IX Jornadas de Ciencia y Tecnología
de Facultades de Ingeniería del NOA
Santiago del Estero, 3 y 4 de Octubre de 2013

XPE: Un enfoque híbrido

C. Marcelo Pérez Ibarra[1], Pablo E. Márquez Ruíz[1] & Darío Zavoluk[1]

(1) Grupo de Ingeniería de Software, Facultad de Ingeniería, Universidad Nacional de Jujuy.
pablo.marquez88@gmail.com, zavolukadg@gmail.com, cmperezi@gmail.com

RESUMEN: En este trabajo se presenta una propuesta metodológica para el desarrollo de software que se sustenta en las metodologías ágiles, patrones de diseño y la filosofía de reutilización. La propuesta integra al ciclo de vida de la Programación Extrema el uso de patrones de diseño y la reutilización de soluciones de proyectos anteriores lo que deriva en la modificación de las fases de este ciclo de vida. A fin de incluir en el diseño de los componentes software la aplicación de patrones y la reutilización de soluciones desarrolladas previamente se plantean cambios en el modelo de documentación de la Programación Extrema y se enfoca el desarrollo hacia las pruebas, no sólo como proceso de validación sino también como hilo conductor del desarrollo. Además se presenta un caso de estudio que muestra la aplicación de este enfoque híbrido al desarrollo de un sistema de gestión de activos.

1 INTRODUCCIÓN

En la literatura existen numerosas propuestas metodológicas que inciden en distintas dimensiones del proceso de desarrollo de software. Las más tradicionales se centran en el control del proceso y establecen rigurosamente las actividades involucradas, las soluciones que se deben producir, las herramientas y notaciones que se usarán. La rigurosidad impuesta por estas metodologías obliga a que el cliente tome la mayor parte de las decisiones al iniciar el proyecto, consecuentemente, un cambio en los requisitos puede impactar significativamente en el costo del proyecto según el grado de avance de éste. Si bien el enfoque clásico demostró ser efectivo y necesario en un gran número de proyectos, también es cierto que presentó problemas al aplicarse en otros casos (Pfleeger, 2010) (Sommerville, 2005).

Hoy, la demanda de productos que satisfagan las urgentes necesidades de un mercado creciente, hace necesario que el desarrollo de software sea capaz de acelerar el proceso de construcción y adaptar continua y rápidamente cambios en los requisitos, de modo que se consiga un producto de calidad en el menor tiempo posible. Ante la dificultad de utilizar metodologías tradicionales para cumplir con los plazos de tiempo y la flexibilidad requerida por los proyectos actuales, es que muchos equipos de desarrollo asumen el riesgo de trabajar con las llamadas metodologías ágiles.

Las metodologías ágiles, especialmente orientadas a proyectos pequeños, constituyen una solución a medida que aporta una elevada simplificación sin dejar de lado las prácticas esenciales que aseguren la calidad del producto. En esta filosofía pequeños equipos de proyecto altamente motivados aplican métodos informales a fin de generar entregas tempranas de software incremental cuya evaluación a través de una comunicación continua y activa entre desarrolladores y clientes permite adaptar el proceso de desarrollo. Así, con un mínimo de productos de trabajo de ingeniería del software se consigue una simplicidad general del desarrollo. El enfoque ágil representa entonces una opción razonable a las metodologías tradicionales para ciertas clases de software y tipos de proyecto (Pressman, 2006).

En este trabajo se presenta una propuesta metodológica que integra a las fases de desarrollo de la metodología ágil Programación Extrema el uso de patrones de diseño de modo que se aprovechen las ventajas de ambas filosofías para lograr un desarrollo rápido sustentado en el formalismo de los patrones de diseñado.

El presente artículo se organiza como sigue: en el apartado 2 se describen la metodología ágil Programación Extrema y los patrones de diseño, en el apartado 3 se presenta el enfoque híbrido propuesto en este trabajo, en el apartado 4 se muestra un caso de estudio y en el apartado 5 se discuten las conclusiones del trabajo.

2 ANTECEDENTES

En este apartado se presentan brevemente los conceptos que dan sustento al trabajo realizado.

2.1 *Programación Extrema*

La permanencia y éxito de las empresas en el actual contexto de negocios, requiere que éstas se adapten rápidamente a los cambios en su entorno, tanto en lo que refiere a políticas organizacionales

como a la adopción de nuevas tecnologías. Respecto a esto último, la construcción del software necesario para agilizar los procesos de negocio que permitan encarar los nuevos desafíos y oportunidades, presenta severas restricciones de tiempo.

Considerando esta realidad, los métodos ágiles de desarrollo de software constituyen una alternativa más que atractiva al proporcionar un enfoque rápido para el desarrollo de sistemas.

Las metodologías ágiles dan soporte a los requerimientos cambiantes de los usuarios y permiten entregar el producto en el menor tiempo posible. Para ello, durante el desarrollo los usuarios pueden proponer cambios que se implementan a través de un proceso iterativo (Sommerville, 2007).

La Programación Extrema (eXtreme Programing o XP) es uno de los métodos agiles más conocido y más ampliamente utilizado. Este enfoque se basa en el desarrollo iterativo, el prototipado y la participación del cliente. XP hace hincapié en el equipo de trabajo, en el que desarrolladores y usuarios colaboran intensamente para alcanzar una alta productividad. La filosofía de trabajo de XP permite acomodar cambios incluso en etapas avanzadas del desarrollo del producto. La Fig. 1 muestra el ciclo de vida de la Programación Extrema (Hurtado & Bastiarrica, 2005).

Como puede observase en la Fig. 1 el punto de partida de XP son las historias de usuario. Los requisitos planteados como historias describen las características y funcionalidad requeridas para el software. Estas historias derivan posteriormente en tareas de desarrollo que permiten establecer el diseño (el más simple posible) para la implementación de cada historia, el tiempo requerido para entregar cada incremento y el conjunto de pruebas adecuado. Antes del desarrollo cada incremento se planifican las pruebas unitarias que deberán superar los componentes a construir. La programación de estos componentes se realiza en parejas observando que se sigan los estándares de codificación y que lo codificado coincida con lo especificado en las historias. Cada lanzamiento (generación del incremento) permite al usuario evaluar (pruebas de aceptación) las características visibles del software y reformular sus historias en caso de ser necesario. Este proceso iterativo se lleva a cabo hasta alcanzar la satisfacción del cliente (Pressman, 2006). La Fig. 2 muestra las actividades genéricas contempladas en el marco de trabajo de XP.

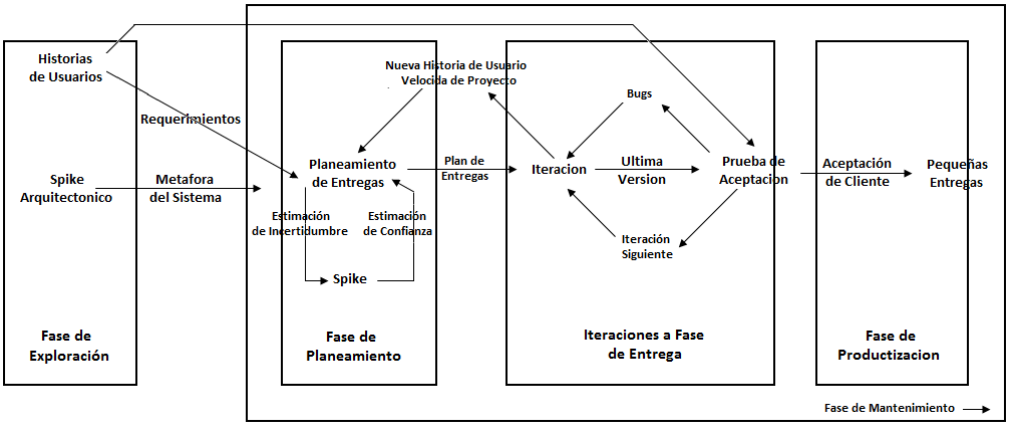


Figura 1. Ciclo de vida de XP.

Figura 2. Actividades del marco de trabajo de XP.

2.2 *Patrones de Diseño*

En los años 90 Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides (Gang of Four o GoF, que en español significa la pandilla de los cuatro) escribieron el famoso libro "Design Patterns: Elements of Reusable Object Oriented Sofware" en el que recopilaron y documentaron 23 patrones de diseño aplicados usualmente por expertos diseñadores de software orientado a objetos. Este fue el inicio del uso de patrones en el desarrollo de software.

Los patrones de diseño describen estructuras que resuelven un problema de diseño particular dentro de un contexto específico. Cada patrón representa una solución a un problema que aparece de forma recurrente en el desarrollo de software. En otras palabras, los patrones brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares. De forma general, puede definirse a los patrones de diseño como el esqueleto de las soluciones a problemas comunes en el desarrollo de software (Pressman, 2006)

Es así que el uso de patrones de diseño puede contribuir al proceso de desarrollo de software proporcionando catálogos de elementos reusables en el diseño de sistemas; evitando la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente; formalizando un vocabulario común entre diseñadores; estandarizando el modo en que se realiza el diseño y facilitando el aprendizaje de las nuevas generaciones de diseñadores al condensar el conocimiento existente. Asimismo, no impone ciertas alternativas de diseño frente a otras y no pretende eliminar la creatividad inherente al proceso de diseño.

Según GoF considerando el propósito de los patrones de diseño puede clasificárselos en 3 grandes categorías (Gamma et al., 2003):

- Patrones Creacionales: se encargan de las formas de crear instancias de objetos, es decir, tienen por finalidad abstraer el proceso de instanciación y ocultar los detalles de cómo los objetos son creados o inicializados.
- Patrones Estructurales: separan la interfaz de la implementación, se ocupan de cómo las clases y objetos se componen o agrupan, para formar estructuras más grandes. La flexibilidad añadida mediante la composición de objetos viene dada por la capacidad de cambiar la composición en tiempo de ejecución, que es imposible con la composición de clases.
- Patrones de Comportamiento: describen la comunicación entre objetos y clases, están relacionados con algoritmos y asignación de responsabilidades. Estos patrones se focalizan en el flujo de control dentro de un sistema. Algunos ejemplos de estos patrones incluyen la definición de abstracciones de algoritmos, las colaboraciones entre objetos para realizar tareas complejas reduciendo las dependencias o asociar comportamiento a objetos e invocar su ejecución.

Lógicamente el uso de patrones de diseño implica para el equipo de desarrollo un profundo conocimiento del contexto del problema, de los patrones aplicables y de las opciones de diseño disponibles.

3 UN ENFOQUE HÍBRIDO: XP EXTENDIDA

El desarrollo de software evoluciona constantemente integrando modelos de proceso y nuevas tecnologías en busca de optimizar el proceso de desarrollo o ampliarlo a nuevos horizontes para así dar respuesta a las exigencias de un mercado cada vez más competitivo.

En este marco, se propone adaptar la metodología XP para incluir en su ciclo de vida los patrones de diseño y la reutilización de soluciones (generadas en proyectos anteriores) mediante la modificación de sus fases. La inclusión de patrones de diseño permitirá entonces identificar, por ejemplo, posibles soluciones a cuestiones de arquitectura del sistema independientemente del dominio de la aplicación. En tanto que la reutilización de soluciones permitirá que una organización aplique el conocimiento (requerimientos de usuario, modelos de análisis, diseño de interfaces gráficas, diseños de arquitecturas, pruebas, etc.) obtenido en proyectos anteriores para encarar problemas de iguales o similares características con mayor velocidad y adaptar o incluso mejorar las soluciones ya desarrolladas a través de la reingeniería y la práctica de la reutilización.

En vista de alcanzar el objetivo planteado fue necesario estudiar y modificar los siguientes aspectos de XP:

- Reutilización de soluciones: el ciclo de vida de XP no contempla explícitamente la reutilización de software o la aplicación de patrones.
- Pruebas: en XP las pruebas tienen por objetivo validar componentes o incrementos.
- Documentación: si bien para el desarrollo y administración de tareas el modelo de historias es efectivo, éste no permite documentar el desarrollo completo ya que sólo registra el caso inicial (no documenta el caso final). Consecuentemente el trabajo realizado no será accesible para otros

desarrolladores que pudieran aprovechar las soluciones obtenidas (diseños, artefactos, pruebas, etc.).

3.1 *Reutilización*

La reutilización en XP Extendida (XPE) se sustenta en el concepto de solución, concepto que comprende los requerimientos de usuario, modelos de análisis, diseño de interfaces gráficas, diseños de arquitecturas, pruebas, etc. Al aplicar soluciones basadas en patrones de diseño o proyectos anteriores es posible capitalizar las semejanzas de familias de productos software, empleando arquitecturas reutilizables y componentes básicos comunes lo que agiliza las etapas iniciales del desarrollo (Sommerville, 2006). Evidentemente para lograr esto es preciso crear, gestionar y mantener un repositorio de soluciones bien documentadas. De este modo, en la etapa de diseño se podrán aplicar los patrones más convenientes o las soluciones desarrolladas en proyectos anteriores con mayor rapidez.

3.2 *Pruebas*

Las pruebas constituyen uno de los puntos clave en XP, éstas se especifican antes de codificar cada componente del sistema estableciendo así los criterios a cumplir. En XPE el concepto de prueba se extiende más allá de la validación constituyéndose en un indicador del grado de avance del proyecto. Así, el enfoque de XPE es un desarrollo dirigido por pruebas.
Un equipo selecciona una historia, la interpreta y diseña las pruebas correspondientes, las que están basadas en las características de la historia y los requerimientos que propone. Preparada la batería de pruebas y codificado el componente o incremento de software éste se somete a prueba. Aquí cabe aclarar que en primera instancia el componente de software no pasa ninguna de las pruebas y falla la batería en un 100%. El objetivo es que el componente o incremento se vaya desarrollando para superar una a una las pruebas de la batería de pruebas. Se considera que un componente no está completo hasta que supere el 100% de las pruebas. Nótese que la cantidad de pruebas superadas se convierte en un indicador del progreso del desarrollo. Al final del ciclo de desarrollo el componente estará terminado y completamente probado.

3.3 *Documentación*

XPE sugiere modificaciones a la documentación base de XP. Estas modificaciones, ideadas para registrar las pruebas realizadas y las soluciones elaboradas, son las que dan soporte a la filosofía de reutilización planteada en esta propuesta.
Básicamente, la documentación se sirve de 2 tipos de ficha: fichas de historia y fichas de grupo.
Al formato habitual de las fichas de historia se agrega un campo que permite identificar el grupo (N° de ficha de grupo) al que pertenecen. Los grupos, como se verá, se conforman con fichas de historia que comparten ciertos aspectos o requerimientos.
La ficha de grupo contiene referencias a un conjunto de historias. Este conjunto es determinado en el momento de la elaboración de las fichas en base a la relación o características comunes existentes entre ellas, por ejemplo, una historia para los ABMS de productos puede ser agrupada con la historia de ABMS de Clientes.
Las fichas de grupo se componen de las siguientes secciones:

- Cabecera (Fig. 3): contiene el nombre de la empresa y el proyecto al que pertenece la ficha, los N° de fichas de historia que agrupa, el equipo de trabajo y por último las fechas de inicio y fin de desarrollo. Además, incluye un campo de observaciones para indicar detalles si es necesario.
- Solución tentativa (Fig. 4): esta sección se completa con la solución propuesta para las historias que se indicaron en la sección anterior. La solución se expresa mediante un diagrama de clases y observaciones que expliquen en detalle la solución propuesta según lo expresado en la historias. En este punto los desarrolladores deben analizar patrones de diseño o soluciones anteriores aplicables al caso.
- Solución final (Fig. 5): es común que la solución inicial sufra cambios al implementarse, por ello, esta sección registra la solución final obtenida de modo que esté disponible para reutilizarse en proyectos futuros.

Como se mencionó anteriormente, las pruebas son un elemento fundamental en XPE ya que dirigen el desarrollo de software. Continuando con la filosofía de reutilización las pruebas unitarias, de integración y aceptación se documentan como se describe a continuación.

- Las pruebas unitarias diseñadas para cada historia comprueban las funcionalidades concretas de un subsistema. Por cada ficha de grupo se llevan a cabo una batería (o suite) de pruebas que se documentan como se observa en la Fig. 6.

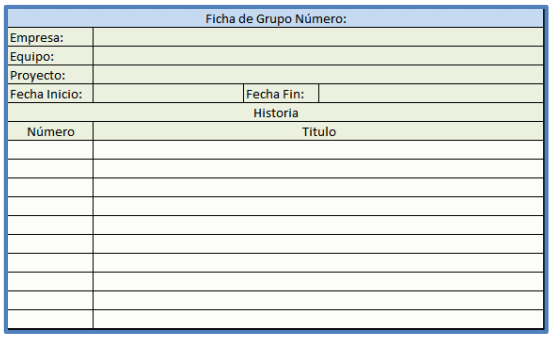


Figura 3. Cabecera de la ficha de grupo.

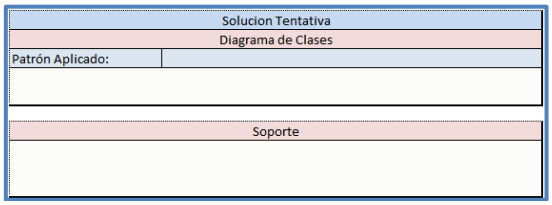


Figura 4. Sección de solución tentativa.

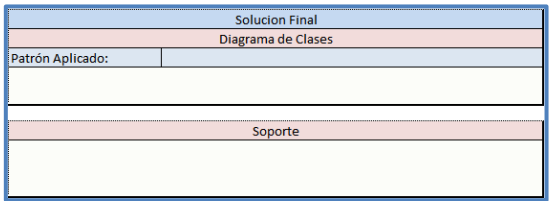


Figura 5. Sección de solución final.

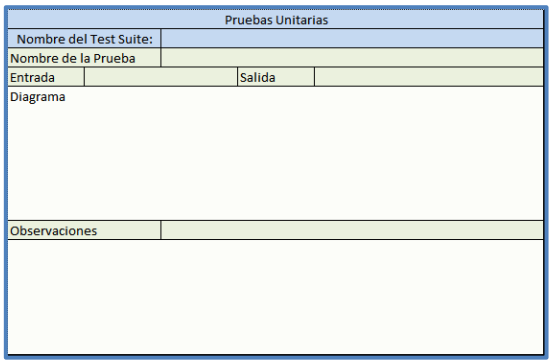


Figura 6. Documentación de pruebas unitarias.

Se asigna un nombre al test o suite que contendrá todas las pruebas, se indica el nombre para la prueba en cuestión, la entrada y salida correspondiente, un diagrama de flujo de la prueba y una sección de observaciones. Es necesario recordar que las pruebas se documentan de esta forma debido que dirigen el desarrollo, si no hay pruebas no se puede empezar a codificar.

- Las pruebas de integración se realizan al completar el desarrollo de los componentes especificados en las fichas de grupo. Estas pruebas tienen por objetivo comprobar la integración de los subsistemas de cada historia. La documentación se realiza como se observa en la Fig. 7.

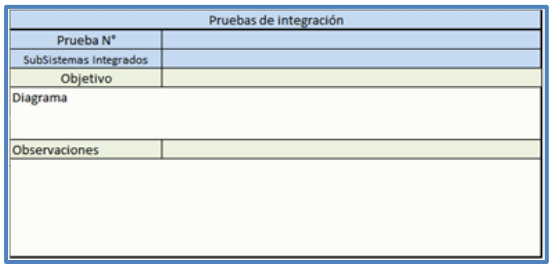


Figura 7. Documentación de pruebas unitarias.

De forma similar a las pruebas unitarias, la documentación incluye nombre de la prueba, objetivo y especificación de los subsistemas que se prueban y el diagrama correspondiente, además de las observaciones que se consideren necesarias.

- Las pruebas de aceptación ejecutadas por el usuario se documentan registrando las acciones realizadas con el software y la funcionalidad probada. Estas pruebas se llevan a cabo en conjunto con las pruebas de las otras fichas de grupo desarrolladas en un mismo incremento. La Fig. 8 muestra el registro de las pruebas de aceptación.

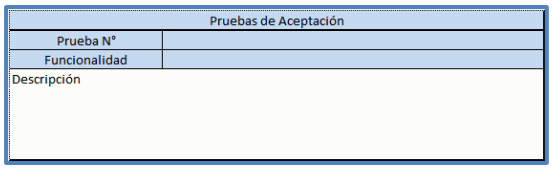


Figura 8. Documentación de pruebas unitarias.

3.4 *Fase de iteraciones en XPE*

XPE modifica la fase de iteraciones de la XP clásica enfocando el desarrollo a la dirección por pruebas y adicionando ciertas etapas destinadas al análisis y especificación de soluciones.
En la Fig. 9 se observa las etapas de la fase de iteraciones modificada.
Las iteraciones comprenden el desarrollo de un conjunto de fichas de grupos que constituirán el incremento planeado. Las fichas de una iteración determinada se seleccionan en la fase de planificación.
La primera tarea en el desarrollo de una ficha de grupo es el planteo de la solución inicial. Esta solución se especifica mediante un diagrama de clases que indica a grandes rasgos la solución propuesta para el problema a tratar. En este punto se debe analizar los patrones de diseño aplicables así como las soluciones utilizadas en otras fichas de grupo. En la especificación de la solución se realiza el diagrama de clases y las observaciones necesarias. Con la solución especificada se lleva a cabo el diseño y la correspondiente codificación de las pruebas que definirán la funcionalidad del componente a construir. Una vez que el

componente construido supera la batería de pruebas se considera el desarrollo completo y se continúa con la siguiente historia. El diseño y codificación de las pruebas de componentes se documenta en la sección de pruebas unitarias. Finalizada la prueba de todas las historias de la ficha de grupo se llevan a cabo las pruebas de integración, documentándose en la sección de pruebas de integración. Finalmente, se preparan las pruebas de aceptación con el cliente y si todo resulta correcto se documenta la solución final junto a las observaciones necesarias para cerrar el incremento. Completado el desarrollo de todo el sistema se realiza el despliegue correspondiente.

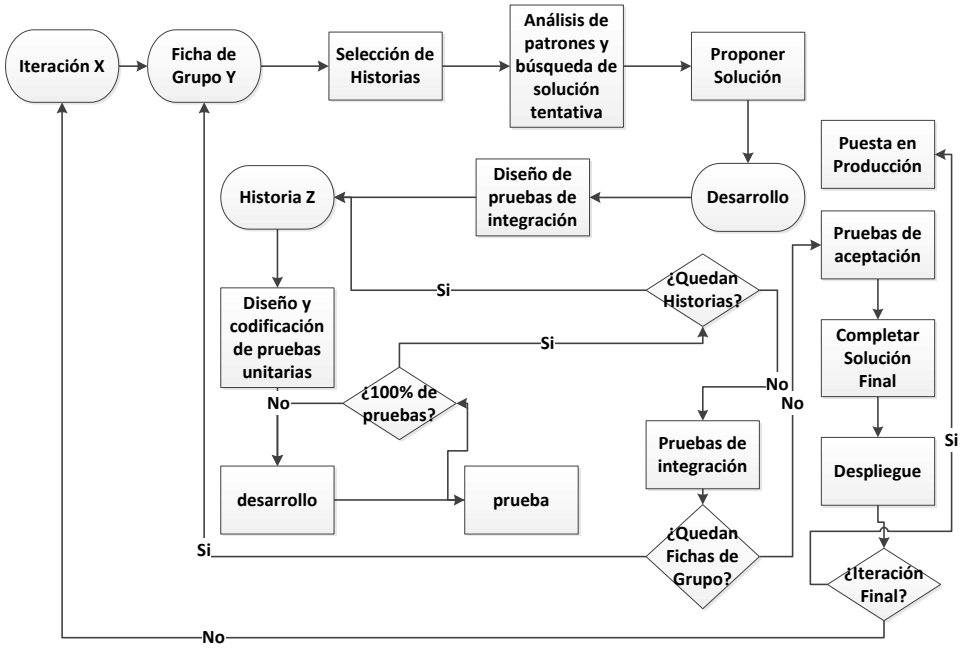


Figura 9. Etapas de fase de iteraciones

4 CASO DE ESTUDIO

En este apartado se muestra cómo se aplicó XPE al desarrollo del módulo de mantenimiento de un sistema Gestión de Mantenimiento Asistido por Computadora (Fig. 10) (García Garrido, 2003).

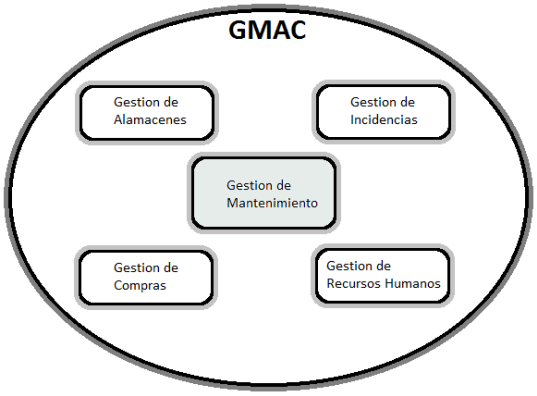


Figura 10. Módulos de un sistema GMAC.

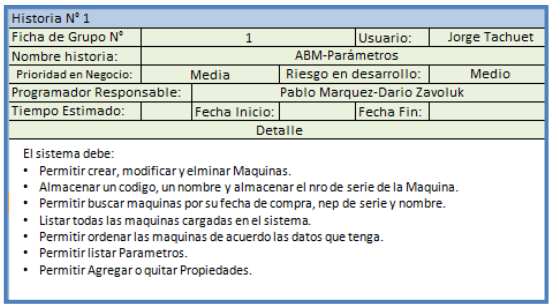
El proyecto comprendió el desarrollo de 3 subsistemas principales de este módulo:

- aplicación de escritorio: destinada a facilitar las tareas de reparación y mantenimiento de los activos al organizar la política de mantenimiento contemplando el ciclo de vida completo de las órdenes de trabajo (OT), pedidos de trabajo, planificación de OT semanales, carga horaria por empleado, materiales, herramientas, servicios de terceros, estadísticas y pedidos de materiales así como registro de las operaciones realizadas para control y auditorias.
- aplicación móvil: destinado a registrar las mediciones realizadas por distintos dispositivos (horas de trabajo, kilometraje, presión, niveles, caudales, temperatura) para su posterior almacenamiento en el sistema.
- aplicación web: destinada a generar pedidos de trabajo al área de mantenimiento (con acceso restringido al personal autorizado) y los informes correspondientes a pedidos de trabajo.

4.1 *Desarrollo del sistema*

A partir de la información recabada en la fase de exploración se identificaron 24 historias de usuario (Fig. 11) que, según su prioridad en el desarrollo, se agruparon en 16 fichas de grupo.

Para la realización del proyecto, el desarrollo de las fichas de grupo se organizó en 8 iteraciones como puede observarse en la tabla 1.



Historia N° 1				
Ficha de Grupo N°	1		Usuario:	Jorge Tachuet
Nombre historia:	ABM-Parámetros			
Prioridad en Negocio:	Media		Riesgo en desarrollo:	Medio
Programador Responsable:	Pablo Marquez-Dario Zavoluk			
Tiempo Estimado:		Fecha Inicio:		Fecha Fin:
Detalle				

El sistema debe:
- Permitir crear, modificar y elminar Maquinas.
- Almacenar un codigo, un nombre y almacenar el nro de serie de la Maquina.
- Permitir buscar maquinas por su fecha de compra, nep de serie y nombre.
- Listar todas las maquinas cargadas en el sistema.
- Permitir ordenar las maquinas de acuerdo las datos que tenga.
- Permitir listar Parametros.
- Permitir Agregar o quitar Propiedades.

Figura 11. Ficha de historia ABM Parámetros.

Tabla 1. Organización de historias, grupos e iteraciones.

#	Historia	Grupo	Iteración
1	ABM Parámetros	1	1
2	ABM Propiedades		
3	ABM Máquinas		
4	ABM Empleados	2	
5	ABM Herramientas	3	
6	ABM Grupos	4	2
7	ABM Usuarios		
8	ABM Permisos		
9	ABM Indicadores	5	
10	ABM Mediciones		
11	Graficadora	6	
12	ABM Tareas	7	3
13	ABM Uso		
14	Creación Pedidos de Trabajo	8	4
15	ABM Trabajo	9	5
16	Crear OT Correctiva		
17	Crear OT Preventiva	10	
18	Crear OT Predictiva	11	
19	Visualizador de OT	12	
20	Editor de OT	13	6
21	Impresora de Informes	14	
22	Lector de Mediciones	15	7
23	Generador/Lector de Fichero de Mediciones		
24	Login y Seguridad	16	8

En la primera iteración del desarrollo, tras analizar las alternativas de solución, se aplicó el patrón MVC (Model-View-Controller) para la construcción de la arquitectura del sistema, ya que éste permite separarlo en 3 capas especializadas: 1) la vista o capa de presentación que implementa las pantallas de usuario, 2) la capa de controlador que contiene la lógica destinada a manipular la información para adaptarla a los componentes gráficos de la capa de presentación y aplicar las reglas de negocio y 3) la capa de modelo que contiene la lógica para el acceso a datos y la definición de las entidades del sistema. Además, persiguiendo una mayor abstracción para el acceso a bases de datos y la aplicación de reglas de negocio, se implementó del patrón DAO (Data-Access-Object) y la inyección de dependencias. Teniendo en cuenta que algunos objetos pueden usar las mismas instancias en diferentes situaciones también se aplicó el patrón Singleton Así, definidos los componentes a construir se especificaron y documentaron (Fig. 12) las pruebas unitarias (por historia) y las pruebas de integración y aceptación (por grupo de historias) que permiten verificar la funcionalidad. Luego, se codificaron los componentes planeados (solución inicial) y su desarrollo no se consideró completo hasta superar la batería de pruebas correspondiente. Cuando la solución fue satisfactoria se registró la solución final (Fig. 13).

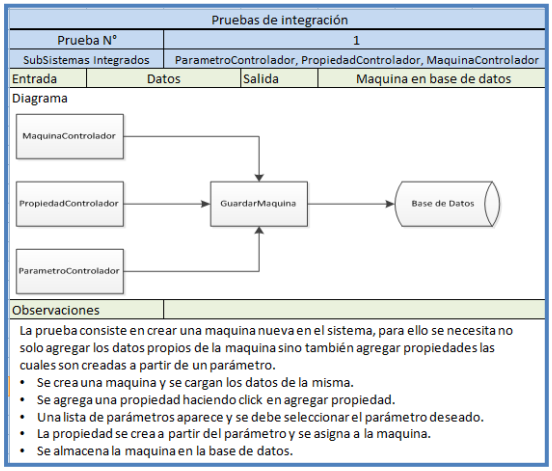


Figura 12. Documentación de pruebas de integración.

Esto se realizó por cada historia y grupo de historias hasta completar el desarrollo del sistema. La tabla 2 resume los patrones y/o reuso de soluciones aplicados en el desarrollo del proyecto. Cabe destacar que en cada iteración es posible, en caso de ser necesario, actualizar la planificación para acomodar cambios en los requisitos, en las tecnologías de implementación o en el tiempo y esfuerzo estimado para desarrollar las actividades.

5 CONCLUSIONES

En este artículo se presentó XPE un enfoque híbrido para el desarrollo de software que integra al ciclo de vida de XP el uso de patrones de diseño y la reutilización de soluciones. XPE modifica la fase de iteración de XP extendiendo el modelo de historias para documentar diseños, pruebas y soluciones; además incluye en el diseño el análisis de patrones y soluciones previas. Las pruebas también adquieren un rol esencial ya que

permiten dirigir el desarrollo y proporcionar un indicador del grado de avance del proyecto.
Para comprobar la aplicabilidad de la propuesta, se usó XPE en un caso de estudio. Los resultados obtenidos permitieron establecer que, si bien fue posible usar patrones e incluso reusar soluciones generadas en el proyecto (a falta de soluciones anteriores), la experiencia de los desarrolla-dores resulta un factor esencial para alcanzar el éxito.

Tabla 2. Resumen del desarrollo del proyecto.

Iteración	Grupo	Solución
1	1	MVC, DAO, Singleton, Inyección de dependencias
	2	MVC, DAO, Singleton, Inyección de dependencias
	3	MVC, DAO, Singleton, Inyección de dependencias
2	4	MVC, DAO, Singleton, Inyección de dependencias
	5	MVC, DAO, Singleton
	6	Strategy
3	7	MVC, DAO, Singleton, Inyección de dependencias
4	8	MVC, DAO, Singleton, Inyección de dependencias
5	9	MVC, DAO, Singleton
	10	Reutilización de solución de ficha de grupo N° 9
	11	Reutilización de solución de ficha de grupo N° 9
	12	Reutilización de solución de ficha de grupo N° 9
6	13	Reutilización de solución de ficha de grupo N° 9
	14	Stratetgy
7	15	MVC (simplificado)
8	16	Reutilización de solución de ficha de grupo N° 1

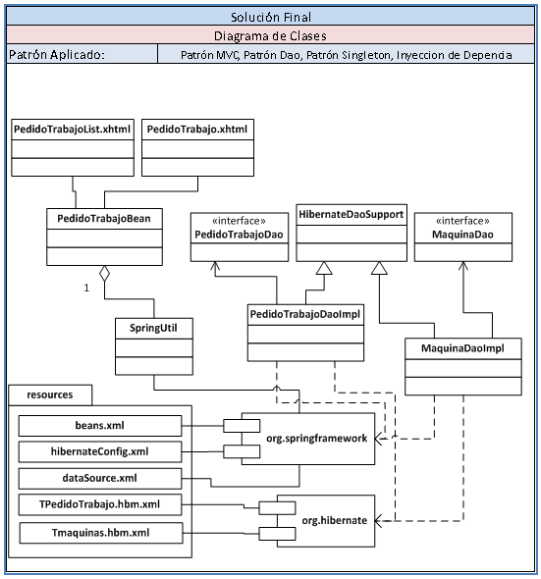


Figura 13. Documentación de la solución final.

Además, respecto al nuevo modelo de documentación se detectaron redundancias que sugieren debe sustentarse mediante un sistema de registro automatizado que permita un rápido acceso a las soluciones documentadas. Los resultados prometedores de esta experiencia permiten suponer que la propuesta de XPE tendrá futuro

6 REFERENCIAS

Gamma, E., Helm, R., Johnson, R. & J. Vlissides. *Patrones de diseño. Elementos de software orientado a objetos reutilizables*. Ed. Pearson Educación. ISBN: 978-84-7829-059-8. Madrid. 2003

García Garrido, S. *Organización y gestión del mantenimiento: manual práctico para la implantación de sistemas de gestión avanzados de mantenimiento industrial*. Ed. Díaz de Santos. ISBN: 9788479785482. Madrid. 2003

Hurtado, J. A. & C. Bastiarrica. Modelo de Procesos, Calidad y Mejoramiento: CMM, TSP, PSP, ISO, IEEE, SPICE, etc. *Proyecto SIMEP-SW*. Mayo 08 de 2005

Pfleeger, S. L. & J. M. Atlee. *Software Engineering. Theory and Practice*. 4° Edition. Ed. Pearson Higher Education. New Jersey. 2010

Pressman, R. S. *Ingeniería de Software. Un enfoque práctico*. 6° Edición. Ed. McGraw Hill. ISBN 970-10-5473-3. México. 2006

Sommerville, I. *Ingeniería de Software*. 7° Edición. Ed. Pearson Educación. ISBN 9788478290741. Madrid. 2005