# Metodología ágil Crystal Clear. Un caso de estudio

## *Agile methodology Crystal Clear. A study case*

**Rogelio Barroso Abreu[1]*, Yoanna Oliveros Guntín[1], Yanniel Álvarez Alfonso[1], Jorge Coello Mena[1], Lenia García Álvarez de la Campa[2]**

[1]CUJAE

[2]Gran Kaiman Teleco S.A

*{rfbarroso, borloros, yalvareza, lcoello}@ceis.cujae.edu.cu*

**Resumen**

Las metodologías ágiles están generado un interés cada vez mayor en el mundo del software. Son caracterizadas como antídoto a la burocracia de las metodologías tradicionales ya que se basan en la velocidad y en la simplicidad a la hora de desarrollar software. Muchos autores plantean que lo que es nuevo en las metodologías ágiles no son las prácticas que ellas siguen sino el reconocimiento de que son las personas las principales responsables en el éxito de un proyecto. Las ágiles son especialmente adecuadas para proyectos donde el entorno del sistema deseado es muy cambiante y en donde se exige reducir drásticamente los tiempos de desarrollo manteniendo una alta calidad. En el artículo se mencionan las principales características de las metodologías ágiles; cuáles son las más extendidas y populares en la actualidad y se muestran los resultados de la aplicación de la metodología ágil para el desarrollo de software Crystal Clear a un caso de estudio.

**Palabras clave:** Crystal Clear, familia Crystal, metodologías ágiles.

***Abstract***

*The agile methodologies are generating an interest every bigger time in the world of software development. They have been characterized as antidote to the bureaucracy of the traditional methodologies since they are based on the speed and simplicity when developing software. Many authors outline that what is new in the agile methodologies is not the practices that they continue but the recognition that they are people the main ones responsible in the success of a project. They are especially appropriate for projects where the environment of the wanted system is very changing and where is demanded to reduce the times of development drastically maintaining a high quality. Presently article is carried out a study on the principles and general characteristic of the agile methodologies; which are the most extended and popular at the present time and is carried out his application in a case of study and the results are analyzed.*

***Key words:*** *Agile methodologies, Crystal Clear, Crystal family.*

## Introducción

Hasta hace unos años el desarrollo de software se realizaba de forma poco organizada. Hoy en día existen numerosas propuestas metodológicas que inciden en distintas dimensiones del proceso de desarrollo para guiarlo, hacerlo más eficiente y lograr alcanzar el objetivo principal: obtener un software sin errores, en el tiempo convenido y que satisfaga las necesidades del cliente. Estas propuestas siguen dos vertientes fundamentales: las que se conocen como metodologías tradicionales, que se centran fundamentalmente en el control del proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir, y las herramientas y notaciones que se usarán; y las metodologías ágiles, que sin eliminar totalmente el control del

proceso sí lo realizan de forma más liberal, centrándose en otras dimensiones como el factor humano o el producto de software final.

A pesar de que las metodologías ágiles surgieron a mediados de la década de 1990 recién ahora se están abriendo camino como un tópico serio de investigación académica. No obstante a esto el crecimiento de los métodos ágiles y su penetración en la industria del software ocurren a un ritmo intenso. Desde que surgieron, según el Cutter Consortium [1], ya el 50% de las empresas definen como ágiles más de la mitad de los métodos empleados en sus proyectos [2].

Se define un proceso como ágil cuando el desarrollo de software es incremental, con entregas pequeñas de software funcional en ciclos rápidos; es cooperativo, el cliente y los desarrolladores trabajan juntos constantemente y mantienen una cercana comunicación; es sencillo, ya que el método en sí mismo es fácil de aprender, modificar y está bien documentado; y es adaptable, permitiendo realizar cambios en el mismo [3]. Las metodologías ágiles presentan otras características novedosas: son muy adaptables a los cambios en el sistema deseado por el cliente, y se orientan más a las personas que al proceso de desarrollo: el cambio para las metodologías ágiles es bienvenido puesto que intentan ser procesos que se adaptan y crecen en el cambio; y siguen el principio de que ningún proceso podrá nunca disimular o tapar las habilidades del equipo de desarrollo, de modo que el papel del proceso es apoyar al equipo de desarrollo en su trabajo y no viceversa. Explícitamente puntualizan el trabajar a favor de la naturaleza humana y enfatizan que el desarrollo de software debe ser una actividad agradable [4]. También se basan en la colaboración cercana entre el equipo de desarrollo y los expertos del negocio (cliente); en la comunicación cara a cara, como forma más efectiva de intercambio y comunicación que la documentación escrita; en la entrega frecuente de versiones con valor para el cliente; y en la necesidad de un equipo de desarrollo unido y organizado. El principal atractivo que tienen estas metodologías para los desarrolladores radica en su oposición a la burocracia, o sea, son menos orientadas al documento exigiendo una cantidad más pequeña de documentación para una tarea dada y centrándose mucho más en el código, siguiendo un camino que propone como parte importante de la documentación al código fuente [4].

Existen unas cuantas metodologías ágiles descritas en la literatura sobre el tema, reconocidas por los especialistas y con casos documentados de su aplicación en proyectos. Entre las seis metodologías ágiles que mayor popularidad han alcanzado se encuentran: Programación Extrema [5] - [ 6], Scrum [7], Desarrollo Manejado por Rasgos [8], Método Dinámico de Desarrollo de Sistemas [9], Desarrollo de Software Adaptable [10] y Crystal Clear [11] - [ 12] - [13].

**Crystal Clear**

Crystal Clear es la menor de la familia de metodologías Crystal desarrollada por el investigador de IBM el Dr. Alistair Cockburn. No existe una única metodología Crystal puesto que su creador cree que tipos diferentes de proyectos requieren tipos diferentes de metodologías; y que cada equipo debe, a partir de estas características comunes, crear nuevos miembros de la familia. Cockburn valora esta variación a lo largo de dos ejes: el número de personas involucradas en el proyecto y las consecuencias de los errores que tenga el software. Crystal Clear está diseñada para ser utilizada por equipos de hasta ocho integrantes y en el desarrollo de sistemas cuyos posibles errores puedan causar una pérdida prudencial de dinero o de confort. Una definición de Crystal Clear, dada por su propio creador, sería la siguiente: Crystal Clear es una metodología centrada en el factor humano, donde un diseñador líder y de dos a siete desarrolladores más se encuentran juntos en un local grande o en locales adyacentes con radiadores de información como pizarras y diagramas bien visibles en la pared, teniendo acceso fácil a usuarios claves; eliminando las distracciones; entregando código funcional, testeado y utilizable en intervalos de uno a tres meses; reflexionando periódicamente y ajustando continuamente su estilo de trabajo [11].

Siguiendo su idea de que cada proyecto requiere una metodología propia, Cockburn no especifica completamente Crystal Clear, definiéndola como la metodología para equipos pequeños más tolerante y flexible que aún pudiera guiar exitosamente el desarrollo de un proyecto de software [12].

Crystal Clear consiste en una serie de prioridades, propiedades, estrategias, técnicas, ciclos, roles y artefactos.

Las prioridades son mantener la seguridad del proyecto, desarrollar de forma eficiente y mantener la habitabilidad de las convenciones. Cockburn describe Crystal Clear no a través de las estrategias y técnicas que se deben aplicar, sino mediante una serie de propiedades claves. Esto lo hace motivado por el hecho de que en general los procedimientos que se sigan no necesariamente producen o cumplen las propiedades, y que otros procedimientos diferentes a los elegidos pueden producir esas propiedades para un equipo particular, por tanto es más importante cumplir las propiedades que seguir determinados procedimientos. Por este motivo ninguna de las estrategias, técnicas o artefactos que Crystal Clear define es de uso obligado y pueden usarse cualquiera de las descritas en otras metodologías ágiles. Crystal Clear se centra en tres propiedades claves, que constituyen su núcleo: efectuar Entregas Frecuentes, que consiste en liberar código ejecutable y testeado a usuarios reales cada pocas semanas o meses; realizar una Mejora Reflexiva dedicando todo el equipo unido un pequeño tiempo para determinar qué está y qué no está funcionado, discutir qué se puede mejorar y qué se debe mantener; y el requerimiento de Crystal Clear en cuanto al tamaño del equipo y su ubicación física en un mismo local permite crear Comunicación Osmótica y que la información esté "flotando" en el ambiente, así el equipo puede obtenerla como si fuera por ósmosis. Otras propiedades importantes que pueden ser añadidas para incrementar la seguridad del proyecto son: crear Confianza Personal entre los miembros del equipo, mantener el Foco en la tarea, tener Acceso Fácil a Usuarios Expertos y trabajar en un Ambiente Técnico con pruebas automáticas, administración de configuración e integración frecuente [13].

Las estrategias que propone Crystal Clear son: Exploración de 360º, Victoria Temprana, Esqueleto Ambulante, Rearquitectura Incremental y Radiadores de Información. Las tres primeras guían el camino del equipo durante los primeros momentos del desarrollo y las dos restantes pueden aplicarse durante todo el proyecto [13].

Las técnicas propuestas por Crystal Clear permiten aplicar las estrategias mencionadas anteriormente. Las técnicas son: Taller de Perfilación de la Metodología, Talleres de Reflexión, Planeación Rápida, Estimación Delfos, Reuniones Diarias, Diseño de Interacciones Esenciales, Miniatura del Proceso, Programación Lado a Lado y Gráficos de Quemado. Estas permiten fijar y mejorar continuamente las convenciones de trabajo del equipo, realizar la planeación del proyecto a corto y a largo plazo, hacer estimaciones en cuanto a tamaño y duración del mismo, mantener seguimiento sobre el avance del proyecto, etc. y conforman un buen punto de inicio para equipos que empiecen a aplicar Crystal Clear hasta que las vayan refinando o encuentren otras nuevas que se adapten mejor a su estilo de trabajo [13].

En Crystal Clear existen ocho roles nominados: Patrocinador Ejecutivo, Usuario Embajador, Diseñador Líder, Diseñador-Programador, Experto del Negocio, Coordinador, Verificador y Escritor. Los cuatro primeros necesariamente deben ser desempeñados por personas distintas, mientras que los restantes pueden ser roles adicionales asignados a algunos miembros del equipo. Cada rol es responsable de crear y mantener actualizados determinados artefactos. El número y formalismo de los artefactos intermedios en Crystal Clear es considerablemente reducido ya que el equipo suple la necesidad de estos con las Entregas Frecuentes, la Comunicación Osmótica, el Acceso Fácil a Usuarios Expertos y la estrategia Radiadores de Información. Los artefactos que Crystal Clear propone constituyen sólo una plataforma de inicio. El equipo puede agregar, eliminar, modificar o sustituir artefactos de acuerdo a las técnicas, tecnologías, hábitos de comunicación e incluso estilos de trabajo que tenga para crear así su propia metodología.

Crystal Clear define su proceso como un conjunto de ciclos anidados de diferentes duraciones. Cada ciclo tiene su propia secuencia, y pueden desarrollarse simultáneamente varias actividades pertenecientes a distintos ciclos. Crystal Clear posee siete

ciclos: el Ciclo del Proyecto, que abarca todo el desarrollo del software. El Proyecto tiene varios Ciclos de Entrega, en los que se libera una parte determinada del sistema y tienen de una semana a tres meses de duración. Cada Ciclo de Entrega tiene al menos un Ciclo de Iteración, que es una unidad de planeación, desarrollo y celebración que según la cantidad de iteraciones que se planifiquen tiene de una semana a tres meses de duración. Crystal Clear requiere múltiples Entregas por proyecto pero no muchas Iteraciones por Entrega. Cada Iteración se divide en Ciclos Semanales y la Semana se divide en Ciclos Diarios. Cada Día ocurren varios Ciclos de Integración, que son una unidad de desarrollo, integración y prueba del sistema, de media hora a tres días de duración. Dentro de cada Ciclo de Integración ocurren múltiples Episodios de Desarrollo, que comprenden la escritura y comprobación de una sección de código, con una duración de pocos minutos a algunas horas. En la Figura 1 se aprecia una expansión de cada ciclo por separado con las actividades específicas que ocurren en cada uno de ellos. El Ciclo del Proyecto comienza con la actividad de preparación (P) y finaliza con la de empaquetamiento (Ep) o puesta a punto del sistema. La actividad de entrega (E) de la parte del sistema desarrollada marca el fin del Ciclo de Entrega. Cada Ciclo de Iteración comienza con la planificación de la iteración (p), y al finalizar esta el equipo reflexiona y celebra (R). Al final del Ciclo de Integración se produce la integración y prueba (i) de la parte del sistema desarrollada. Cada Día el equipo realiza reuniones diarias (r) y ocurren numerosos Episodios donde se desarrolla (d) y comprueba (c) el código fuente.

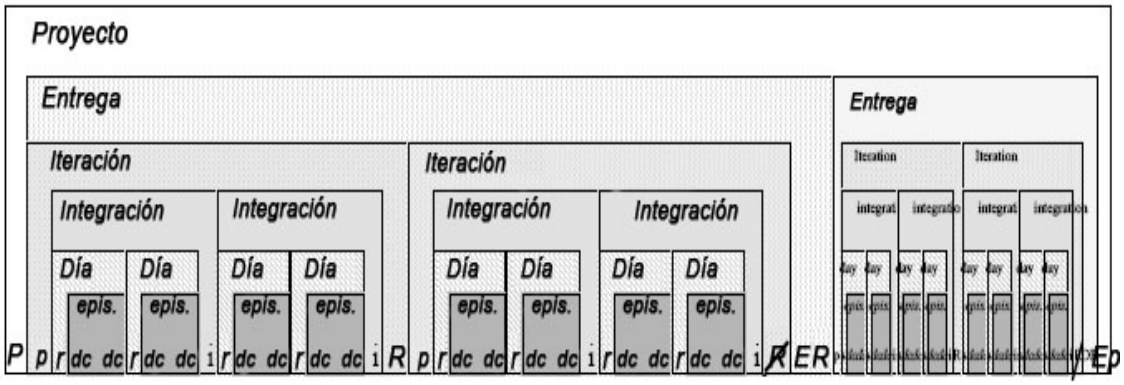


Fig. 1. Ciclo de Crystal Clear con actividades específicas de cada ciclo (Adaptado de [11]).

Debido al bajo nivel de detalles y especificaciones existente en la bibliografía disponible sobre Crystal Clear se confeccionó una guía de aplicación, disponible en [13], como referencia inicial para los equipos que deseen utilizar Crystal Clear como metodología de desarrollo y se aplicó la metodología en un caso de estudio, detallado en [13], utilizando la guía de aplicación creada.

**Resultados del caso de estudio**

Para el desarrollo del caso de estudio se simuló la siguiente situación: el cliente posee una empresa de juegos educativos y desea lanzar un nuevo producto basado en el Juego del Ahorcado para ser usado por niños y de esta forma ayudarlos en el aprendizaje de nuevas palabras.

Se desea que el sistema posea todas las características del clásico Juego del Ahorcado: el niño debe poder adivinar las palabras letra a letra y el sistema debe mostrar el árbol y el ahorcado según el niño juegue y falle en la letra que haya seleccionado. El sistema debe constar de 2 niveles: fácil y avanzado. En el nivel fácil el niño puede ver la definición de la palabra a adivinar mientras que en el avanzado no. Los niños tienen cinco oportunidades de fallo para adivinar las letras que forman la palabra. Las palabras deben ser cargadas desde un fichero texto con el formato palabra: definición en cada línea y deben ser mostradas aleatoriamente.

El juego debe estar terminado en 15 días y no se debe reusar ningún tipo de código porque el cliente desea tener todos los derechos sobre el software. Puede ser desarrollado usando cualquier lenguaje de programación, pero el juego debe ser multiplataforma. El cliente quiere conocer constantemente como va el avance del desarrollo del producto y plantea que estará todo el tiempo a disposición del equipo de desarrollo.

En la etapa de preparación se realizó una reunión inicial con el cliente. Ese día se conformó el equipo de trabajo y se fijaron requerimientos importantes sobre el juego y el futuro software. Se determinó que el equipo iba a tener cuatro miembros trabajando en un mismo local y se realizó la distribución de roles como se aprecia en la Figura 2.

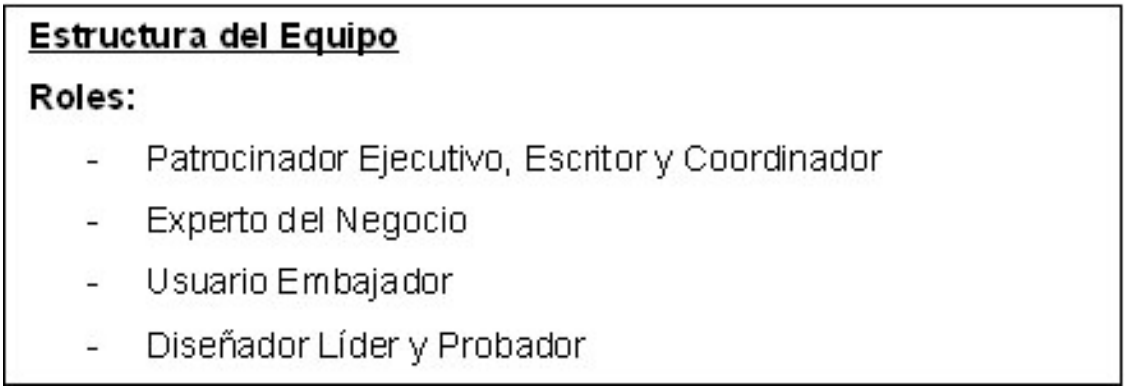


Fig. 2. Artefacto Estructura del Equipo [13].

Después de concluir la reunión se realizó la Exploración de 360º. El Patrocinador Ejecutivo creó la Declaración de Misión y Prioridades de Compromiso. Una vez fijadas las prioridades se comenzó a aplicar la técnica Diseño de Interacciones Esenciales: el Usuario Embajador creó el Modelo de Roles, identificándose un único rol, Jugador, con cuatro objetivos a cumplir en el sistema: iniciar nuevo juego, cambiar nivel de dificultad, seleccionar letra y cerrar el juego. A partir del Modelo de Roles, el Usuario Embajador y el Experto del Negocio crearon la Lista de Actores – Objetivos y a partir de esta confeccionaron un modelo de las tareas que el sistema debía realizar para satisfacer las necesidades del rol Jugador e identificaron y describieron los Casos de Uso del sistema que fueron añadidos al Archivo de Requerimientos junto a la Lista de Actores - Objetivos y a la Declaración de Misión y Prioridades de Compromiso. En el Archivo de Requerimientos, creado por el Experto del Negocio, se listaron además los requisitos que el cliente deseaba que cumpliera el sistema. Al concluir la Exploración de 360º el equipo analizó la información capturada hasta el momento y decidió que el proyecto era factible de realizarse.

Posteriormente se realizó la técnica Perfilación de la Metodología con el propósito de definir las convenciones que se iban a seguir durante el desarrollo del proyecto. Durante la Perfilación se analizaron las propiedades, estrategias y técnicas que Crystal Clear propone. Se decidió intentar cumplir con las siete propiedades de Crystal Clear y que las estrategias serían aplicadas en su totalidad. Con respecto a las técnicas se decidió no implementar la Miniatura del Proceso ni la Programación Lado a Lado. Se acordó realizar los Talleres de Reflexión al final de cada iteración.

El equipo realiza la técnica Estimación Delfos y concluyó que los factores que más peso tendrían en la realización del proyecto eran la interfaz del sistema y la clase que manejaría la lógica del juego y se estimó que le tomaría al Diseñador - Líder dos semanas implementarlas completamente.

Se realizó la primera planificación del proyecto usando la técnica Planeación Rápida. Debido a la brevedad del proyecto se decidió realizar la planificación completa de este. El Experto del Negocio, el Diseñador Líder y el Patrocinador Ejecutivo en el papel de Coordinador crearon el Mapa del Proyecto y se determinó realizar dos Entregas de dos Iteraciones cada una con una Vista del Usuario planificada al final de cada Iteración. Se acordó que las Vistas y los Talleres de Reflexión tendrían una duración

de medio día y se determinó cuáles tareas formarían el Esqueleto Ambulante, cuáles se implementarían en el primer Ciclo de Entrega y cuáles en el segundo. El Coordinador y el Diseñador Líder crearon el Plan de Liberaciones a partir del Mapa del Proyecto. El Coordinador creó el Cronograma de Vistas del Usuario, acordando el equipo tener una Vista del Usuario al final de cada Iteración; y el Estado del Proyecto con el listado de tareas a realizar y un gráfico de quemado como se aprecia en la Figura 3. Una vez finalizada la planificación se inició el primer Ciclo de Entrega.

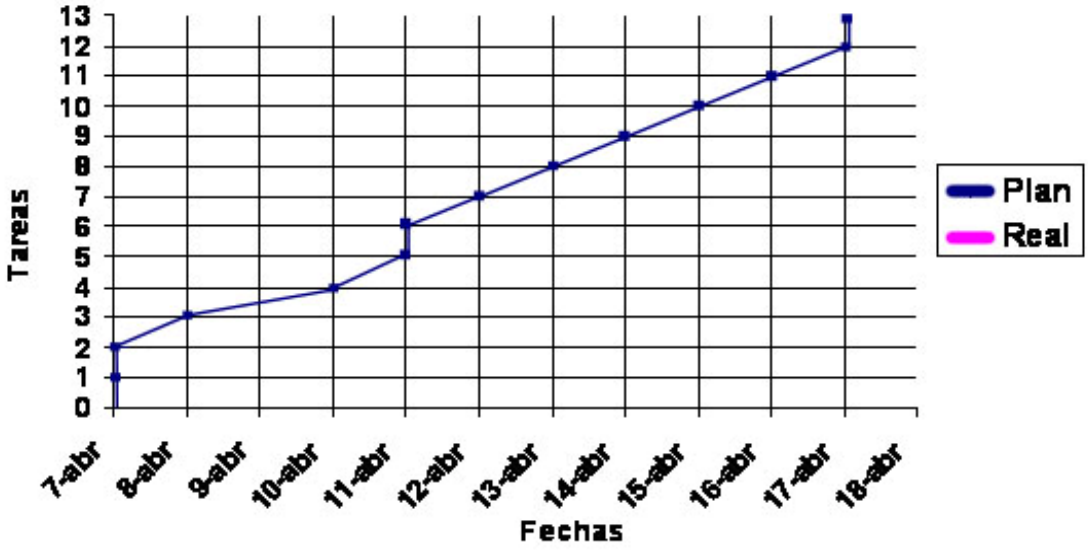


Fig. 3. Gráfico de Quemado por fechas de finalización de las tareas [13].

Al iniciarse el primer Ciclo de Entrega del proyecto no fue necesario revisar el plan puesto que estaba recién creado. Al comienzo de la primera Iteración el Diseñador Líder realizó una propuesta de arquitectura inicial para construir el Esqueleto Ambulante. Durante los Episodios de Desarrollo del Esqueleto Ambulante el Diseñador Líder creó el Modelo Común del Dominio en el que se definieron las clases necesarias para implementar el juego. Se usó el Plan de la Iteración como Estado de la Iteración mostrándose las tareas a ejecutar y su estado en cada momento de la iteración. Al concluirse la implementación del Esqueleto Ambulante se realizó la Vista del Usuario en la cual el cliente quedó muy satisfecho al ver el software con una funcionalidad completa desarrollada en tan poco tiempo. No hizo ningún señalamiento ni agregó información. Se realizó el Taller de Reflexión en el cual el equipo acordó mantener las normas y convenciones fijadas como se aprecia en la Figura 4.

**Resultados del Taller de Reflexión Primera Iteración**

**Mantener:**

1. Las normas y convenciones acordadas.

**Intentar:**

1. Aplicar la convención de no correo, internet y teléfono.

**Problemas:**

1. El equipo sufre demasiadas interrupciones.
2. No se está manteniendo el foco en la tarea.

Fig. 4. Artefacto Resultados del Taller de Reflexión de la primera iteración [13].

Dentro del Taller de Reflexión el equipo decidió continuar el avance del proyecto según la planificación prevista y comenzar la Segunda Iteración.

Teniendo en cuenta las funcionalidades a desarrollar en esta Iteración el Diseñador Líder evolucionó la Arquitectura del Sistema añadiendo una interfaz para realizar el cambio de nivel de dificultad durante el juego. Este cambio no afectó el Modelo Común del Dominio. Se realizaron todas las tareas planificadas en la Iteración en los Episodios de Desarrollo y se chequeó constantemente el estado de las mismas. Durante la realización de una de las tareas surgió una duda sobre los requerimientos del sistema y esta fue rápidamente aclarada por el Experto del Negocio y el Usuario Embajador. Al finalizar la implementación de las tareas planificadas para la iteración se produjo la Segunda Vista planificada del Usuario. El cliente estuvo conforme con la interfaz de juego creada pero hizo algunos señalamientos y realizó cambios importantes en los requerimientos iniciales. Después de la Vista se realizó el Taller de Reflexión. No se detectaron problemas de ningún tipo y se acordó mantenerse trabajando de la misma forma. Se concluyó el primer Ciclo de Entrega del proyecto y producto de la importancia que poseían los cambios requeridos por el cliente para la Liberación recién finalizada se decidió realizar una replanificación del proyecto en el comienzo del segundo Ciclo de Entrega.

En el inicio del segundo Ciclo de Entrega se actualizó el Archivo de Requerimientos y los Casos de Uso con la información obtenida de la segunda Vista del Usuario. Se realizó nuevamente la técnica Planeación Rápida adicionándose nuevas tareas al Mapa del Proyecto. Se actualizaron el Plan de Liberaciones y el Calendario de Vistas añadiéndose las nuevas tareas a la primera Iteración de la segunda Entrega en el Plan de Liberaciones. Después de un análisis se decidió que los nuevos requerimientos no implicaban cambios en el Modelo de Roles, en la Lista de Actores-Objetivo, ni en la Arquitectura del Sistema. Se actualizó el Estado del Proyecto y el gráfico de quemado, iniciándose la primera Iteración a partir de la nueva planificación realizada. Al comienzo de la Primera Iteración el Coordinador junto al equipo construyó el Plan de la Iteración y según se fueron realizando las tareas planificadas para la Iteración fue actualizando el Estado de la Iteración. Al concluirse las tareas se realizó la Vista del Usuario. El cliente quedó muy satisfecho al ver el software casi completamente terminado y en el plazo convenido. No hizo ningún otro señalamiento ni agregó información. El equipo realizó el Taller de Reflexión en el que tampoco se detectaron problemas y se mantuvieron las convenciones del Taller anterior. El equipo decidió continuar el avance del proyecto según la planificación prevista. Los objetivos de la Segunda Iteración eran probar completamente el sistema, crear la ayuda del juego y preparar el sistema para entregar su versión final al cliente. El Coordinador creó el Plan de la Iteración y actualizó el Estado de la Iteración según se fueron realizando las tareas. Anteriormente durante los Episodios de Desarrollo el Diseñador Líder implementó y usó Pruebas ejecutables para comprobar la calidad del código fuente. En esta iteración el Probador creó casos de prueba donde se verificaron todos los casos extremos del sistema. Una vez probado el sistema se escribió la ayuda del juego. Al concluirse todas las tareas se produjo la Vista del Usuario final en la que el cliente probó exhaustivamente el juego y se le entregó listo para ser usado. El equipo posteriormente realizó el último Taller de Reflexión donde se discutió el proceso seguido y las técnicas y estrategias aplicadas. Se guardaron las Convenciones finales acordadas como referencia para futuros proyectos de software y se concluyó el caso de estudio.

Durante el desarrollo del caso de estudio se apreció que Crystal Clear es una metodología sencilla de aprender e implementar. Permite realizar la planificación para períodos cortos de tiempo, esto le da gran capacidad de asimilación y replanificación ante requisitos cambiantes. La entrega continua y en plazos cortos de software funcional y el trabajo conjunto entre el cliente y el equipo de desarrollo permiten mantener al equipo siempre bien informado de que es lo que el cliente desea exactamente y que cambios y mejoras deben hacerse a las partes entregadas para satisfacer las necesidades de los usuarios. Crystal Clear realiza una mejora continua de su proceso de desarrollo, esto permite que el equipo se adapte fácilmente a las necesidades del proyecto y se sienta a gusto con la manera en que se trabaja. En Crystal Clear puede empezarse a implementar el sistema aún sin haberse hecho una captura completa de requisitos, el nivel de detalle de la documentación escrita es bajo y los productos del trabajo son fáciles de crear y actualizar; esto permite al equipo dedicar más tiempo a programar el sistema y disminuir los tiempos de desarrollo y

por último se apreció que Crystal Clear promueve la comunicación entre los miembros del equipo mediante actividades que fortalecen el trabajo colectivo.

**Conclusiones**

Hoy en día las metodologías ágiles constituyen uno de los temas de más actualidad y debate dentro de la Ingeniería de Software. Su rechazo a la burocracia y a la documentación innecesaria hace que gocen de gran popularidad en una parte, cada vez mayor, del mundo del desarrollo de software. Sin embargo esta misma característica conspira contra su expansión e implementación ya que se hace difícil tener guías útiles de aplicación de dichas metodologías. Este trabajo pretendió servir como una guía inicial para aquellos que deseen utilizar Crystal Clear en sus proyectos de desarrollo de software.

**Referencias Bibliográficas**


1. A. Cockburn. "Agile Software Development". ISBN 0-201-69969-9 C.-H. S. Editor, 2001.
2. A. Cockburn. "Crystal Clear: A Human-Powered Methodology for Small Teams". ISBN 0-201-69947-8. Addison-Wesley, 2005.
3. C. Consortium. "About Cutter". Disponible en: http://www.cutter.com/about-cutter.html. [Accedida: 30/5/2008].
4. C. Reynoso. "Métodos Heterodoxos en el Desarrollo de Software". Disponible en: http://www.microsoft.com/spanish/msdn/arquitectura/roadmap_arq/heterodox.mspx [Accedida: 12/9/2007].
5. D. Consortium. "DSDM Consortium". Disponible en: http://www.dsdm.org/ [Accedida: 16/9/2008].
6. J. Highsmith. "Messy, Exciting, and Anxiety-ridden: Adaptive Software Development". Disponible en: http://www.jimhighsmith.com/articles/messy.htm [Accedida: 16/9/2008].
7. K. Beck. "Extreme Programming Explained: Embrace Change". ISBN 0-201-61641-6. Addison-Wesley, 2000.
8. K. Schwaber. "What is Scrum?" Disponible en: http://www.controlchaos.com/about/index.php [Accedida: 24/3/2008].
9. M. Fowler. "The New Methodology". Disponible en: http://www.martinfowler.com/articles/newMethodology.html [Accedida: 12/9/2007].
10. N. P. Ltd. "Feature Driven Development ". Disponible en: http://www.featuredrivendevelopment.com/ [Accedida: 16/9/2008].
11. P. Letelier. "Proceso de desarrollo de software". Universidad Politécnica de Valencia. Disponible en: https://pid.dsic.upv.es/C1/Material/Documentos%20Disponibles/Introducci%C3%B3n%20Proceso%20de%20Desarrollo%20de%20SW.doc
12. R. B. Abreu. "Investigación de la Metodología Ágil Crystal Clear y su Aplicación a un Caso de Estudio". Y. O. Guntín y L. G. A. d. l. Campa (Tutor). Tesis de Grado. La Habana, Instituto Superior Politécnico José Antonio Echeverría. 2008.
13. R. Jeffries. "What is Extreme Programming?" XP Magazine, Disponible en: http://www.xprogramming.com/xpmag/whatisxp.htm [Accedida: 24/3/2008].