general, la finalidad de los sistemas informáticos interactivos consiste en lograr que el usuario haga lo que tiene o quiere hacer, o encuentre lo que busca en el menor tiempo posible y en forma satisfactoria. Esto es posible cuando los sistemas son usables. A continuación se presenta algunas definiciones de usabilidad, el enfoque de diseño centrado en el usuario y diversas técnicas de evaluación de usabilidad que permiten determinar el grado de usabilidad de los sistemas. Estas evaluaciones pueden ser llevadas a cabo por los usuarios y/o especialistas en Interacción Humano Computador.

2.1 Definición

Intuitivamente, la usabilidad puede definirse como la facilidad de uso de cualquier sistema computacional con el cual interactúe el usuario. Formalmente, existen diversas definiciones para esta cualidad del software, entre las cuales se pueden citar:

- Es una cualidad del software que se refiere a su capacidad de ser comprendido, aprendido, usado y ser atractivo para el usuario, en condiciones específicas de uso (ISO/IEC 9126)
- Usabilidad es una cualidad del software que determina la efectividad, eficiencia y satisfacción con la que un producto permite alcanzar objetivos específicos a usuarios específicos en un contexto de uso específico (ISO/IEC 9241-11)
- Según Nielsen [3], la usabilidad es un atributo de calidad que indica qué tan fácil de usar es una interfaz de usuario. Un sistema es usable si es funcionalmente correcto

(efectividad), eficiente de usar (eficiencia), fácil de aprender para usuarios novatos, fácil de recordar para usuarios ocasionales, tolerante a errores y subjetivamente agradable (satisfacción)

- "Effective (efectividad), Efficiency (eficiencia), Engaging (ser atractivo), Error-Tolerant (tolerante a errores) y Easy-to-Learn" (fácil de aprender) [4]

Como se observa, los autores citados anteriormente coinciden en atribuirle cualidades de eficiencia, comprensión, satisfacción, comodidad en el uso de los sistemas interactivos. Además, es importante destacar que un producto no es intrínsecamente usable. La facilidad de uso o usabilidad dependerá de un contexto y de usuarios particulares. La usabilidad no puede ser valorada en un producto aislado. Se trata de una cualidad centrada en el concepto de calidad en el uso. En definitiva, se refiere a cómo el usuario realiza tareas específicas en escenarios específicos con efectividad y satisfacción.

Según la ISO 9126, la usabilidad forma parte de los atributos que impactan en la calidad interna y externa de los productos de software, al igual que la funcionalidad, fiabilidad, eficiencia, mantenibilidad y portabilidad.

Los sistemas usables requieren menos entrenamiento, menos soporte para el usuario y menos mantenimiento; además se aprecia mejora en la productividad; se reduce el esfuerzo y permite a los usuarios manejar una variedad más amplia de tareas. Adicionalmente mejora la calidad del producto ya que el diseño centrado en el usuario resulta en productos de mayor calidad de uso, más competitivos en el mercado. Los sistemas difíciles de usar disminuyen la salud, bienestar, productividad y motivación de los usuarios, y pueden incrementar el absentismo.

Alcanzar la usabilidad en el desarrollo de software implica aplicar un Diseño Centrado en el Usuario, a continuación se explica en qué consiste este enfoque.

2.2 Diseño centrado en el usuario

El Diseño Centrado en el Usuario es un conjunto de métodos y técnicas que tiene como paradigma central la inclusión del usuario como centro de desarrollo. A fin de realizar un diseño centrado en el usuario, se debe tomar en cuenta las características del usuario, las actividades que realiza y el escenario donde desempeña su actividad. Todos estos factores permitirán conocer cuáles son los requisitos y metas que se deben satisfacer en el sistema y cómo es la forma más apropiada de ofrecérselos a los usuarios para facilitar el uso del mismo.

Con la finalidad de crear una aplicación que ofrezca una Interfaz Centrada en el Usuario, es necesario considerar las necesidades y demandas de los usuarios en cada etapa del desarrollo de software, lo cual podría incluir:

- Planificación
- Recopilación de datos de los usuarios
- Desarrollo de prototipos
- Aplicación de evaluaciones de usabilidad con usuarios

El primer paso debe consistir en definir claramente la totalidad o parcialidad de las necesidades, metas y objetivos, tanto de los usuarios como de la organización que requiere la aplicación.

En definitiva, el diseño centrado en el usuario propone el diseño de un producto o sistema tomando en cuenta en todo momento las necesidades del usuario. Este enfoque debe aplicarse en cada una de las fases del diseño, haciendo énfasis en las primeras etapas del ciclo de vida, pero sin descuidar el resto de las etapas del desarrollo.

2.3 Evaluación de usabilidad

En general, la evaluación es un aspecto fundamental a considerar en el diseño centrado en el usuario, debido a que permite saber si un sistema cumple las expectativas de los usuarios

y se adapta a su contexto social, físico y organizacional. En particular, la evaluación de usabilidad determina el grado de usabilidad de un sistema interactivo. Su aplicación constante y continua en el desarrollo de software permitirá construir la usabilidad del producto final durante su ciclo de vida. Así, a fin de garantizar la usabilidad de un producto de software, es necesario aplicar técnicas de evaluación de usabilidad desde etapas iniciales del desarrollo.

Existen diversas técnicas de evaluación de usabilidad, que pueden ser aplicadas por distintos actores (usuarios, especialistas, entre otros) y en diferentes momentos del desarrollo del producto, inclusive una vez que ya se ha puesto en producción. La Fundación Sidar-Acceso Universal (SIDAR) [5], en su recopilación de "métodos de usabilidad", traducido por Alejandro Floría, los agrupa en cuatro (4) categorías, a saber: indagación, inspección, prueba y prototipaje. A continuación se describen estas categorías:

2.3.1 **Indagación.** Trata de llegar al conocimiento de algo a través de un conjunto de preguntas e investigaciones; especialmente si se refiere a un asunto oculto o secreto. Las técnicas que se agrupan en esta categoría deben proporcionar información acerca de la usabilidad de un producto que aún no se ha empezado a construir. En estas técnicas se debe hacer un gran trabajo de hablar con los usuarios y observarlos usando el sistema en tiempo real, o analizando respuestas que ellos dan a preguntas verbales y/o escritas. Las evaluaciones enmarcadas en esta categoría permiten identificar los requerimientos del usuario. Son indispensables en una etapa temprana de un proceso de desarrollo que culmina en la satisfacción de una necesidad del usuario, quien con eficiencia y efectividad podrá realizar las funciones le ofrece el producto. Técnicas de este tipo relevantes a este trabajo de investigación son: tormentas de ideas, entrevistas, encuestas, cuestionarios y análisis de sistemas existentes.

Tormentas de ideas: consiste en la generación de ideas por parte de un grupo multidisciplinario, liberando la mente para aceptar cualquier idea que se proponga, permitiendo, así, la libertad para la creatividad. En el caso del proceso de desarrollo de

software, corresponde al equipo de desarrollo. El resultado de una sesión será un conjunto de buenas ideas, y un sentimiento general de haber resuelto el problema. Es recomendable combinar con otras técnicas a fin de lograr los objetivos planteados. Para llevar a cabo una tormenta de ideas exitosa, al inicio se establecen los objetivos de la sesión y los participantes. Se debe establecer un acuerdo acerca de las técnicas de registro o grabación a utilizar (audio, vídeo,...), un horario de la sesión y conducir una prueba piloto para comprobar que tal horario es realista. Durante la sesión, el moderador deberá fomentar la discusión, no permitir la crítica y reunirá los resultados obtenidos al final de cada aspecto tratado. Será importante distinguir entre el consenso del grupo y la opinión de los diferentes participantes. Esta técnica se debería aplicar en las etapas tempranas del proceso de desarrollo, cuando se conoce poco acerca del diseño real y hay necesidad de nuevas ideas, a fin de determinar los requerimientos de usabilidad del sistema a desarrollar

Encuestas: consisten en entrevistas con los usuarios en las que se formula un conjunto de preguntas y se registra las respuestas. Las encuestas se diferencian de los cuestionarios en que son interactivas, aunque no son estructuradas. Se formulan preguntas acerca del producto basadas en el tipo de información que se quiere conocer y se les solicita sus respuestas a los usuarios. Esta técnica puede ser utilizada en cualquier etapa del proceso de desarrollo, dependiendo de las preguntas formuladas en la encuesta y de la información que se requiera por parte de los usuarios.

Entrevistas: es una técnica de indagación común, en la cual un entrevistador le hace preguntas a un experto en algún dominio, a fin adquirir conocimientos en ese dominio. Existen, al menos tres (3) tipos: entrevistas no estructuradas, semi-estructuradas y entrevistas estructuradas. El detalle y la validez de los datos recopilados varían dependiendo del tipo de la entrevista y la experiencia del entrevistador. Las entrevistas son una de las técnicas más aplicadas a fin de determinar los requerimientos de los usuarios. Las entrevistas no estructuradas perecen conversaciones en la que el entrevistado relata su experiencia personal con algún producto o sus expectativas si éste aún no existe. Las

entrevistas semiestructuradas son útiles cuando la gama de respuesta de los entrevistados puede ser muy amplia, lo cual corre el riesgo de alejarse del objetivo original de la entrevista. La entrevista estructurada se recomienda realizar únicamente cuando el rango de respuesta es bien conocido y es necesario decidir desde el punto de vista del usuario cuál de estas respuesta es la más repetida. Los datos obtenidos ofrecen información acerca de las reglas y principios generales y es más rápida que las técnicas de observación. Las entrevistas hacen sentir al entrevistado que su opinión se ha tomado en cuenta en el desarrollo del producto, generando en él un sentido de pertenencia hacia el mismo. Antes de las entrevistas, el equipo evaluador decide qué preguntas le va a hacer a cada usuario e identifica las estrategias a seguir en caso en que se le haga difícil a algún usuario contestar alguna pregunta. Entrevistar a los usuarios respecto de su experiencia con un sistema interactivo resulta una manera directa y estructurada de recoger información. Además las preguntas se pueden dependiendo del contexto. Las entrevistas aportan información muy valiosa sobre aspectos que a veces no son tenidos suficientemente en cuenta por los diseñadores. Las entrevistas son realmente efectivas si el evaluador las ha preparado eficientemente de manera que conduce la misma y se tratan los temas que son realmente necesarios. Las entrevistas son muy bien complementadas por los cuestionarios.

Cuestionarios: es menos flexible que la entrevista, pero puede llegar a un grupo más numeroso y se puede analizar con más rigor. Se puede utilizar varias veces en el proceso de diseño. Deben prepararse muy bien ya que como es un documento a ser completado por los usuarios, debe ser muy claro y exento de ambigüedades que puedan confundirlos. Los cuestionarios son listas escritas de preguntas que se distribuyen entre los usuarios. Requieren un esfuerzo adicional por parte de los usuarios, quienes tendrán que completarlos y regresarlos al equipo evaluador. Se comienza formulando preguntas acerca del producto basadas en el tipo de información que se quiere conocer. Esta técnica puede ser utilizada en cualquier etapa del proceso de desarrollo, dependiendo de las preguntas formuladas en el cuestionario.

Evaluación de sistemas existentes: consiste en la revisión de versiones anteriores del mismo sistema, así como sistemas de la competencia o afines, con el objetivo de identificar problemas de usabilidad. Esto permite, prevenir estos problemas en el sistema que se construye y obtener una medida de base para la usabilidad del mismo. Entre los principales beneficios de esta técnica se tiene que identifica problemas que podrán ser evitados en el diseño del nuevo sistema y provee medidas de efectividad, eficiencia y satisfacción que pueden ser usadas como base para el nuevo sistema. A fin de realizar esta evaluación, se selecciona un conjunto de tareas relevantes y grupos de usuarios que van a ser evaluados.

2.3.2 **Inspección.** Consiste de un tipo de evaluación que puede implicar observación, medición y comparación con respecto a ciertas características. Aplicado a la usabilidad agrupa un conjunto de técnicas para evaluar la usabilidad en los que especialistas, conocidos como evaluadores, verifican el grado de usabilidad de un sistema basándose en la inspección o examen de la interfaz de usuario. El término inspección, en general, consiste de una evaluación que puede implicar observación, medición y comparación con respecto a ciertas características; aplicado a la usabilidad agrupa un conjunto de técnicas para evaluar la usabilidad en los que especialistas conocidos como evaluadores verifican el grado de usabilidad de un sistema basándose en la inspección o examen de la interfaz de usuario. Los métodos de inspección permiten determinar la madurez conceptual del producto que se está construyendo. Dentro de las técnicas de este tipo relevantes a este trabajo de investigación se encuentran: evaluación heurística, listas de comprobación y guías de estilo. Dentro de las técnicas de este tipo relevantes a este trabajo de investigación se encuentran: evaluación heurística, listas de comprobación y guías de estilo.

Evaluación Heurística: es una técnica de inspección de usabilidad en la que los especialistas buscan problemas de usabilidad en la interfaz de usuario. Un grupo inspectores escudriñan la interfaz de usuario y evalúan cada uno de sus elementos ante una lista de principios o heurísticas, comúnmente aceptadas. Nielsen propone una lista de diez (10) heurísticas, resultando suficiente y aceptable para cualquier evaluación de diseño.

Cuanto mayor sea el número de especialistas ante la interfaz, mayor será el número de errores que se podrán encontrar, pero el costo se incrementa. En un estudio, Nielsen concluyó que se podría encontrar la mayoría de los problemas de usabilidad con un número de evaluadores entre tres y cinco. Una vez que se dispone de los especialistas, estos han de proceder a efectuar la evaluación individualmente. Necesitan fijarse en la interfaz ellos solos y que sus compañeros no influyan de ninguna forma en ellos. Asimismo, será preciso asignarles los papeles y los escenarios a utilizar de modo que puedan adquirir la disposición mental y la perspectiva apropiada cuando hayan de interactuar con el producto. El experto revisará la interfaz al menos dos veces, fijándose en cada elemento de la misma (cada menú, control, botón,...) y evaluando su diseño, localización e implementación de acuerdo con la lista de heurísticas. Cuando los expertos llevan a cabo la evaluación, pasan a proporcionar la información obtenida de diversas formas: informe estructurado, una expresión oral de sus hallazgos o categorías de problemas previamente establecidas. Los expertos se reúnen entonces para discutir los hallazgos individuales. En la mayoría de las ocasiones se genera un resumen de los problemas de usabilidad encontrados tanto si los evaluadores discrepan como si un aspecto particular no constituye un auténtico problema. La mayoría de los informes indican la heurística o heurísiticas no respetadas, proporcionando una orientación para su solución. La evaluación heurística puede ser utilizada en, prácticamente, cualquier momento del ciclo de desarrollo. Se puede proporcionar maquetas de papel o incluso especificaciones de diseño a los expertos y detectar una buena cantidad de problemas de usabilidad antes de que el trabajo real de producción de comienzo.

Listas de comprobación: son recomendaciones que el equipo de desarrollo ha acordado considerar en el diseño de la interfaz de usuario. Hay que comenzar decidiendo qué tipo de listas se van a utilizar para juzgar los atributos y los métodos de interacción de la interfaz de usuario. Es posible utilizar las heurísticas de Nielsen o en algunos casos adaptar a los aspectos a los que se enfrenta el usuario del producto en desarrollo. Las listas de

comprobación se pueden utilizar en cualquier etapa del ciclo de vida del software, siempre que se cuente con un prototipo del sistema en desarrollo.

Guías de estilo: Las guías de estilo para el diseño de interfaces de usuario resumen buenas prácticas y provee guías de bajo y alto nivel en el diseño de interfaces usables. La adopción de guías de estilo específicas se puede establecer como parte de los requerimientos no funcionales o requerimientos de usabilidad. El equipo de deserrolo debe estar bien familiarizado con estas guías, y se recomienda aplicar evaluaciones de expertos, similar a la lista de comprobación a fin de verificar el cumplimiento de las mismas. Algunos de los beneficios del uso de estas guías de estilo son: Las guía de estilo incorporan buenas prácticas en el diseño de interfaces de usuario Aplicar esta guía de estilo mejorarán la calidad de la interfaz. Como resultado de aplicar se obtiene una interfaz de usuario mejor diseñada.

2.3.3 **Pruebas**. Las pruebas de usabilidad son una forma de medir qué tan bien puede una persona usar un objeto hecho por el hombre, en particular un software. En las técnicas de evaluación de usabilidad que se agrupan en esta categoría, un conjunto de usuarios representativos trabajan con el sistema –o un prototipo– y los evaluadores toman nota de la interacción, particularmente de los errores y dificultades con las que se encuentren los usuarios. Dentro de las técnicas de este tipo relevantes en esta investigación se pueden citar: protocolo del pensamiento manifestado, protocolo de preguntas y las pruebas de aceptación. En general, una prueba consiste en hacer uso de un producto para ver cómo funciona o qué resultado tiene. Las pruebas de usabilidad son una forma de medir qué tan bien puede una persona usar un objeto hecho por el hombre, en particular un producto de software. En las técnicas de evaluación de usabilidad que se agrupan en esta categoría, un conjunto de usuarios representativos trabajan con el sistema –o un prototipo– y los evaluadores toman nota de la interacción, particularmente de los errores y dificultades con las que se encuentren los usuarios. Dentro de las técnicas de este tipo relevantes en esta

investigación se pueden citar: Protocolo del Pensamiento Manifestado, Protocolo de preguntas y las pruebas de aceptación.

Protocolo del Pensamiento Manifestado: El protocolo del pensamiento manifestado es una técnica de evaluación que consiste en observar a los usuarios mientras están realizando una tarea con el sistema, y a quien se le ha solicitado que exprese en voz alta sus pensamientos, sensaciones y opiniones mientras interactúa con el producto. Se comienza proporcionando al usuario el sistema o un prototipo de su interfaz, así como un escenario de tares a realizar. Se solicita de los usuarios que lleven a cabo las tareas con el producto mientras explican lo que piensan cuando trabajan con su interfaz. El pensamiento en voz alta va a permitir entender la aproximación del usuario a la interfaz y las consideraciones que mantiene en mente mientras hace uso de ella. Algunas de las ventajas de esta técnica son: mejor comprensión del modelo mental del usuario y de su interacción con el sistema, así como de la terminología utilizada por el usuario para expresar una idea o función puede ser susceptible de ser incorporada en el diseño del sistema. Esta técnica se puede utilizar en cualquier fase del proyecto de desarrollo, no es costosa y es una buena fuente de información cualitativa.

Protocolo de preguntas: este método lleva un paso más allá al protocolo del pensamiento manifestado al provocar las manifestaciones del usuario respecto del producto mediante la formulación de preguntas directas acerca del mismo. La capacidad del usuario (o su ausencia) para contestar dichas preguntas sirven de ayuda para detectar qué partes de la interfaz resultan obvias y qué otras resultan oscuras. Se proporciona a los participantes el producto que va a ser sometido a prueba (o bien un prototipo de su interfaz) y un escenario de tareas a realizar. Se solicita de los participantes que realicen las tareas haciendo uso del producto y explicando lo que piensan mientras trabajan con la interfaz del producto. De igual manera, se les formula preguntas puntuales y directas; por ejemplo "¿Cómo enviaría este correo electrónico?". La respuesta, tanto si se basa en el producto que está siendo testeado como si se basa en la experiencia pasada, proporcionará elementos suficientes para

empezar a vislumbrar el modelo mental que ha elaborado el usuario. Esta técnica puede ser utilizada en cualquier fase del proceso de desarrollo.

Pruebas de aceptación: Las pruebas de aceptación conducidas por el cliente verifican que el sistema satisface los requerimientos funcionales y no funcionales. Estas pruebas las realiza el cliente. En general son básicamente pruebas funcionales, sobre el sistema completo, y buscan una cobertura de la especificación de requisitos y del manual del usuario; sin embargo, se pueden utilizar para verificar la satisfacción de los clientes en relación a los requerimientos no funcionales. Estas pruebas se realizan sobre el producto terminado e integrado o pudiera ser una versión del producto pactada previamente con el cliente. La experiencia muestra que aún después del más cuidadoso proceso de pruebas por parte del desarrollador, queda una serie de errores que sólo aparecen cuando el cliente comienza a usarlo. Una prueba de aceptación puede ir desde un informal caso de prueba hasta la ejecución sistemática de una serie de pruebas bien planificadas. De hecho, las pruebas de aceptación pueden tener lugar a lo largo de semanas o meses, descubriendo así errores latentes o escondidos que pueden ir degradando el funcionamiento del sistema. Estas pruebas son muy importantes, ya que definen el paso nuevas fases del proyecto como el despliegue y mantenimiento. Se emplean dos técnicas para las pruebas de aceptación: alfa y beta.

La *prueba alfa* se lleva a cabo, por un cliente, en el lugar de desarrollo. Se usa el software de forma natural con el desarrollador como observador del usuario. Las pruebas alfa se llevan a cabo en un entorno controlado. Para que tengan validez, se debe primero crear un ambiente con las mismas condiciones que se encontrarán en las instalaciones del cliente. Una vez logrado esto, se procede a realizar las pruebas y a documentar los resultados.

La *prueba beta* se lleva a cabo por los usuarios finales del software en los lugares de trabajo de los clientes. A diferencia de la prueba alfa, el desarrollador no está presente normalmente. Así, la prueba beta es una aplicación "en vivo" del software en un entorno que no puede ser controlado por el desarrollador. El cliente registra todos los problemas

(reales o imaginarios) que encuentra durante la prueba beta e informa a intervalos regulares al desarrollador. Como resultado de 1os problemas informados durante la prueba beta, el desarrollador del software lleva a cabo modificaciones y así prepara una versión del producto de software para toda la clase de clientes.

El objetivo principal de realizar este proceso de pruebas de aceptación es lograr un producto con una alta calidad; así las pruebas de aceptación permiten validar que un sistema cumple con el funcionamiento esperado de la manea como el usuario lo espera, y entonces el usuario del sistema determina su aceptación, desde el punto de vista de su funcionalidad, rendimiento y satisfacción. La ejecución y aprobación final de las pruebas de aceptación la corresponden al usuario final.

2.3.4 **Prototipaje.** La aplicación de la técnica de prototipaje consiste en la construcción de un modelo del sistema que se desarrolla. Esta técnica es fundamental en el desarrollo e implementación de los métodos descritos para la inspección y pruebas de un producto, debido a que, habitualmente, no se aplican al producto final, sino un prototipo del mismo con unas determinadas características. Dentro de las técnicas de este tipo relevantes a esta investigación se encuentran: prototipaje en papel, prototipaje rápido y patrones de Interacción.

Prototipaje en papel: esta técnica se caracteriza por el uso de materiales y equipo sencillos para crear una simulación basada en papel de la interfaz de un sistema con el objetivo de explorar los requerimientos de usuario. El resultado obtenido se denomina frecuentemente prototipo de baja fidelidad. Hay que seleccionar a los usuarios apropiados para validar el prototipo y preparar escenarios de tareas realistas para la evaluación. Se podrán tomar notas de determinados problemas, así como de soluciones potenciales durante la sesión para su posterior consideración. Si es preciso, además, se conducirán las entrevistas necesarias con el usuario tras la sesión basándose en preguntas preestablecidas así como otros aspectos que hayan surgido durante la evaluación. Tras el análisis, resumir y evaluación de la información obtenida y las observaciones realizadas se habrá de la

severidad de los problemas identificados, lo que se traducirá en una serie de implicaciones en los requerimientos de usuario. Donde sea necesario, se refinará el prototipo de papel y se repetirá el proceso anterior. Se requieren materiales y recursos mínimos para lograr una sensación de producto. En cuanto a consideraciones temporales, un prototipo de papel y lápiz puede desarrollarse en cualquier momento en cuestión de horas.

Prototipaje rápido: esta técnica está asociada a la idea de desarrollar diferentes conceptos propuestos mediante prototipos de software o hardware, para su posterior evaluación. El desarrollo de la simulación o prototipado del sistema futuro puede ser de gran ayuda, permitiendo a los usuarios visualizar el sistema (su concepto) e informar sobre el mismo pudiéndose utilizar para aclarar opciones sobre los requerimientos de usuario y para especificar detalles de la interfaz de usuario a incluir en el sistema futuro. El prototipaje rápido se describe como una técnica basada en ordenador que pretende reducir el ciclo iterativo de desarrollo. Los prototipos iterativos desarrollados podrán ser rápidamente reemplazados o modificados según los informes procedentes de otras evaluaciones a medida que se evoluciona en el desarrollo de las tareas a realizar. Existen muchas herramientas para la generación de prototipos rápidos, siendo habituales una secuencia de imágenes en Microsoft PowerPoint o Visual Basic, incluso la implementación de los patrones de interacción que se presentan en la sección siguiente. Se debe seleccionar a los usuarios apropiados para la validación del prototipo, intentando cubrir un amplio rango de usuarios dentro de la población objetivo y se prepararán tareas realistas en las que se ocupará a los usuarios mientras trabajen con el prototipo. Se alecciona al usuario para trabajar a través de las tareas seleccionadas, interactuando y respondiendo al sistema de forma apropiada. Si se requiere, se puede obtener información adicional entrevistando a los usuarios inmediatamente después a su interacción con el prototipo. Una vez que se ha analizado, resumido y evaluado la información reunida y las observaciones realizadas se determinará la severidad de los problemas identificados y se resumirán las implicaciones de diseño y las recomendaciones para las mejoras e informar al equipo de diseño. Puede ser preciso refinar el prototipo donde sea necesario y repetir el proceso anterior. Los prototipos

creados por este método presentan una alta fidelidad respecto del producto final, lo que será de interés en el desarrollo de software, pudiendo darse a lo largo de todo el ciclo, por las características que se explican de estos prototipos.

Patrones de Interacción: también conocidos como *patrones de interfaces de usuario o patrones de Interacción Humano-Computador (IHC)* son patrones que describen aspectos concernientes a la interfaz de usuario; están orientados a presentar soluciones apropiadas a problemas recurrentes que se les presentan a los usuarios cuando utilizan las aplicaciones interactivas. Deben estar centrados en los usuarios, esto es, deben proveer soluciones que garanticen interfaces usables. Los patrones de interacción surgen a partir de los modelos de casos de uso y objetos del dominio, los cuales describen las funcionalidades y objetos/operaciones que se manipulan en la aplicación, así como la consideración de los requerimientos no funcionales relacionados a la interfaz de usuario. Para construir un patrón de interacción se debe indicar su nombre, el problema de interacción que resuelve, la solución, y el contexto en el cual esa solución tiene éxito, además los aspectos de usabilidad que se implementan en esa solución, las fuerzas buenas y malas de esa solución que están constituidas por sus fortalezas y debilidades, las consecuencias de usar el patrón que puede consistir en la respuesta por parte del sistema, ejemplos y contraejemplos de su uso y por último se indica si el patrón que se construye está relacionado con algún otro patrón de interacción que describa alguna interfaz que surja a partir de la solución que se describe. Los patrones de interacción asociados a una aplicación deben describir todas las interfaces de usuario provistas por el sistema que se desarrolla, y conforman un lenguaje de patrones para esa aplicación. Existen deferentes tipos de patrones que dependen de lo que se desea describir: usuario, dominio, sistema, tarea, elemento compuesto y elemento simple. Se deben construir en la etapa de análisis de una aplicación y permiten construir un prototipo rápido, al implementar todas las soluciones descritas en un lenguaje de patrones.

Evaluación Heurística: es una técnica de inspección, que tiene como ventaja la facilidad y rapidez con la que se puede llevar a cabo. Normalmente la realiza un grupo reducido de

evaluadores que, con base en su propia experiencia y fundamentándose en principios de usabilidad (heurísticas), evalúan de forma independiente la interfaz de usuario, contrastando finalmente los resultados con el resto de evaluadores. Los evaluadores inspeccionan los sitios web individualmente y sólo después de la evaluación pueden comunicarse sus hallazgos. Se recomienda que los evaluadores naveguen a través de toda la aplicación al menos dos veces para familiarizarse con su estructura y antes de comenzar con la evaluación propiamente dicha. Las sesiones de evaluación duran aproximadamente una o dos horas por interfaz. Los problemas encontrados son jerarquizados con base a su gravedad lo cual facilita su corrección. Las Heurísticas de Nielsen, ampliamente divulgadas son: Visibilidad del estado del sistema, Lenguaje común entre sistema y usuario, Libertad y control por parte del usuario, Consistencia y estándares, Prevención de errores, Es mejor reconocer que recordar , Flexibilidad y eficiencia de uso, Diseño minimalista, Permitir al usuario solucionar el error, y Ayuda y Documentación

Pensamiento Manifestado o en Voz Alta: En esta técnica de evaluación conocida como *thinking aloud* (pensando en voz alta) descrita por Nielsen se les pide a los usuarios que expresen en voz alta sus pensamientos, sentimientos y opiniones mientras que interaccionan con el sistema –o un prototipo del mismo–. Es muy útil en la captura de un amplio rango de actividades cognitivas. Se realiza con usuarios únicos que expresan libremente todo lo que piensan sobre el diseño y la funcionalidad del sistema.

3 Métodos ágiles de desarrollo de software

Un proceso de desarrollo de software consiste en la realización de un conjunto de actividades y resultados asociados, a partir de los cuales se obtiene un producto de software.

Uno de los grupos de metodologías de desarrollo iterativas e incrementales con mayor aceptación a nivel mundial es el conformado por las metodologías "ágiles", basadas en un

manifiesto publicado en 2001 por un grupo de influyentes desarrolladores de la comunidad de la orientación a objetos.

Las metodologías basadas en el "Manifiesto de desarrollo ágil de software" [7] siguen cuatro patrones de valoración preferencial:

Individuos e interacciones *sobre* procesos y herramientas

Software funcional *sobre* documentación exhaustiva

Colaboración con el cliente *sobre* negociación del contrato

Responder al cambio *sobre* seguir un plan

Esto es, a pesar de que dan importancia a los tópicos de la derecha, se da más importancia a los de la izquierda (subrayados).

Más específicamente, el manifiesto ágil es descrito por sus doce principios:

1. La prioridad fundamental es proveer satisfacción al cliente a través de entregas tempranas y continuas de software útil.
2. Los cambios en los requerimientos son bienvenidos, incluso en etapas avanzadas del desarrollo. Los procesos ágiles usan los cambios para proporcionar ventajas competitivas al cliente.
3. Entregar software funcional frecuentemente, desde unas pocas semanas hasta un par de meses, con preferencia hacia las escalas de tiempo más cortas.
4. El cliente y los desarrolladores deben trabajar juntos a diario durante el desarrollo del proyecto.
5. Mantener motivados a los individuos que construyen el proyecto. Darles el ambiente y apoyo que necesitan, y confiar en que lograrán el trabajo.
6. El método más eficiente y efectivo para hacer llegar la información a y dentro de un equipo de desarrollo es la conversación cara a cara.
7. El software funcional es la medida de progreso primordial.

8. Los procesos ágiles promueven el desarrollo sustentable. Los patrocinadores, desarrolladores y usuarios deben ser capaces de mantener un paso constante indefinidamente.
9. La atención continua a la excelencia técnica y al buen diseño realza la agilidad.
10. La simplicidad (el arte de maximizar la cantidad de trabajo que no se hace) es esencial.
11. Las mejores arquitecturas, requerimientos y diseños surgen de los equipos que se organizan solos.
12. A intervalos regulares, el equipo reflexiona sobre cómo volverse más efectivo, entonces afina y ajusta su comportamiento en consonancia.

Las principales características que distinguen a las metodologías ágiles de otras metodologías iterativas e incrementales podrían resumirse de este modo:

- Los tiempos de las metodologías ágiles son medidos en escalas mucho más cortas, semanas en lugar de meses.
- Los equipos de trabajo son altamente colaborativos, auto-organizados y horizontales (sin liderazgo jerárquico), debiendo reunirse todos los días, incluso varias veces de ser posible. Las metodologías ágiles favorecen que el sitio físico de desarrollo sea una oficina sin paredes en la que cada miembro del equipo (que debe ser pequeño, hasta 10 miembros preferiblemente) pueda simplemente caminar de un lado a otro en cuestión de segundos para discutir con otros miembros. Equipos más grandes podrían y deberían ser divididos en sub-equipos con tareas específicas, pero aún colaborando estrechamente con los miembros de otros equipos.
- Las fechas de entrega suelen ser manejadas como "cajas de tiempo" (*timebox*); esto es, si la duración de una iteración es de dos semanas, cualquier meta que se haya trazado para esa iteración debe alcanzarse en ese período; si no se logra alcanzar la meta, no se permite rodar la fecha de entrega; en cambio, se deberá disminuir la lista o complejidad de las tareas que se esperaba lograr en la iteración y ésta será considerada como un

fracaso parcial. Al concluir la iteración, se deberá discutir las causas e implicaciones del fracaso parcial con el objetivo de evitar que algo similar en iteraciones posteriores.

Algunas metodologías ágiles de uso común en la actualidad son: Extreme Programming (XP), Scrum, OpenUP y Agile Unified Process (UP)

4 El Método AgilUs

En la Ingeniería de Software existen diversos tipos de metodologías de desarrollo de software, entre las que se destacan las tradicionales y las ágiles. Las metodologías tradicionales son aquellas con mayor énfasis en la planificación y control del proyecto, en especificación precisa de requerimientos y modelado. Las ágiles se caracterizan por ser iterativas e incrementales, y fácilmente adaptables a los cambios, involucrando al usuario en el desarrollo [8]. En la actualidad, estas metodologías ágiles tienen gran aceptación a nivel mundial y están basadas en un manifiesto publicado en 2001, cuyo objetivo fue esbozar los valores y principios que deberían permitir a los equipos desarrollar software rápidamente y respondiendo a los cambios que puedan surgir a lo largo del proyecto. Se pretendía ofrecer una alternativa a los procesos de desarrollo de software tradicionales.

Las metodologías basadas en el "Manifiesto de desarrollo ágil de software" [7] les dan más importancia a los individuos, funcionalidad, colaboración con el cliente y adaptación al cambio, que a los procesos y herramientas, exceso de documentación, contratos y el seguimiento estricto de un plan. Las metodologías ágiles buscan ser más adaptables a los continuos cambios que se presentan durante el desarrollo de un sistema y para esto emplean un enfoque iterativo e incremental, con interacciones cortas, planificación adaptativa y entrega evolutiva. Se busca lograr que los cambios sean menos costosos, permitiendo que sean incorporados más fácilmente [9].

El Método AgilUs es un método de desarrollo ágil, resultado de una de las líneas de investigación desarrolladas en el Centro de Ingeniería de Software y Sistemas (ISYS) de la

Escuela de Computación, Universidad Central de Venezuela[10]. Se basa en el concepto de usabilidad, en la necesidad de desarrollar software usables. Se fundamenta en el análisis centrado en el usuario y en la participación de especialistas, con el objetivo de evolucionar el software, a fin de que éste alcance el mayor grado de usabilidad una vez culminado su desarrollo. AgilUs es un método de desarrollo iterativo e incremental que pone el mayor peso del desarrollo en la consecución de la usabilidad. Se centra en que la construcción y desarrollo de las interfaces de usuario no debe ser una adición estética que se da al final del desarrollo del sistema sino, muy por el contrario, el desarrollo de interfaces de usuario debe guiar las decisiones en Ingeniería de Software. En AgilUs son los usuarios, y no el cliente ni los programadores quienes guían el desarrollo del proyecto. Algunos trabajos relacionados con esta investigación pero más orientados a métodos de desarrollo tradicionales son la Tesis Doctoral de Granollers [11] y la propuesta de Ferré [12].

El Método AgilUs busca proporcionar un conjunto de actividades organizadas para construir la usabilidad en el diseño de interfaces de usuario durante el desarrollo de un producto de software. El proceso de desarrollo de software engloba las actividades de requisitos, análisis, prototipaje y entrega; así como las evaluaciones de usabilidad correspondientes a cada etapa del proceso. Se realizan en ciclos iterativos hasta alcanzar el producto final. En cada etapa del proceso de desarrollo de software, se incluyen actividades propias para la construcción de la usabilidad.

4.1 Principios

AgilUs centra el desarrollo de software en los siguientes principios:

- Integra la Interacción Humano Computador (IHC) y la Ingeniería de Software (IS). IS y IHC son complementarias, no son disciplinas excluyentes. Un diseño centrado en el usuario impacta positivamente en la calidad del software (ISO 9126-1).
- La usabilidad debe considerarse desde el principio del desarrollo. Si la IS y la IHC son complementarias y no excluyentes, y si la usabilidad aumenta la calidad del software,

entonces es conveniente incluir la usabilidad desde el principio en el desarrollo como uno de los requisitos para impactar positivamente la calidad del producto final.

- La usabilidad determina la utilidad. Un software se considera útil en la medida que pueda ser usado a fin de producir resultados, en forma eficiente, intuitiva y satisfactoria para los usuarios.
- El usuario determina la usabilidad. La usabilidad no es una propiedad abstracta. Un software sólo será considerado usable en un contexto específico y por un tipo de usuario específico. El objetivo es lograr que todos los usuarios del software encuentren usables las tareas que pueden realizar.

4.2 Buenas prácticas

Algunas de las "buenas prácticas" del desarrollo de software que se aplican en AgilUs, las cuales están enfocadas en satisfacer las demandas del usuario y el desarrollo iterativo e incremental, procurando la usabilidad en cada paso del proceso de desarrollo, son las siguientes:

- Diseño centrado en el usuario (DCU). El DCU es un enfoque de diseño y desarrollo que se centra en los deseos, limitaciones y necesidades de los usuarios finales de un software. En las técnicas de DCU es relevante que los desarrolladores realicen pruebas constantes para verificar el curso que lleva el desarrollo del sistema y su interfaz de usuario. De este modo, el usuario guía indirecta pero influyentemente el proceso de desarrollo del sistema. La diferencia fundamental entre éste y otros enfoques de diseño es que en el DCU se procura construir el sistema para adaptarse, a través de su interfaz, a cómo el usuario desea trabajar, en lugar de forzar al usuario a cambiar su modo de trabajar para adaptarse a lo que los desarrolladores consideraron apropiado [13].
- Diseño basado en prototipos. El desarrollo de software en AgilUs está guiado por la construcción de prototipos de alta fidelidad y la evaluación de los mismos por los usuarios y por especialistas en usabilidad. Se entiende entonces que, tras una inspección

inicial, los desarrolladores producen un primer prototipo, los especialistas y el usuario lo evalúan, los analistas preguntan directamente al usuario sus opiniones sobre el desarrollo, y con esa retroalimentación, los desarrolladores se disponen a producir un siguiente prototipo. Este ciclo continúa hasta que se tiene un producto listo para la entrega, cuando las evaluaciones de usabilidad, requerimientos y calidad del software están completamente satisfechas.

- Desarrollo ágil, iterativo e incremental. Una de las máximas del desarrollo iterativo e incremental y del Manifiesto Ágil es la simplicidad. Se recomienda entonces desarrollar el sistema más simple que satisfaga las necesidades actuales de los usuarios, preparándose para cambios futuros. El desarrollo por incrementos permite proveer resultados sin necesidad de esclarecer todo los requisitos de una vez al inicio del desarrollo. La iteratividad permite regresar a etapas anteriores una vez recibida la retroaliemnatción producto de las evaluaciones realizadas.
- Usabilidad como atributo de la calidad. Como se indica en el estándar ISO/IEC 9126-1, la usabilidad es considerada un atributo de la calidad interna y externa del software, y AgilUs hace énfasis en la producción de software usable, siguiendo la recomendación de este estándar internacional.
- Interacción continua con el usuario, propiciando un intercambio cara a cara. Naturalmente, para AgilUs la presencia constante y participativa del usuario es fundamental. El equipo de desarrollo sólo puede tomar decisiones tras realizar evaluaciones de usabilidad, y la usabilidad del sistema sólo puede ser determinada por el usuario.

4.3 Ciclo de vida

El ciclo de vida de AgilUs hace énfasis en la importancia del usuario y sus evaluaciones. Está basado en el desarrollo iterativo e incremental de prototipos de alta fidelidad hasta que se convierten en el producto final para entrega. Este producto *final* puede ser

posteriormente modificado a través de un mantenimiento correctivo y/o evolutivo, que no está contemplado como parte del método.

En cada etapa del desarrollo se incluyen actividades para la construcción de la usabilidad. Se busca proporcionar una manera de proceder organizadamente para construir la usabilidad durante el desarrollo de un producto. El ciclo de vida engloba la definición de requisitos, análisis, prototipaje y entrega.

La figura 3.1 muestra un diagrama de la relación entre cada una de las etapas del ciclo de vida de AgilUs, con las actividades que se realizan y artefactos que se generan en cada etapa. A continuación se describen las etapas de este método:

- Requisitos: Se realiza el análisis global del problema a solucionar, se estudian productos similares existes, se genera un perfil de usuario, y se define la lista de requerimientos a desarrollar. Esta etapa es importante en el desarrollo del software, ya que un mal análisis de requisitos traería como consecuencia un software que no cumple con las necesidades del usuario.
- Análisis: Se lleva a cabo el análisis de la solución a desarrollar, se emplean diagramas de casos de uso y modelo de objetos del dominio, siguiendo la notación UML, para definir las funcionalidades que tendrá el producto a desarrollar.
- Prototipaje: Se implementa un prototipo rápido de la interfaz de usuario a partir de los patrones de interacción, el cual va evolucionando hasta convertirse en el producto final, se genera la guía de estilo, y se realizan evaluaciones de usabilidad apropiadas a esta etapa: las evaluaciones heurísticas y las listas de comprobación.
- Entrega: Se aplican las pruebas al sistema para certificar que la aplicación desarrollada sea un software usable y sin errores, finalmente se pone en producción la aplicación.

Fig. 3.1 El método AgilUs: etapas, actividades y artefactos

4.4 Algunos errores comunes

Varios de los errores que pueden cometerse al implementar AgilUs surgen a causa de una disminución deliberada o accidental de la participación del usuario en el proceso de desarrollo, o de una mala comprensión del rol que debe ocupar a la hora de tomar decisiones con respecto al diseño del sistema y sus interfaces, con base en las evaluaciones de usabilidad:

- El equipo de desarrollo, sin incluir al usuario, puede determinar la usabilidad del sistema. Los desarrolladores pueden llegar a sentir, gracias a la experiencia acumulada o a la poca complejidad del sistema, que están completamente familiarizados con los deseos del usuario y que pueden predecir o aproximar con exactitud sus reacciones, deseos, capacidades y carencias. Este es uno de los errores más frecuentes en el desarrollo de software y, si bien es muy importante que el equipo de desarrollo entienda lo mejor posible al usuario, esto no debe confundirse con la posibilidad de

reemplazarlo. Sólo el usuario puede decidir si el sistema lo satisface; esto es, si el sistema es usable.

- El cliente y el usuario no es necesariamente el mismo. El cliente puede sentir que, ya que es el propietario del sistema, es su deber tomar las decisiones, por ejemplo en términos de usabilidad, dejando a los desarrolladores únicamente los aspectos técnicos. En AgilUs se considera que los usuarios deberían decidir, indirectamente a través de sus evaluaciones y comentarios, qué se debe hacer y por qué.

5 Aplicación de AgilUs en el desarrollo de casos de estudio

En esta sección se describe la experiencia y resultados obtenidos una vez que se ha aplicado el Método AgilUs a fin de validar su efectividad, y seguir evolucionando en la definición del mismo. Los casos de estudio en cuestión corresponden a investigaciones realizadas en el marco del Proyecto titulado "Acercamiento de la tecnología informática al ciudadano", financiado por el Fonacyt; así como trabajos de grado de tercer y cuarto nivel.

5.1 Caso de Estudio Nro 1: GEPECO

El Generador de Periódicos Comunales (GEPECO) es un producto de software cuyo objetivo principal es permitir al usuario, sin conocimientos especializados en diagramación, diseño de periódicos, ni informática, editar una publicación para así obtener un periódico comunal adaptado a las necesidades de su comunidad. GEPECO fue desarrollado como parte de un Trabajo Especial de Grado de las Licenciadas Izarra y Guayaquil [14] y constituye uno de los casos de estudio que han permitido validar el método AgilUs. La aplicación de este método de desarrollo implicó la realización de cuatro etapas, a saber: requisitos, análisis, prototipaje y entrega, con sus respectivas evaluaciones de usabilidad; lo cual se resume a continuación.

5.1.1 Etapa 1: Requisitos

En el análisis de los requisitos se hizo un estudio de aplicaciones afines existentes en el mercado. Además se realizó una tormenta de ideas con el equipo de desarrollo y se aplicó una encuesta a los usuarios potenciales, en la cual se indagó sobre sus opiniones y preferencias en cuanto a características deseables en la aplicación. Los resultados obtenidos en esta etapa se resumen a continuación.

Tormenta de ideas. Esta técnica fue utilizada en el inicio del proceso de desarrollo de una manera un tanto informal. En tal sentido, se realizó una serie de reuniones donde se discutieron diferentes ideas y opiniones referentes a la aplicación. Esto con el fin de llegar a un consenso entre todos los integrantes del equipo de desarrollo, las ideas se concretaron una vez realizadas las siguientes técnicas de evaluación de usabilidad, a saber: análisis de sistemas existentes y una encuesta, las cuales permitieron determinar los requerimientos iniciales del sistema y los perfiles de los usuarios potenciales del mismo.

Análisis de aplicaciones afines existentes. Se realizó un análisis de sistemas de diagramación existentes con la finalidad de puntualizar sus ventajas y aplicarlas al programa a desarrollar; así como minimizar las desventajas o errores que estos pudieran tener. Las aplicaciones estudiadas fueron: Adobe PageMaker 7.0, Scribus, Quakxpress y Publisher 2007.

La figura 4.1 muestra una tabla comparativa de las cuatro aplicaciones analisadas,. De este análisis se tomaron ideas que más correspondan a la aplicación a desarrollar, tales como: minimizar la carga de memoria del usuario reutilizando metáforas conocidas, ofrecer platillas diseñadas, ofrecer un asistente de guía al usuario para crear una publicación, entre otras.

	Adobe PageMaker	**Scribus**	**QuarkXpress**	**Publisher**
Ofrece Plantillas	Si	Si	Si	Si
Provee *toolTips*	Si	Si	Si	Si

Idioma	Inglés	Español	Inglés	Español
Uso Intuitivo	No	No	No	Si
Interfaz sencilla	No	Si	Si	No
Sugerencias al usuario	Si	Si	Si	Si
Software propietario	Si	No	Si	Si

Figura 4.1 Tabla comparativa entre QuarkXpress, Adobe Pagemaker

Análisis de encuestas. A fin de continuar determinando los requisitos funcionales y no funcionales, entre estos últimos se incluyen aspectos de la interfaz de usuario, se aplicó una encuesta, la cual permitió indagar: preferencias en el aspecto de la interfaz de usuario, algunas funcionalidades, experiencias previas, entre otras. De los resultados obtenidos se destacan aquellos relacionados con la forma de presentar la información en la interfaz de usuario, con características del periódico a ser generado y una pregunta que reflejó parte del perfil de los usuarios de la aplicación. A continuación se presenta algunos de los resultados obtenidos con la aplicación de la encuesta. El análisis completo de los resultados se puede consultar en [14]. En la *Pregunta 4*, los resultados se dividieron entre las cuatro (4) opciones, 30% para la primera y última opción y 20% para la segunda y tercera opción, por lo que se incluyeron los cuatro formatos en el sistema de diagramación, estos resultados se aprecian en la figura 4.2. En la *Pregunta 10*, la cual se refería a las expectativas de los encuestados con respecto al un sistema que facilitaría la creación de periódicos comunales, la mayoría de los comentarios estuvieron ligados a que el sistema debía ser de fácil manejo y que le brindara la oportunidad de editar el periódico a su gusto, adicionalmente se observó una gran cantidad de comentarios apoyando la idea del desarrollo de este sistema.

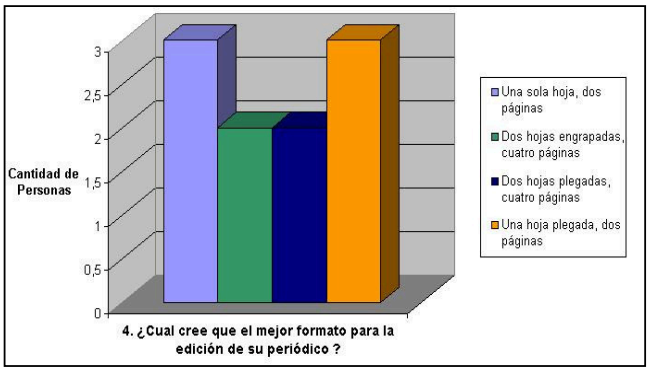

Fig. 4.2. Gráfico de barras correspondiente a la Pregunta 4

Perfil de usuario. Los usuarios potenciales del sistema son los miembros de las comunidades; es decir, personas sin conocimientos avanzados sobre periodismo ni diagramación de periódicos, por lo que no se considerarán usuarios expertos en el dominio del sistema, ni en las tecnologías necesarias para la generación automática de periódicos comunales impresos. Los únicos conocimientos que se requieren por parte de los usuarios es el uso de herramientas ofimáticas.

Lista de requerimientos funcionales y no funcionales. Luego de realizar el análisis de requerimientos a través de la tormenta de ideas, análisis de sistemas existentes y la aplicación de la encuesta, se obtuvo una lista de requerimientos funcionales y no funcionales para el desarrollo del sistema, de los cuales se destacan los siguientes. Entre los requerimientos funcionales: Crear un periódico a todo color, y da la posibilidad de permitir imprimir en blanco y negro para reducir costos; permitir la previsualización del periódico antes de imprimirlo; proveer plantillas de diseño; proveer ayuda en cada paso del asistente (wizard); mostrar mensajes de ayuda estilo *tooltips* en los botones y menús de la aplicación y proveer una funcionalidad que le permita al usuario guardar un periódico, para editarlo posteriormente. Como requerimientos no funcionales: que el sistema fuese sencillo e intuitivo, con un alto grado de usabilidad; fácilmente extensible ya que se trata de un producto de código abierto; mantenible y que pueda ejecutarse en un computador con una configuración básica.

5.1.2 Etapa 2: Análisis

En esta etapa se realizó el análisis de la solución: se definieron los casos de uso, el modelo de objeto del dominio (expresados ambos en UML), se realizaron prototipos en papel y los patrones de interacción que describen la interfaz de usuario.

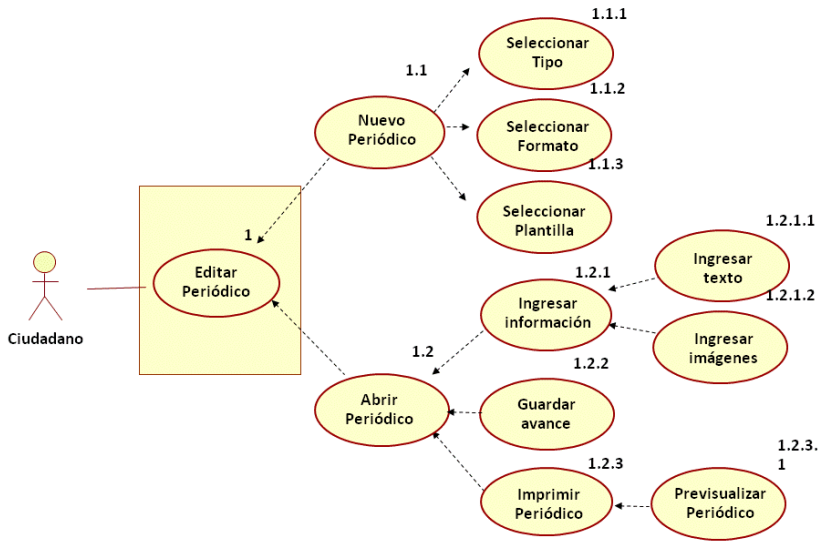

Fig. 4.3: Modelo de Casos de Uso de GEPECO

Modelo Casos de Uso. Un modelo de caso de uso describe la secuencia de las interacciones que se desarrollarán entre los actores y el sistema, en respuesta a un evento que inicia un actor. En esta aplicación se definió un actor principal denominado *Ciudadano*, el cual utilizará la aplicación para la creación de un periódico comunitario. Este actor tendrá acceso a todas las funcionalidades de la aplicación (Ver figura 4.3). El caso de uso *Editar Periódico* permite a un ciudadano iniciar la creación de un periódico desde una plantilla o abrir un periódico existente. El ciudadano tiene la opción de elegir crear un periódico para imprimir o un periódico para colocarlo en la Web, escoger el formato (una o dos hojas), seleccionar la plantilla que va a utilizar e ingresar información al periódico (texto e imágenes).

Modelo Objetos del Dominio. Un modelo objeto del dominio describe los objetos que del dominio de la aplicación y las relaciones estáticas entre estos. En la figura 4.4 se muestra el modelo de objeto del dominio de GEPECO. Se especifican los siguientes objetos: Periódico Comunal (Publicación impresa generada), Plantillas (Diseño y diagramación a seleccionar para un periódico), Páginas (Conjunto de artículos), Artículos (Agrupación de imágenes y texto), Imágenes, Textos. Entre estos objetos se establecen las siguientes relaciones: un periódico comunal se basa en una plantilla, una plantilla posee de una (1) a cuatro (4) páginas, una página tiene asociado uno (1) o más artículo, un artículo tiene una (1) o más imágenes y tiene uno (1) texto.

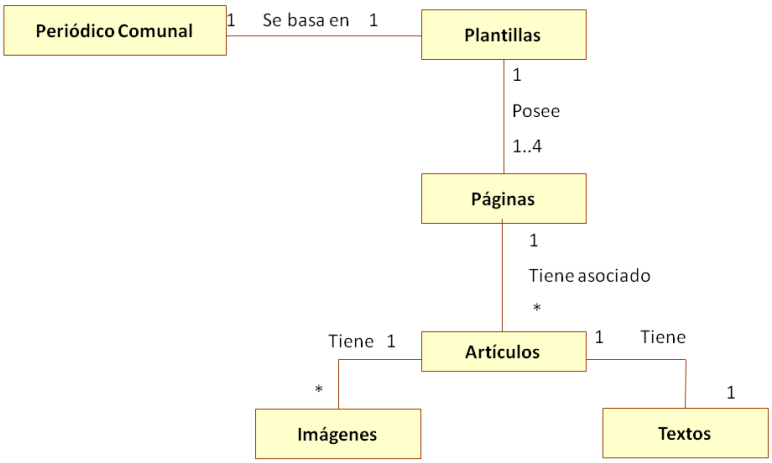

Fig. 4.4: Modelo Objetos del Dominio de GEPECO

Prototipo en papel. Durante el análisis de la aplicación se elaboró de un prototipo en papel de la primera pantalla que se le ofrece al usuario, el cual se muestra en la figura 4.5.

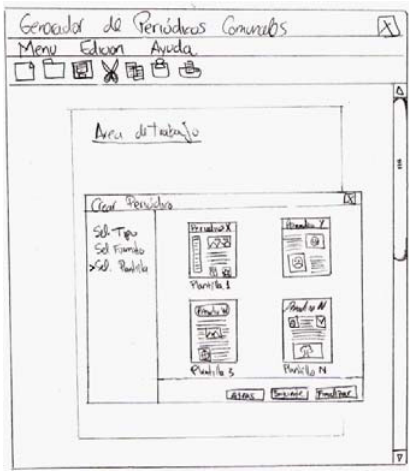

Fig. 4.5: Prototipo papel de Gepeco

Patrones de interacción. Un patrón de interacción describe una solución exitosa a un problema recurrente concerniente a la interfaz de usuario, en un contexto dado [15]. El lenguaje de patrones que describe la interfaz de usuario que ofrece GEPECO se puede consultar en detalle en [14] y se puede visualizar gráficamente en la figura 4.6, la notación utilizada para expresar el lenguaje y cada uno de los patrones es la propuesta en [15].

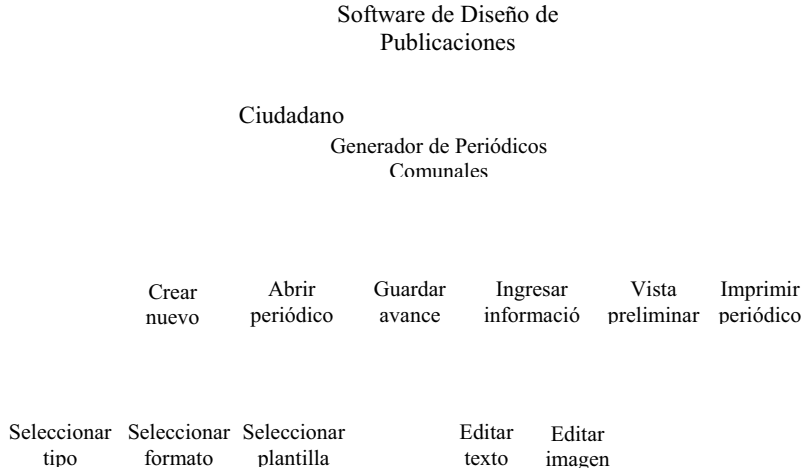

Fig. 4.6: Lenguaje de Patrones de Interacción de Gepeco

A manera ilustrativa, la figura 4.7 muestra el patrón de sistema de la aplicación en el cual se describe el problema general, se propone la solución, se determina el contexto y el mecanismo de activación del mismo.

Nombre	Generador de Periódicos Comunales
Problema	El usuario desea generar un periódico impreso para su comunidad, de forma sencilla y rápida sin tener que tener conocimientos ni en diseño de publicaciones ni en el área informática.
Solución	Desarrollar una aplicación que permita que algún miembro de una comunidad pueda generar un periódico comunitario 
Contexto	Usuarios con acceso al Software Generador de Periódicos Comunales
Usabilidad	Fácil de aprender, Tolerante a errores ySatisfacción del usuario.
Fuerzas	El usuario no necesita tener conocimientos en diseño gráfico ni diagramación de publicaciones; ni necesita tener conocimientos informático avanzados pero sí debe tener conocimientos en el manejo de aplicaciones de ofimáticas.
Patrones Relacionados	Crear nuevo periódico, abrir periódico, guardar avance, ingresar información, vista preliminar, imprimir periódico, seleccionar tipo, seleccionar formato, seleccionar plantilla, editar texto, editar imagen.

Fig. 4.7: Patrón de sistema de GEPECO

5.1.3 Etapa 3: Prototipaje

En esta etapa se ejecutó la implementación del sistema a partir de los patrones de interacción, se generó la guía de estilo, y se realizó la evaluación heurística de GEPECO.

Guía de estilo. Se define como guía de estilo al documento que recoge normativas y patrones básicos relacionados con el aspecto de un interfaz para su aplicación en el desarrollo de pantallas dentro de un entorno concreto. La guía de estilo de GEPECO describe los colores primarios, el aspecto de los botones, las operaciones por el teclado, entre otras. A continuación se muestra algnos aspectos de la guía de estilo de la aplicación Generador de Periódicos Comunales (GEPECO).

Los colores primarios de la aplicación se muestran en la figura 4.8.

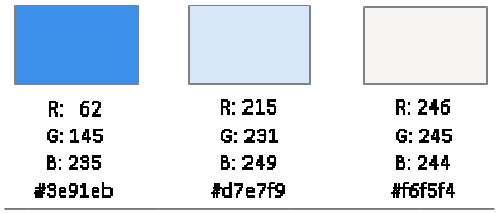

Fig 4.8: Colores primarios

Las diversas apariencias de los botones se muestran en las figuras Fa y Fb. Al pasar el cursor sobre un botón éste debe cambiar su color, de gris a azul claro (figura 4.9a). El color de la fuente de los botones inactivos debe ser gris, para diferenciarlo de los activos, cuyo color de fuente es negro (figura 4.9b).

Fig. 4.9a Cambio de color al hacer mouse over

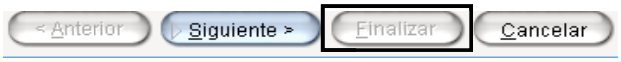

Fig. 4.9b botones activos e inactivos

La combinación ALT+tecla activa la función a la que está asociada, esto se conoce como mnemotécnicos. Se elige para la tecla, la letra la inicial o en su defecto una consonante prominente, la cual aparece subrayada en el nombre de la función (ver figura 4.10)

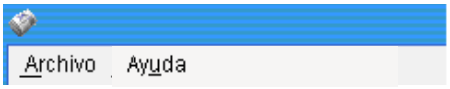

Fig. 4.10: Mnemotécnicos o teclas de acceso

Al situar el ratón sobre algún elemento gráfico, se mostrará una ayuda adicional acerca del elemento. Esta ayuda se presenta en forma de *ToolTips* o mensajes emergentes. El *tooltip* debe ser un recuadro amarillo con letras negras, esto se muestra en la figura 4.11

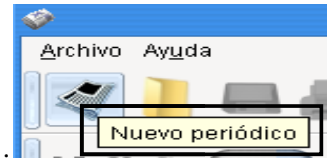

Fig 4.11: *ToolTips* o mensajes emergentes

Evaluación Heurística. La evaluación heurística es un método de inspección cuyo objetivo es encontrar problemas de usabilidad en el diseño de una interfaz de usuario, tal que puedan ser atendidos como parte de un proceso de diseño iterativo. Se consideraron las diez (10) heurísticas de Nielsen propuestas en [16]. La escala usada para la valoración de los problemas va de 0 (menos importante) a 4 (catastrófico). La Evaluación Heurística fue realizada por los alumnos de la asignatura Interacción Humano Computador, dictada en el 5to semestre de la Licenciatura en Computación de la Universidad Central de Venezuela. A cada uno de los participantes se le entregó una planilla, y se procedió a realizar dos recorridos de la aplicación. En la figura 4.12 se exponen los problemas más resaltantes encontrados durante la evaluación y que fueron corregidos por el equipo de desarrollo de GEPECO.

Problema	Heurística	Valoración	Solución
La metáfora de abrir tiene asociado un color muy parecido a una opción deshabilitada.	H3 y H4	1	Colocarle colores apropiados a la metáfora.
No previene errores al cerrar el programa, tal como guardar documento o guardar un periódico antes de cambiar a otro.	H9 y H10	4	Incluir mensajes de alerta para el usuario en casos donde sea necesario.
No muestra feedback al usuario cuando se está realizando alguna acción	H5	3	Implementar mensajes que le indiquen al usuario que su acción está en proceso.
No se puede cambiar la posición del texto, como centrar, justificar, etc.	H1 y H10	2	Agregar las opciones correspondientes.
En los tips de ayuda cuando se crea un periódico se pueden modificar	H4	0	Bloquear los campos de los tips de ayuda
La opción salir no funciona.	H6 y H10	2	Deben realizar la opción salir.

Fig. 4.12: Resultados de la evaluación heurística de GEPECO

5.1.4 Etapa 4: Entrega

En esta etapa se llegó a una versión del sistema desarrollado que se consideró listo para su liberación. Así, se aplicaron las pruebas de aceptación al sistema para certificar que la aplicación desarrollada cumplía con los requisitos de usabilidad exigidos por los usuarios. Finalmente se entrega a producción la aplicación. El ambiente de producción consta de una máquina tipo PC o MAC, en la cual se puede estar ejecutando cualquier sistema operativo

que admita correr la máquina virtual de Java. A continuación se presentan los resultados más importantes de las pruebas de aceptación.

Pruebas de aceptación. Las pruebas de aceptación permiten conocer la opinión general de los usuarios. Se utilizan para determinar la satisfacción subjetiva del usuario. Para los fines de la aplicación desarrollada, se aplicaron cuestionarios a fin de conocer el nivel de aceptación de los usuarios. Fue entregado a una muestra de 20 personas. A manera ilustrativa se presenta el resultado de la *Pregunta 6*. Se refleja en la figura 4.13 que para el 80% de los usuarios el periódico cumplió con sus expectativas, mientras que el 20% opina que sólo cumplió más o menos. Partiendo de los resultados obtenidos, se puede hacer un promedio de aceptación con base en las preguntas que califican de manera positiva y las que califican de manera negativa la aplicación. Al hacer este cálculo aproximado es posible evidenciar que un porcentaje mayor a la mitad de los usuarios tuvo opiniones positivas acerca de aspectos de usabilidad de GEPECO. Es importante hacer notar que las preguntas del cuestionario buscaban indagar opiniones acerca de aspectos de usabilidad. Es así como al obtener un mayor número de respuestas favorables, se puede decir que esta aplicación es usable.

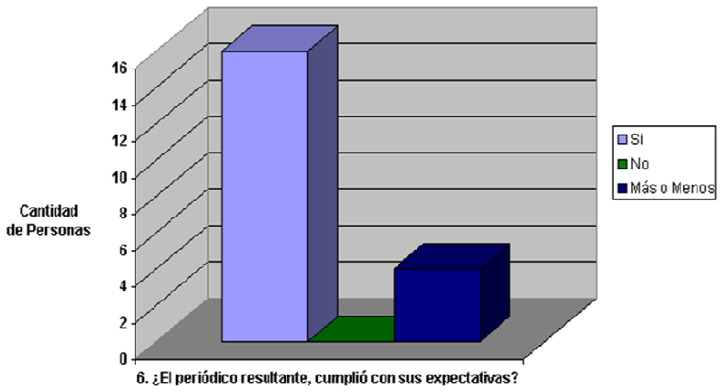

Fig. 4.13: Gráfico correspondiente a la pregunta 6

5.2 Caso de Estudio Nro 2: IVIChem

En esta sección se describe el desarrollo de un Sistema Integrador para Química Computacional denominado IVIChem [17], en el cual se utilizó el método AgilUs. Esta apliación está dirigida, fundamentalmente a los investigadores del laboratorio de Química Computacional (LabQC) del Instituto de Investigaciones Científicas y tecnológicas (IVIC), aún cuando puede ser utilizada desde cualquier parte a través de la Web. La Química Computacional (QC) es una disciplina relativamente reciente de la Química que contempla "el modelaje de todos los aspectos de la química por medio de la computación en lugar del experimento" (Schleyer, 1998). Los programas para cálculos de QC, por lo general, son manejados a través de la línea de comandos del sistema operativo, reciben archivos de texto como datos de entrada, y producen igualmente archivos de texto como datos de salida. La Química Computacional puede resultar una disciplina complicada, no únicamente a causa de su avanzado trasfondo matemático y teórico, sino también debido a la diversidad de aplicaciones que existen y no interactúan directamente, funcionan sobre sistemas operativos distintos, poseen diversas restricciones de licencias, entre otras razones. El Sistema Integrador que se describe a continuación asiste al usuario a lo largo de toda la línea de trabajo, permitiéndole completar las tareas propias de la Química Computacional desde cualquier navegador web y sin requerir gran entrenamiento, a través de una interfaz de usuario usable y accesible. A continuación se describen los resultados obtenidos en cada una de las etapas de la aplicación del método AgilUS.

5.2.1 Etapa 1: Requisitos

En esta etapa se realizaron diversas tormentas de ideas y análisis de sistemas existentes, que permitieron esclarecer las necesidades de los usuarios y, a partir de éstas, los requerimientos del sistema de software desarrollado.

Tormentas de ideas Durante el desarrollo de IVIChem se realizaron varias tormentas de ideas. La figura 4.14 muestra el resultado de una estas tormentas, en la cual se identificaron

los estados posibles de los trabajos en la cola de cálculo, y cómo se expresarían esos trabajos al usuario.

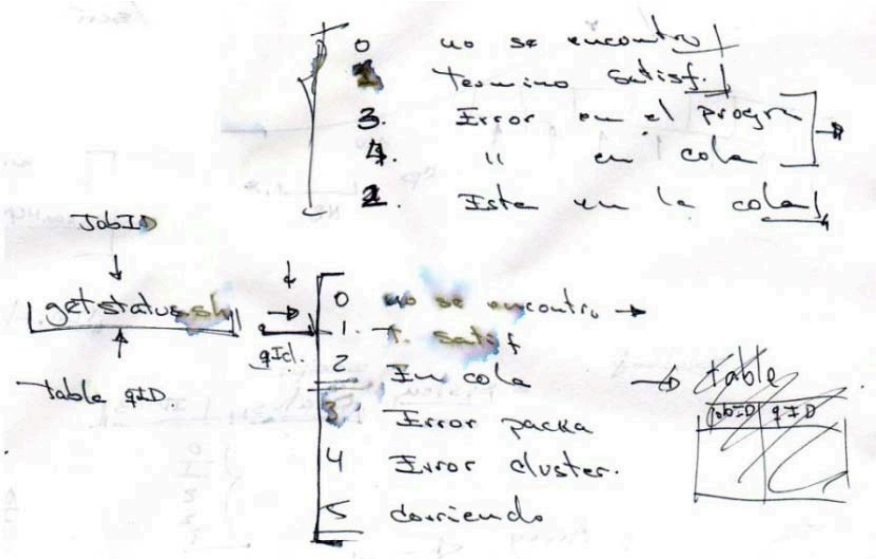

Fig. 4.14: Notas resultado de una tormenta de ideas

Análisis de sistemas existentes En este punto se realizó una revisión de varias aplicaciones y *suites* de programas relacionados con la aplicación a desarrollar. Los resultados obtenidos en esta revisión ayudaron a establecer los perfiles de los usuarios; así como los requerimientos funcionales y no funcionales del sistema. La figura 4.15, a continuación, resume las características de varias aplicaciones evaluadas.

Ninguna de las aplicaciones integradoras evaluadas satisfizo todas las necesidades de los usuarios del LabQC del IVIC pero, en conjunto, casi todos los elementos deseados estaban presentes en alguna u otra de las diversas aplicaciones. Todas las aplicaciones mencionadas fueron usadas como referencia en mayor o menor medida. La utilidad cualitativa de cada aplicación durante la construcción de IVIChem es señalada en la última columna de la derecha en la Fig 4.15.

Aplicación	Disponibilidad del código	Tipo de Licencia	Interfaz de usuario		Plataforma					Utilidad
			línea de comando	GUI						
Aplicaciones para cálculos de QC										
Mopac			✓		✓	✓	✓			★★★★★
CATIVIC	LabQC del IVIC	Pertenece al LabQC	✓				✓			★★★★☆
Gaussian		$	✓	✓	✓	✓	✓			★★★★☆
Hyperchem		$	✓	✓	✓	✓	✓			★★★★☆
Ambientes Integradores										
WebCativic	LabQC del IVIC	Pertenece al LabQC		✓	✓	✓	✓	✓		★★★☆☆
GaussView		$		✓	✓	✓	✓			★★★★★
GabEdit		GNU		✓	✓	✓	✓			★★★★☆
Hyperchem		$	✓	✓	✓	✓	✓			★★★★☆
VEGA ZZ			✓	✓	✓	✓				★★★★☆
WebMO Pro		$		✓				✓	✓	★★★★☆
ChemBio 3D		$		✓	✓	✓				★★★☆☆
Avisto				✓	✓	✓				★★★☆☆
MolWorks				✓	✓	✓	✓	✓		★★☆☆☆
Winmostar				✓	✓					★★☆☆☆
Visualizadores										
Jmol		GNU y LGPL		✓				✓	✓	★★★★★
Jamberoo		Apache2, GPL y LGPL		✓	✓	✓	✓	✓		★★★★☆
Molden				✓	✓	✓	✓			★★★☆☆
Chime Pro				✓	✓					★★☆☆☆
Facio				✓	✓	✓	✓			★★☆☆☆
Otros										
MOP2009				✓	✓					★★☆☆☆
Mol. formats converter		gratuita para uso en web		✓					✓	★★☆☆☆

Leyenda: MS Windows, Apple MacOS, Linux/Unix/Irix, Java, Web
no disponible, abierto, $ comercial, gratuita para usos académicos

Fig. 4.15: Resumen de las aplicaciones para QC evaluadas

A continuación se descrben los requerimientos funcionales y no funcionales identificados inicialmente en esta etapa.

Requerimientos Funcionales: Un *trabajo* (*job*) es la unidad fundamental de IVIChem; el usuario realiza todas sus acciones principales a través de trabajos. Todo trabajo posee dos informaciones de entrada principales: los datos moleculares y las opciones del cálculo. Asimismo, una vez creado, cada trabajo se encuentra en uno de varios estados posibles. La figura 4.16 presenta un resumen descriptivo de estos requerimientos.

Requerimientos Funcionales
Funciones de gestión de trabajos: Crear trabajo a partir de un constructor molecular, Crear trabajo a partir de coordenadas previas, Especificar opciones del cálculo, Detener cálculo, Editar trabajo, Crear carpeta, Editar carpeta, Ver estado de trabajos previos
Funciones para análisis de resultados: Analizar resultados de un trabajo, Analizar resultados de varios trabajos simultáneamente
Funciones de administración del sistema: Añadir usuario, Editar usuario, Añadir institución, Editar institución, Detener cálculo, Añadir opción de análisis, Editar opciones de análisis , Añadir paquete de cálculo, Ver reportes de errores y sugerencias
Funciones de ayuda: Ver manual del usuario, Hacer sugerencia, Reportar error
Requerimientos No Funcionales
Cambio de idioma, Usabilidad, Accesibilidad

Fig 4.16: requerimientos funcionales y no funcionales de IVIChem

Perfiles de los Usuarios: Los resultados del análisis de sistemas existentes, así como las tormentas de ideas, facilitaron el estudio y determinación de diversos tipos potenciales de usuarios: *Investigador (usuario general):* Tendrá disponibles todas las funciones de índole científica del sistema y *Administrado:* Además de poder realizar cálculos de pruebas y cualquier otra función disponible para el Investigador, podrá realizar las funciones administrativas.

5.2.2 Etapa II: Análisis del Sistema

Con base en la información recabada durante la Etapa I de cada iteración de AgilUs, se procedió a ejecutar las actividades de la Etapa II correspondiente, a saber elaboración del

modelo de casos de uso y del modelo objetos del dominio, prototipos en papel y el lenguaje de patrones que describe la interfaz de usuario que ofrece IVIChem. Los resultados finales obtenidos durante la ejecución de esta etapa se describen en esta sección.

Modelo de Casos de Uso: A manera ilustrativa, se presentan en la Figura 4.17 el diagrama de casos de uso correspondiente a las funcionalidades que puede realizar el usuario Investigador, haciendo uso de la Notación UML.

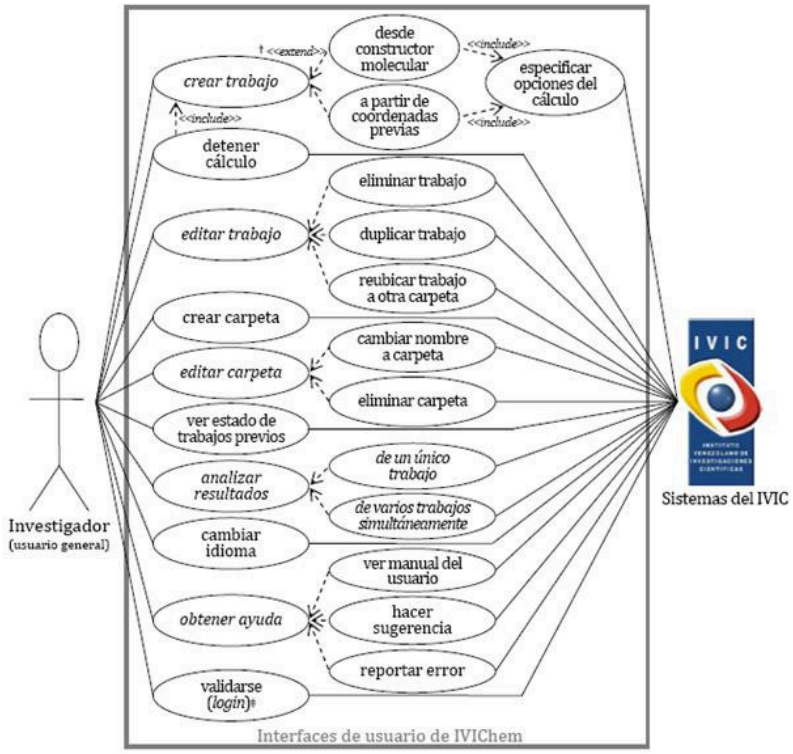

Fig 4.17: Modelo de Casos de Uso

Modelo de Objetos del Dominio: El Modelo objetos del dominio presentado en la Figura 4.18 presenta los objetos más importantes del sistema desarrollado y sus relaciones estáticas.

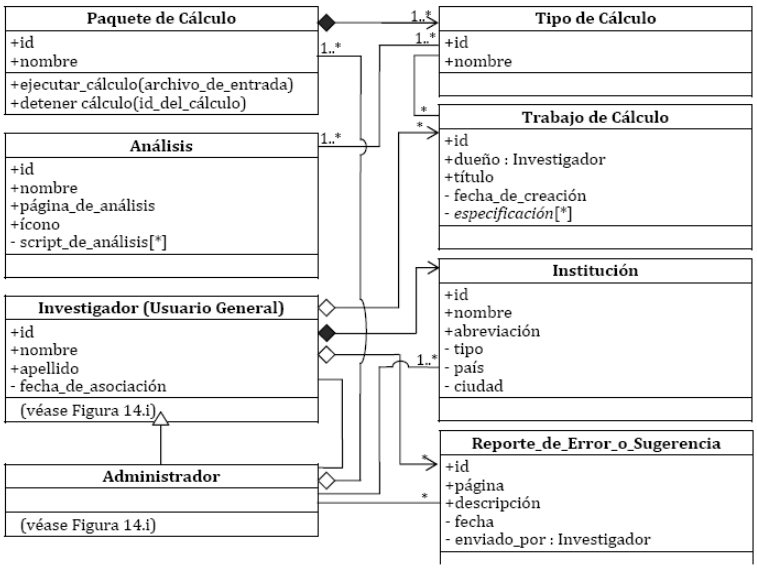

Fig 4.18: Modelo Objetos del Dominio

Patrones de Interacción: La interfaz de usuario de IVIChem se describió a través de la creación de un lenguaje de patrones, el cual se presenta en la Figura 4.19.

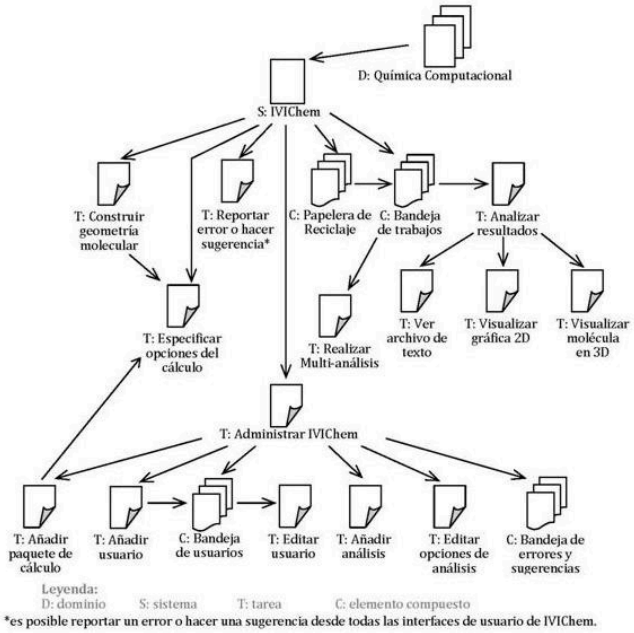

Fig 4.19: Lenguaje de Patrones de Interacción

Uno de los patrones de interacción que describe la interfaz de IVIChem correspondiente a la manipulación de los trabajos se presenta en la figura 4.20.

BANDEJA DE TRABAJOS

Problema

El usuario necesita saber el estado en que se encuentran sus trabajos, requiere conocer alguna información esencial de varios trabajos a la vez, y desea a la vez tener algún modo de acceder a la información completa de cada trabajo para poder analizar los resultados o revisar los detalles de un cálculo previo.

Solución

Ofrecer al usuario una interfaz similar a la de una bandeja de correo electrónico, donde es posible observar al mismo tiempo los datos principales de cada trabajo, conocer el estado en que se encuentra cada uno, y acceder rápidamente a cualquier trabajo deseado para ver detalles del cálculo y analizar resultados o errores.

jobID	Title	Package	Created on	Status
0000028	coroneno 3	mopac	2010-09-16 11:37:38	in queue
0000018	benceno	mopac	2010-09-15 11:41:05	calculation error
0000017	benceno	mopac	2010-09-15 11:40:41	calculating
0000015	benceno	mopac	2010-09-15 09:26:43	complete
0000014	benceno	mopac	2010-09-15 08:53:52	not found
0000013	benceno	mopac	2010-09-15 08:50:57	cluster error

Figura 14.vi - La bandeja de trabajos con todos los estados (status) posibles

Contexto

El usuario ya ha creado al menos un trabajo de cálculo con anterioridad.

Consecuencias

- El usuario puede acceder rápida y eficientemente a todos sus trabajos de cálculo previos, ya sea para analizarlos, para usarlos como referencia, o simplemente para consultar alguna información de interés.
- El usuario no se ve forzado a entrar por la consola de comandos hasta el *cluster* de cálculo y realizar consultas avanzadas a la cola de manejo de trabajos, lo cual requiere familiaridad con el lenguaje de *Shell Scripting* de Linux, además de accesos de prioridad al *cluster*.
- Cualquier usuario registrado en IVIChem puede conocer inmediatamente el estado de sus trabajos sin requerir permisos avanzados ni manejo directo de la cola de cálculo a través de la consola de comandos.

Usabilidad

- Control del usuario y libertad
- Consistencia y estándares
- Ver y reconocer en lugar de recordar
- Ayudar al usuario a diagnosticar y recuperarse de errores

Cómo usarlo

- Se construye una aplicación tipo bandeja de correo electrónico por considerársele familiar para cualquier usuario típico del sistema, y donde cada trabajo previo corresponde análogamente a un correo electrónico recibido o enviado con anterioridad.
- Presentar la información más relevante de un trabajo resumida en una única línea horizontal.
- Usar colores diferentes para estados diferentes, con preferencia hacia los colores fríos (verdes y azules) cuando el estado es positivo, y colores cálidos (naranjas y rojos) cuando el estado es erróneo.

Patrones relacionados

- IVIChem
- Papelera de reciclaje
- Realizar Multi-análisis
- Analizar resultados
- Reportar error o hacer sugerencia

Fig 4.20: El Patrón de Interacción Bandeja de Trabajos

5.2.3 Etapa III: Prototipaje

Las actividades y artefactos de esta etapa consisten en la construcción iterativa e incremental de prototipos ejecutables de la aplicación a partir de los patrones de interacción; así como sus posteriores evaluaciones por medio de diversas pruebas de usabilidad. En esta fase se realizó una evaluación basada en una lista de comprobación, para lo cual se utilizó las 10 heurísticas de Nielsen [16]. A continuación se describen algunos ejemplos que muestran cómo se incorporaron en la interfaz de usuario las recomendaciones planteadas en cada una de las heurísticas

1. *Visibilidad del estado del sistema:* Es importante para el usuario saber en todo momento qué está haciendo el sistema. Un ejemplo se presenta en la Figura 4.21. La aplicación JChemPaint toma tiempo para cargar en el navegador, en particular si la conexión es lenta. Se incluyó una imagen animada junto con un mensaje que permiten al usuario saber que el constructor está cargando. La figura central (el logotipo del IVIC) da vueltas mientras carga JChemPaint.

Fig 4.21: Aplicación JChemPaint iniciando

2. *Coincidencias entre el sistema y el mundo real:* En el ejemplo de la Figura 4.22, correspondiente a la opción para editar opciones de análisis en el módulo de

Administración, se usan matraces para hacer referencia al análisis, considerando que el usuario asociará fácilmente ese instrumento con un análisis.

Fig 4.22: Coincidencia del sistema con el mundo real y con el virtual

La imagen muestra también el aprovechamiento de un elemento virtual muy típico: el lápiz de edición. En este caso se usó una mezcla conveniente de referencias al mundo real y al virtual, minimizando así la carga cognitiva del usuario.

3. *Control del usuario y libertad:* El usuario puede iniciar un trabajo de cálculo a partir del constructor molecular, pero también dispone también de una segunda forma de iniciar un trabajo nuevo, a partir de coordenadas moleculares obtenidas previamente. La Figura 4.23 muestra la selección de esta segunda opción para Mopac.

Fig 4.22: Otra opción para crear un trabajo nuevo

Al ejecutar esta opción el usuario es llevado directamente al módulo de cálculo (en lugar de al constructor molecular), donde puede pegar el texto de su geometría en el campo del archivo de texto, como se indica en la Figura 4.23.

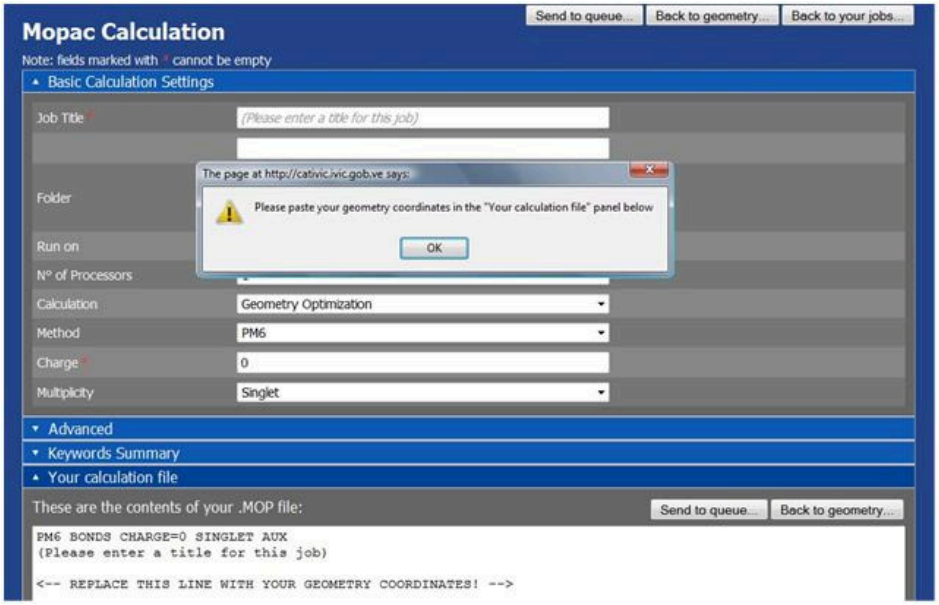

Fig 4.23: Pegar coordenadas en lugar de construir molécula

Se permite de este modo a usuarios avanzados introducir directamente una geometría que hayan obtenido desde algún otro programa de modelado o de cálculo. Con el fin de prevenir errores y minimizar la carga cognitiva del usuario, el sistema muestra una advertencia recordándole que debe pegar el texto de sus coordenadas en el lugar correspondiente.

4. *Consistencia y estándares*: Un ejemplo, el header o cabezal (Figura 4.24a) y el footer o pie (Figura 4.24b) son idénticos para todas las páginas de IVIChem.

Fig.4.24a: Cabezal(*header*) de IVIChem

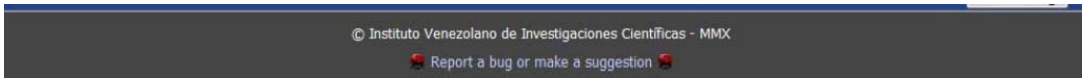

Fig. 4.24b: Pie (*footer*) de IVIChem

5. *Prevención de errores:* Los ejemplos más importantes de prevención de errores en IVIChem contemplan el uso de listas seleccionables, estrechamente relacionados con la heurística "Ver y reconocer en lugar de recordar". Otra herramienta esencial en la

prevención de errores es provista por los tooltips, ampliamente usados en IVIChem. Todos los formularios del sistema contienen tooltips, por ejemplo en el presentado en la Figura 4.25 a continuación.

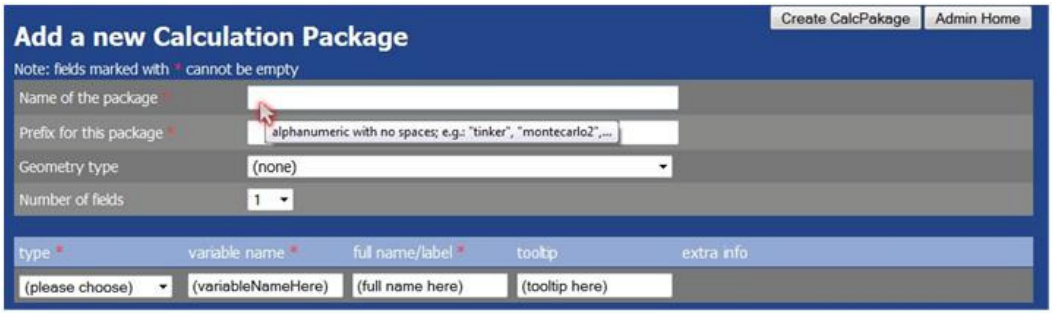

Fig. 4.25: *Tooltips*, presentes en todos los formularios de IVIChem

En la Figura 4.25 el *tooltip* informa al Administrador que debe introducir un nombre alfanumérico sin espacios para el nuevo paquete de cálculo, previniendo así la introducción de nombres inválidos.

6. *Ver y reconocer en lugar de recordar:* El mayor efecto positivo sobre esta heurística se obtuvo de las listas seleccionables. El uso de listas desplegables no sólo previene errores de escritura, también permite al usuario reconocer las opciones disponibles en lugar de verse obligado a recordarlas todas. Por ejemplo, en la Figura 4.26 se muestran todas las opciones de cálculo para Mopac.

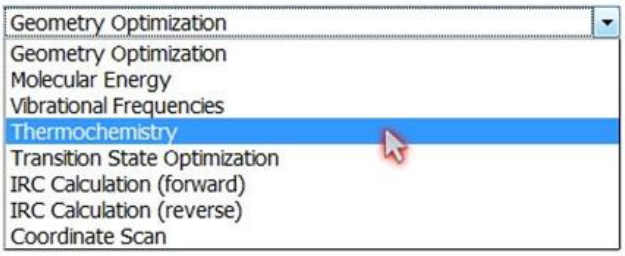

Fig. 4.26: Opciones de cálculo en Mopac

La lista desplegable permite al usuario reconocer todos los tipos de cálculos disponibles, en lugar de forzarlo a recordar lo que se puede hacer.

7. *Flexibilidad y eficiencia de uso:* La eficiencia en el uso fue una de las motivaciones principales y pruebas de usabilidad realizadas durante el desarrollo de IVIChem permitieron ofrecer interfaces de usuario para acelerar todos los procesos involucrados en la ejecución y análisis de resultados de un trabajo. Uno de los ejemplos que causan un importante aumento tanto en la flexibilidad como en la eficiencia del uso de IVIChem es la conformada por la combinación de adición de un nuevo tipo de análisis con el tablero de interruptores virtuales para habilitar o inhabilitar tipos de análisis para cada tipo de cálculo. Esta solución combinada resulta sumamente intuitiva y fácil de usar, y acelera la administración de las opciones de análisis. Con respecto a las funciones de Investigación, cabe mencionar la creación de trabajos como copia de trabajos previos al presionar el botón que se muestra en la Figura 4.27

Fig. 4.27: Crear un trabajo como copia del trabajo actual

Esta acción, que permite crear un trabajo nuevo usando como base una copia idéntica del trabajo actual, aumenta la eficiencia para el usuario al acelerar la creación de trabajos similares con diferencias pequeñas, algo muy común en una investigación típica en Química Computacional.

8. *Diseño minimalista y estético*: Algunos ejemplos de resultados logrados al consultar la opinión de los usuarios sobre la apariencia de las interfaces pueden mencionarse. El uso de un número limitado pero relevante de colores permite alcanzar el balance entre estético y minimalista. Sin embargo, este balance siempre debe ser consultado con los usuarios. Así, se decidió que era conveniente usar colores para indicar estados diferentes, como se muestra en la Figura 4.28, a continuación, sacrificando el minimalismo a favor de mayor información.

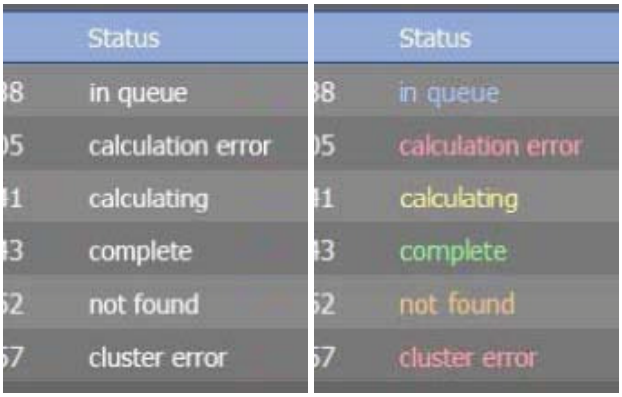

Fig. 4.28: Colores en los estados posibles de un trabajo

9. *Ayudar al usuario a diagnosticar y recuperarse de errores:* En la Figura 4.29 se muestra el uso de una advertencia emergente tras un error por parte del usuario. Si el usuario olvida introducir un título para el trabajo y presiona "Send to queue...", IVIChem detiene la acción, envía una advertencia, y colorea en rojo el campo donde debe ser insertado el título.

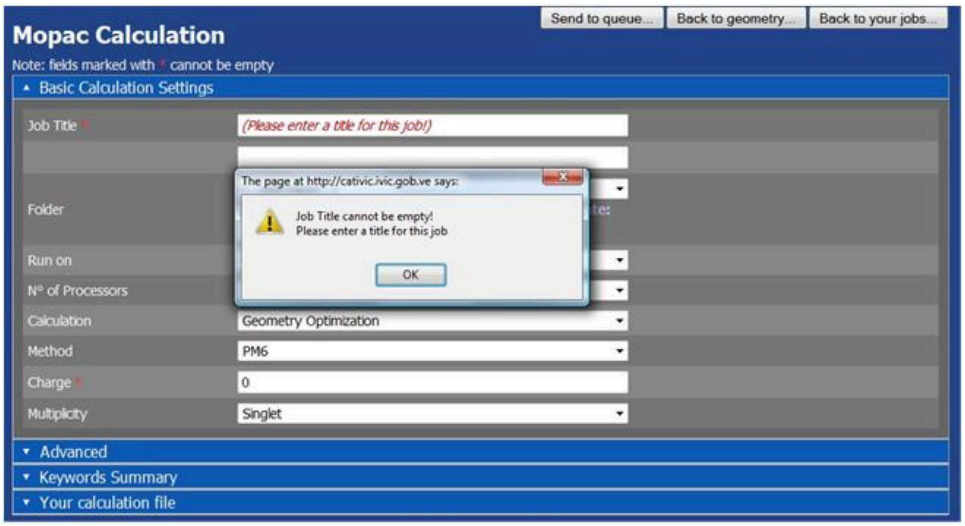

Fig. 4.29: El usuario debe introducir un título para poder continuar

Sin embargo, en varios casos se consideró apropiado y ventajoso incluir soluciones más sutiles e inmediatas, que se adelantasen al envío del formulario. Las advertencias de la

Figura 4.30 se muestran instantáneamente en cuanto el usuario comete un error; esto es, sin esperar a que se ejecute la acción para enviar el formulario.

Fig. 4.30: Advertencias de error instantáneas

En el ejemplo, todos los campos de error han sido activados deliberadamente. Para prevenir errores más graves, la acción del botón "*Create User*" se bloquea y no se ejecutará mientras al menos uno de los campos se encuentre en estado de error.

10. *Ayuda y documentación:* Siguiendo los lineamientos de AgilUs y en general de las buenas prácticas de desarrollo de software usable, se considera que la propia interfaz de usuario debe ser transparente y predecible para el usuario hasta el punto de que los manuales de usuarios, a pesar de estar presentes, no necesitarían ser usados. En la práctica siempre es adecuado incluir un manual de usuarios, estos manuales deben ser tan cortos y concisos como sea posible. Con estas características en mente se diseñó el manual de usuarios de IVIChem (ver mecanismo de activación en la figura 4.31), el cual puede ser consultado directamente desde la aplicación.

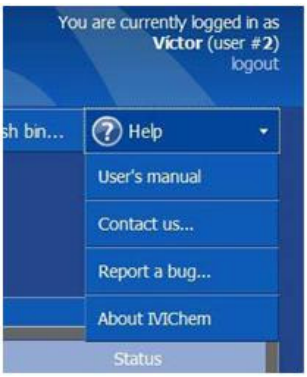

Fig. 4.31: Mecanismo de activación de la ayuda en línea

5.2.4 Etapa IV: Entrega

En esta etapa se logra alcanzar una versión del sistema que se considera suficiente para ser puesta en producción. A fin de evaluar la aceptación por parte de los usuario se realizaron diversas pruebas, las cuales se soportaron en un módulo desarrollado adicionalmente, incorparado a partir de algunas pruebas de usabilidad aplicadas al usuario tipo Administrador. Este módulo permite reportar errores y sugerencias, y aceleró la eficiencia en el reportaje y resolución de las tareas por hacer, errores por corregir y sugerencias por implementar. En la Figura 4.32 se puede observar el formulario para reportar un error o hacer sugerencias. Este formulario es pre-alimentado con el nombre de la página desde la cual el usuario hizo clic, como se muestra en el campo "*Where's the bug*".

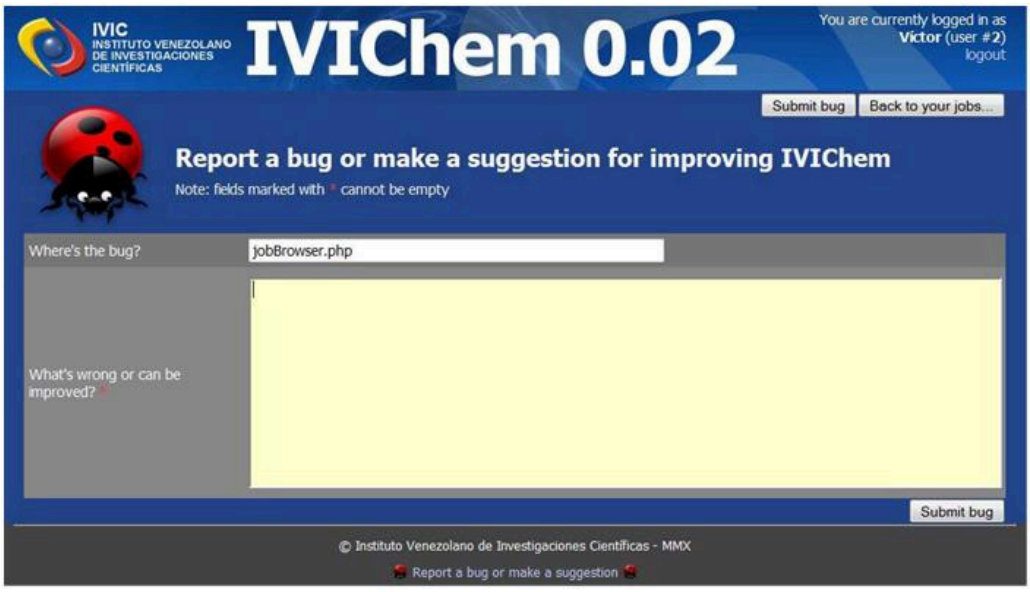

Fig. 4.32: Reportar error o hacer sugerencia

La Figura 4.33 muestra la lista de errores y sugerencias reportados previamente por los usuarios de la aplicación, a través del formulario de la Figura 16.xxvi. Estos errores o sugerencias pueden ser de índole funcional y de usabilidad indistintamente, lo cual

incrementa la utilidad de este componente y permite la evolución constante del sistema facilitando su mantenimiento.

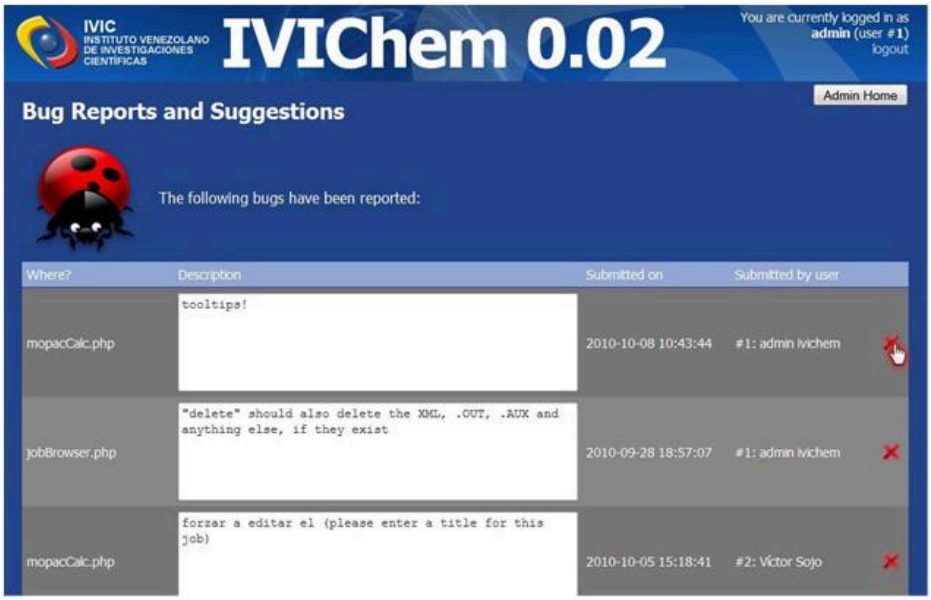

Fig. 4.33: Lista de reportes de errores y sugerencias

El Administrador puede, una vez atendido y resuelto el error o sugerencia, eliminarlo de la lista al presionar la equis roja a la derecha, como indica el cursor en forma de mano. Este formulario fue de gran utilidad en las fases finales de pruebas de aceptación alfa y beta realizada por los usuarios y cuyos resultados fueron registrado en el sistema, como se muestra en la figura 4.33.

6 Conclusiones

AgilUs constituye una propuesta metodológica en concordancia con las exigencias en el desarrollo de software, incorporando la construcción de la usabilidad en el ciclo de vida. Se inscribe en la categoría de métodos ágiles debido a que reduce la cantidad de actividades y artefactos que se generan, propicia la participación de usuario y su carácter iterativo e incremental permite adaptar el desarrollo a los cambios. Incluye la usabilidad a fin de

aplicar un enfoque de diseño centrado en el usuario y como un mecanismo para asegurar la calidad del software. Está orientado al desarrollo de sistemas con alto grado de interactividad con los usuarios (*front-end*) y menos complejidad en la lógica de la aplicación (*back-end*). Como proceso ágil, se trata de determinar los requerimientos y perfiles de los usuarios para presentar un prototipo al usuario que irá evolucionando hasta convertirse en el producto final, sin realizar un diseño estructural.

Aun cando se pudiera pensar que la incorporación de evaluaciones de usabilidad resta agilidad al proceso de desarrollo, se busca fomentar, a través de estas evaluaciones, la participación del usuario y la interacción cara a cara entre éste y el equipo de desarrollo, logrando un equilibrio entre agilidad y producción de software usable. Siguiendo en la línea de la agilidad, las evaluaciones de usabilidad propuestas son rápidas, económicas y no requieren de plataformas tecnológicas complejas.

El método propuesto se ha validado a través de diferentes desarrollos; GEPECO constituye uno de estos casos de estudio, permitiendo detectar errores de usabilidad rápidamente y corregirlos en cada una de las etapas de desarrollo. La guía de estilo facilitó un diseño único de las pantallas. La evaluación heurística permitió mejorar el uso de los colores, redacción de mensajes, entre otros. Así, se consideraron hasta los pequeños detalles en el diseño de la interfaz de usuario y se pudo conocer las opiniones del usuario; logrando desarrollar un producto usable.

El desarrollo de IVIChem se abordó como un problema esencialmente de usabilidad. Por tal razón, se decidió emplear un método de desarrollo de software que fijara el énfasis sobre el desarrollo de interfaces de usuario usables, y que permitiera adaptación de los requerimientos a lo largo de todo el ciclo de vida. El método seleccionado fue AgilUs debido a que sus dos principios esenciales son la agilidad y la usabilidad, permitiendo alcanzar una solución exitosa al problema inicialmente planteado.

La presentación de estos casos de estudio ha permitido mostrar la flexibilidad en el uso de las técnicas de evaluación de usabilidad, las cuales es posible seleccionar depediendode los

casos específicos, lo que sí debe ser considerado es que en cada etapa se debe incluir, al menos, una evaluación de usabilidad a fin de garantizar que la misma se va construyendo durante el desarrollo de la aplicación.

Actualmente se trabaja en la incorporación de aspectos de accesibilidad y la experiencia del usuario en AgilUs, a fin de producir software usable, accesible y que garantice una excelente experiencia al usuario.

7 Referencias Bibliográficas (arreglar)

1. Nielsen, J.: Designing Web usability. New Riders. USA, 2000.
2. ThoughtWorks, www.thoughtworks.com
3. Nielsen, J., Loranger, H.: Usability Prioridad en el diseño Web. Anaya Multimedia. España, 2006.
4. Quesenbery, W.: What Does Usability Mean: Looking Beyond 'Ease of Use'. Proceedings of the 48th Annual Conference, Society for Technical Communication, 2001.
5. La Fundación Sidar - Acceso Universal , www.sidar.org
6. Nielsen, J., Estimating the number of subjects needed for a thinking aloud test, International Journal of Human-Computer Studies, 41, 3, 385-397, 1994.
7. Beck, K. et all.: Manifiesto por el Desarrollo Ágil de Software, 2001. http://www.agilemanifesto.org/iso/es/
8. Fowler, M.: The New Methodology. Última Actualización: 13 diciembre 2005 http://martinfowler.com/articles/newMethodology.html
9. Gabardini, J., Campos, L.: Balanceo de Metodologías Orientadas al Plan y Ágiles. Herramientas para la Selección y Adaptación. PMI Global Congress Proceedings. Argentina, 2004.

10. Acosta, Alecia Eleonora. AgilUs: Construcción ágil de la usabilidad. XXVI Conferencia Latinoamericana de Informática, CLEI, Paraguay. ISSN 978-99967-612-0, 2010.

11. Granollers, T.: MPIu+a. Una Metodología que integra la Ingeniería del Software, la Interacción Persona-Ordenador y la Accesibilidad en el contexto de Equipos de Desarrollo Multidisciplinares. Tesis Doctoral. Universidad de Lérida, España, 2004. http://www.tesisenxarxa.net/TDX-0218107-133615/index.html#documents

12. Ferre, X.: Integration of Usability Techniques into the Software Development Process. International Conference on Software Engineering. Oregon, 2003.

13. Usability Professionals' Association, www.upassoc.org

14. Izarra, M., Guayaquil, A.: Un Generador de Periódicos Comunales: Llevando la Tecnología de la Información a la comunidad. Trabajo Especial de Grado. UCV. Venezuela, 2006

15. Acosta, E., Zambrano, N.: Patterns and Objects for User Interface Construction. *Journal of Object Technology*. Vol. 3, No. 3, pp 75-90. ETH Zurich, 2004. http://www.jot.fm/issues/issue_2004_03/article1

16. Nielsen, J. Ten Usability Heuristics, http://www.useit.com/papers/heuristic/heuristic_list.html

17. Sojo, Víctor. Desarrollo de un ambiente integrador para química computacional, Trabajo de Grado de Maestría, postgrado en Ciencias de la Computación, Universidad Central de Venezuela, 2011.

18. Nielsen, J., Usability Engineering (Interactive Technologies). Morgan Kauffman. ISBN 0-12-518406-9, 1994.

19. Charlton, Samuel G. y O'Brien, Thomas G., Handbook of Human Factors Testing and Evaluation. LEA Inc. ISBN 0-8058-3290-4, 2001.