Implementando scrum + rad para la gestión y desarrollo de proyectos de software en equipos de trabajo con personal limitado y eventual

Implementing scrum + rad for the management and development of software projects in teams with limited and temporary staff

JM.T.C. José Manuel Gómez Zea
Instituto Tecnológico de Villahermosa
Carretera Villahermosa-Frontera, km.3.5, Ciudad Industrial.
Villahermosa, Tabasco, México, c.p. 86010
Correo-e: jgomez.zea@hotmail.com

PALABRAS CLAVE:

Ciencias computacionales, Gestión de proyectos, Métodos Ágiles, Ingeniería de software

RESUMEN

Las metodologías de desarrollo ágil han transformado la manera de construir software; de métodos rígidos y lentos a procesos flexibles y eficientes. Rad y Scrum son marcos de trabajo ágil que proponen un conjunto de prácticas y roles para: elaborar software de calidad, obtener resultados anticipados, dar flexibilidad y adaptación, permitir el retorno de inversión, promover la productividad; entregar con frecuencia software que funcione, entre otros. Este artículo propone un conjunto de buenas prácticas (combinación de la metodología scrum y el desarrollo rápido de aplicaciones rad) para el análisis, diseño y desarrollo de proyectos de software de calidad en períodos menores a 60 días, aplicado en entornos con poco personal o personal eventual. Además, se plantean variables y elementos para su óptimo funcionamiento y se demuestran en el caso de estudio del departamento de desarrollo del Instituto Tecnológico de Villahermosa, en Tabasco, México

KEYWORDS:

Computer science, Project management, Agile methods, Software engineering

ABSTRACT

The agile development methodologies have transformed the way we build software; from rigid and slow methods to flexible and efficient processes. Rad and Scrum are agile frameworks that propose a set of practices and roles for develop quality software, obtain anticipated results, provide flexibility and adaptation, allow the return on investment, promote productivity, and deliver working software frequently, among others. This paper proposes a set of best practices (combination of the scrum methodology and the rapid application development rad) for the analysis, design and development of quality software projects in periods of less than 60 days, applied in environments with limited staff or staff eventual. Variables and elements are analyzed for optimum performance in the case study of the development department of the Villahermosa Institute of Technology, in Tabasco, México.

Recibido: 31 de julio del 2015 • Aceptado: 11 de noviembre del 2015 • Publicado en línea: 7 de octubre 2016

1 INTRODUCCIÓN

Desde 1980 cuando James Maslow presentó el RAD (acrónimo en inglés de Rapid Application Development), desarrollo rápido de aplicaciones que permite construir sistemas utilizables en poco tiempo, normalmente en 30 o 90 días y consolidado en 1997 por Walter Maner, fueron apareciendo nuevos procesos de desarrollo de software que asociados al concepto "LEAN" implementado por Toyota en los procesos de producción masiva, se busca remover todos los excesos de programación, para poder enfocarse en llevar valor al cliente lo más rápido posible, sin dejar a un lado la calidad [1].

Basado en estos métodos, en el año 2001 un grupo de ingenieros informáticos de Utah, Estados Unidos, escribió el manifiesto Ágil para la gestión de proyectos de software [2].

El Manifiesto Ágil centra su atención en: Individuos e interacciones sobre procesos y herramientas, software funcionando sobre documentación extensiva, colaboración con el cliente sobre negociación contractual [1].

Xp, Scrum, Rup, Fdd, Cristal clear, Lean, entre otros, son algunos de los métodos ágiles [2].

Normalmente para implementar cualquiera de estos marcos de gestión y desarrollo en departamentos de informática o sistemas, se requiere de un equipo humano experimentado y capacitado que integre los supuestos, técnicas y métodos establecidos para la elaboración de proyectos de software. Bajo este contexto es importante mencionar que existen instituciones y empresas que cuentan con áreas de informática, sistemas o gestión de ti, que dentro de sus funciones diarias deben diseñar, desarrollar e implementar proyectos de software; desde aplicaciones de escritorio hasta sistemas web; estos departamentos tienen asignado muy poco personal de base los cuales hacen varias funciones y no les permite en muchas ocasiones elaborar y desplegar sistemas de información de calidad en tiempo y forma.

En este contexto, estas empresas optan por reclutar practicantes o residentes con el fin de cubrir actividades para proyectos de mayor extensión.

Este articulo ofrece un conjunto de elementos, herramientas y actividades obtenidas de la combinación de las metodologías rud y scrum, con el propósito de promover un marco de trabajo adaptativo para los departamentos de informática donde no se tiene personal suficiente o se cuenta con personal eventual por ejemplo: practicantes y residentes para la elaboración de productos de software de calidad y entregables en tiempo y forma.

La metodología empleada en esta investigación fue un estudio de caso complementada marginalmente con la investigación experimental, analizando variables de: Técnicas y estándares, Escenarios de trabajo, Horarios y cantidad de integrantes, Herramientas, Tamaño y tiempo de proyectos, Artefactos, Pruebas y Documentación.

El artículo está organizado de la siguiente manera, en la sección 2 se introduce a los conceptos y características de las metodologías scrum y rad obtenidas de sus manifiestos correspondientes, en la sección 3 se detallan las buenas prácticas de cada elemento que debe ser considerados para implementar las metodologías en un departamento de desarrollo, en la sección 4 se abordan los pasos para integrar todos los elementos descritos en la sección 3, en la sección 5 se expone un caso de estudio donde se implementó la metodología y los resultados se describen en la sección 6. Finalmente se presenta las conclusiones y referencias bibliográficas.

2 METODOLOGÍAS ÁGILES

2.1. RAPID APPLICATION DEVELOPMENT

Rad es un proceso de desarrollo de software que comprende el desarrollo interactivo, la construcción de prototipos y el uso de utilidades case [3].

El enfoque RAD divide el proceso en cuatro fases distintas:

1.Requisitos de planificación de fase: los usuarios, administradores y miembros del personal de TI discuten y están de acuerdo en las necesidades del negocio, el alcance del proyecto, restricciones y requisitos del sistema.

2.Fase de diseño del usuario: durante esta fase, los usuarios interactúan con los analistas de sistemas y desarrollan modelos y prototipos que representan todos los sistemas de procesos, insumos y salidas.

3.Fase de construcción: en esta fase se realiza la programación y el desarrollo de la aplicación, codificación, integración y pruebas del sistema

4.Fase de corte y cambio: se asemeja a las tareas finales en la fase de implementación, incluyendo la conversión de datos, pruebas, transición al nuevo sistema, y la capacitación de los usuarios.

Como puede verse en la Fig. 1, los enfoques de la metodología Rad parten de la planeación de re-

querimientos, diseño de usuario, construcción hasta la fase de corte y cambio.

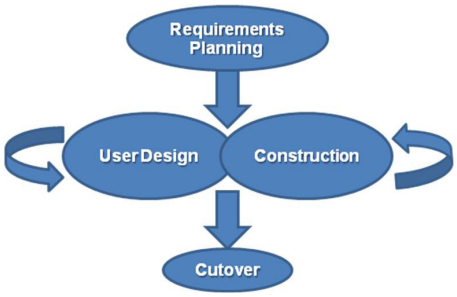

Fig. 1.- Fases del enfoque Rad. Autor: James Martín.

A.Características de Rad

a)Equipos híbridos: Los desarrolladores deben ser analistas, diseñadores y programadores en uno.

b)Herramientas especializadas: Desarrollo visual, Herramientas colaborativas, Interfaces estándares, Control de versiones, Calendario grupal y Componentes reusables.

c)Timeboxing: Las funciones secundarias son eliminadas. Técnica que consiste en fijar el tiempo máximo para conseguir unos objetivos, tomar una decisión o realizar unas tareas.

d) Prototipos iterativos: Reunión JAD, se reúnen los usuarios y los desarrolladores. Lluvia de ideas para obtener un borrador inicial de los requisitos.

e) El facilitador: Tiene clara las metas, prepara la agenda de asuntos y escribe un reporte final [3].

2.2.SCRUM

Es un marco de trabajo para la gestión y desarrollo de software, que se basa a la adaptación continúa a las circunstancias de la evolución de un proyecto [4].

A.Manifiesto Ágil

1.Nuestra principal prioridad es satisfacer al cliente a través de la entrega temprana y continua de software de valor.

2.Son bienvenidos los requisitos cambiantes, incluso si llegan tarde al desarrollo. Los procesos ágiles se doblegan al cambio como ventaja competitiva para el cliente.

3.Entregar con frecuencia software que funcione, en periodos de un par de semanas hasta un par de meses, con preferencia en los periodos breves.

4.Las personas del negocio y los desarrolladores deben trabajar juntos de forma cotidiana a través del proyecto

5.Construcción de proyectos en torno a individuos motivados, dándoles la oportunidad y el respaldo que necesitan y procurándoles confianza para que realicen la tarea.

6.La forma más eficiente y efectiva de comunicar información de ida y vuelta dentro de un equipo de desarrollo es mediante la conversación cara a cara.

7.El software que funciona es la principal medida del progreso

8.Los procesos ágiles promueven el desarrollo sostenido.

9.La atención continua a la excelencia técnica enaltece la agilidad.

10.La simplicidad como arte de maximizar la cantidad de trabajo que no se hace, es esencial.

11.Las mejores arquitecturas, requisitos y diseños emergen de equipos que se auto-organizan. En intervalos regulares, el equipo reflexiona sobre la forma de ser más efectivo y ajusta su conducta en consecuencia [4].

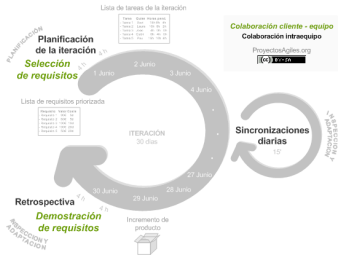

Fig. 2.- Proceso Scrum. Autor: proyectosagiles.org

B.Características de Scrum

El proceso parte de la lista de objetivos/requisitos priorizada del producto, que actúa como plan del proyecto. En esta lista el cliente prioriza los objetivos balanceando el valor que le aportan respecto a su coste y quedan repartidos en iteraciones y entregas. De manera regular el cliente puede maximizar la utilidad de lo que se desarrolla y el retorno de inversión mediante la replanificación de objetivos del producto, que realiza durante la iteración con vista a las siguientes iteraciones [5].

Las actividades que se llevan a cabo en Scrum son las siguientes:

1.Planificación de la iteración (Sprint Planning). El cliente presenta al equipo la lista de requisitos priorizada del producto o proyecto. El equipo examina la lista, pregunta al cliente las dudas que le surgen, añade más condiciones de satisfacción y selecciona los objetivos/ requisitos más prioritarios que se compromete a completar en la iteración, de manera que puedan ser entregados si el cliente lo solicita. Se realiza en un Timebox de cómo máximo 4 horas.

2.Ejecución de la iteración (sprint). Un proyecto se ejecuta en bloques temporales cortos y fijos (iteraciones de un mes natural y hasta de dos semanas).

3.Reunión diaria de sincronización del equipo (Scrum Daily Meeting). Cada miembro del equipo inspecciona el trabajo que el resto está realizando

4.Demostración de requisitos completados (Sprint Review). Reunión informal donde el equipo presenta al cliente los requisitos completados en la interacción.

5.Retrospectiva (Sprint Retrospective). El equipo analiza cómo ha sido su manera de trabajar durante la iteración, por qué está consiguiendo o no los objetivos a que se comprometió al inicio de la iteración y por qué el incremento de producto que acaba de demostrar al cliente era lo que él esperaba o no [5].

Los roles que se identifican en Scrum son:

1.Product Owner.- Cliente interno o externo a la organización.

2.Scrum Master.- Facilitador o líder del equipo.

3.Team.- Equipo de desarrolladores, es posible hacer scrum con 3 o hasta más de 200 personas.

Las herramientas que se identifican en scrum son:

1.Product Backlog. La lista de objetivos/requisitos.

2.Sprint Backlog. Lista de tareas que el equipo elabora en la reunión de planificación de la iteración (Sprint planning) [5].

2.3 XP

La metodología XP se considera una metodología leve de desarrollo de software. Esta es clasificada como un sistema de prácticas que la comunidad de desarrolladores de software viene evolucionando para resolver los problemas de entrega de software de calidad rápidamente, y poder alcanzar las necesidades de negocio que siempre cambian. Esta surgió a partir de ideas de Kent Beck y Ward Cunningham y que fue utilizada por primera vez en un proyecto piloto en marzo de 1996, del cual el propio Beck formaba parte. Lo del Extreme del nombre de la metodología se debe al hecho de que esta emplea al extremo, las buenas prácticas de la ingeniería de software [6].

La XP no se aplica a todos los tipos de proyectos, siendo más apropiada para los proyectos con equipos pequeños y medianos, de dos a doce personas. Sin embargo, algunos defienden su uso en grandes proyectos, ya que al dividirlos en subproyectos independientes. Los proyectos largos deben ser partidos en una secuencia de mini proyectos de autocontenidos, con una duración de una a tres semanas [6].

Según Teles, la XP es un proceso de desarrollo de software apropiado para los siguientes proyectos:

a)Con requisitos que son vagos y que cambian con frecuencia

b)Desarrollo de sistemas orientados a objetos

c)Equipos pequeños

d)Desarrollo incremental.

Según Beck la XP está definida por medio de valores, principios y prácticas. Los valores describen los objetivos de largo plazo y definen criterios para obtener el éxito [7]. Los cinco valores son:

1.Simplicidad: Se simplifica el diseño para agilizar el desarrollo y facilitar el mantenimiento.

2.Comunicación: Para los programadores el código comunica mejor cuanto más simple sea. Si el código es complejo hay que esforzarse para hacerlo inteligible. El código autodocumentado es más fiable que los comentarios ya que éstos últimos pronto quedan desfasados con el código a medida que es modificado

3.Retroalimentación: Al estar el cliente integrado en el proyecto, su opinión sobre el estado del proyecto se conoce en tiempo real.

4.Coraje: Una de ellas es siempre diseñar y programar para hoy y no para mañana

5.Respeto: Los miembros respetan su trabajo porque siempre están luchando por la alta calidad en el producto y buscando el diseño óptimo o más eficiente para la solución a través de la refactorización del código.

Características de XP:

1.Desarrollo iterativo e incremental, pequeñas mejoras, unas tras otras.

2.Pruebas unitarias continuas, frecuentemente repetidas y automatizadas,

3.Programación en parejas, se recomienda que las tareas de desarrollo se lleven a cabo por dos personas en un mismo puesto.

4.Frecuente integración del equipo de programación con el cliente o usuario. Se recomienda que un representante del cliente trabaje junto al equipo de desarrollo.

5.Refactorización del código, es decir, reescribir ciertas partes del código para aumentar su legibilidad y mantenibilidad pero sin modificar su comportamiento.

6.Propiedad del código compartida, en vez de dividir la responsabilidad en el desarrollo de cada módulo en grupos de trabajo distintos, este método promueve el que todo el personal pueda corregir y extender cualquier parte del proyecto.

7.Simplicidad en el código, es más sencillo hacer algo simple y tener un poco de trabajo extra para cambiarlo si se requiere, que realizar algo complicado y quizás nunca utilizarlo [7].

Roles:

1.Cliente: Escribe las historias de usuario y las pruebas funcionales para validar su implementación

2.Programador: Escribe las pruebas unitarias y produce el código del sistema

3.Tester: Ayuda al cliente a escribir las pruebas funcionales. Ejecuta pruebas regularmente, difunde los resultados en el equipo y es responsable de las herramientas de soporte para pruebas.

4.Tracker: Es el encargado de seguimiento. Proporciona realimentación al equipo. Debe verificar el grado de acierto entre las estimaciones realizadas y el tiempo real dedicado, comunicando los resultados para mejorar futuras estimaciones

5.Entrenador: Responsable del proceso global. Guía a los miembros del equipo para seguir el proceso correctamente.

6.Consultor: Es un miembro externo del equipo con un conocimiento específico en algún tema necesario para el proyecto

7.Gestor: Es el dueño de la tienda y el vínculo entre clientes y programadores [7].

2.4 CRYSTAL CLEAR

Crystal Clear no es una metodología en si misma sino una familia de metodologías con un "código genético" común. La idea es poder armar distintas metodologías para distintos tipos de proyectos. Cada proyecto y organización usará este código genético para generar su propia metodología [8].

El nombre Crystal deriva de la caracterización de los proyectos según 2 dimensiones, tamaño y complejidad (como en los minerales, color y dureza).

Por ejemplo, Clear es para equipos de hasta 8 personas, Amarillo para equipos entre 10 – 20 miembros, Naranja para 20 a 50 personas, rojo es para 50 a 100 personas [8].

El código genético consiste en:

1.Modelo de juegos cooperativos: Este modelo ve al desarrollo de software como una serie de partidos que consisten en inventar y comunicar. Cada partido es diferente y tiene como objetivo entregar software y prepararse para el siguiente juego. Esto permite al equipo trabajar concentrado y en forma efectiva con un objetivo claro cada vez.

2.Seleccionar prioridades: conjunto de prioridades y principios que sirven de guía para la toma de decisiones:

a)Eficiencia en el desarrollo, para hacer que los proyectos sean económicamente rentables.

b)Seguridad en lo que se entrega.

c)Habitabilidad, hacer que todos los miembros del equipo adopten y sigan las convenciones de trabajo establecidas por el equipo mismo.

3.Propiedades:

a)Frecuencia en las entregas: entregar al usuario funcionalidad "usable" con una frecuencia de entre 2 semanas y no más de un mes.

b)Comunicación: Promueve prácticas como el uso de pizarrones, pizarras y espacios destinados a que todos (miembros del equipo y visitas) puedan ver claramente el progreso del trabajo.

c)Crecimiento reflexivo: es necesario que el equipo lleve a cabo reuniones periódicas de reflexión que permitan crecer y hacernos más eficientes.

Estas tres propiedades son "obligatorias" para Crystal Clear, las siguientes pueden agregarse en la medida de las necesidades de cada grupo y proyecto.

d)Seguridad personal: lograr que cada miembro del team pueda sentirse cómodo con el trabajo y el entorno.

e)Concentración: las entregas frecuentes permiten que cada desarrollador puede enfocar de a un problema por vez evitando dispersiones.

f)Fácil acceso a usuarios clave: tratar de hacer que el usuario sea una parte más del equipo es fundamental para ir depurando errores de manera temprana.

g)Entorno técnico con: I - Testing automatizado (incorporación, por ejemplo, de UnitTest) - II. Integración frecuente (uso de herramientas específicas como Cruise Control).

4.Principios: El grado de detalle necesario en documentar requerimientos, diseño, planeamiento, etc, varía según el proyecto. Es imposible eliminar toda documentación pero puede ser reducida logrando un modo de comunicación más accesible, informal y precisa que pueda ser accedido por todos los miembros del equipo.

El equipo ajusta constantemente su forma de trabajo para lograr que cada personalidad encaje con los otros miembros, con el entorno y las particularidades de cada asignación

5.Estrategias: Ni las estrategias ni las técnicas son mandatarias para Crystal Clear. Pero es bueno tener en cuenta alguna de ellas al momento de empezar a trabajar.

Tres de las estrategias que están más relacionadas son las de apuntar a tener "Victorias Tempranas", arrancar el desarrollo de lo que se denomina un "Esqueleto que Camine" y pensar siempre en hacer "Rearquitectura Incremental" van de la mano.

El poder arrancar el proceso a partir de un esqueleto sobre el cual se irá agregando funcionalidad en cada una de las entregas ayuda a que se vean los avances desde el comienzo (aunque sea una simple pantalla de ABM que se conecta con la base de datos y muestra un solo dato). A medida que se avanza en el proceso, la rearquitectura permitirá ir agregando más "cuerpo" al esqueleto inicial.

6.Técnicas: Igual que con las estrategias, hay una lista de técnicas propuestas por Crystal Clear, de las cuales se pueden ir tomando las más convenientes según el momento en que se encuentra el proceso de desarrollo del proyecto.

Las reuniones diarias (introducidas por la metodología Scrum) acompañan el seguimiento y mantienen el foco en el próximo paso a seguir, y también permiten la discusión productiva de líneas a seguir.

Las reuniones de reflexión periódicas son fundamentales para que los miembros del equipo se expresen abiertamente, para revisar el trabajo hecho y evaluar qué cosas dan resultado y cuáles no o de empezar a trabajar.

Todo esto permite ir armando una metodología de trabajo que se adecue al equipo, el proyecto y los tiempos que se manejen [8].

3 MODELO PROPUESTO PARA LA GESTIÓN Y DESARROLLO ÁGIL

Para la orquestación de la metodología se contemplaron los elementos presentados en la Fig. 3.

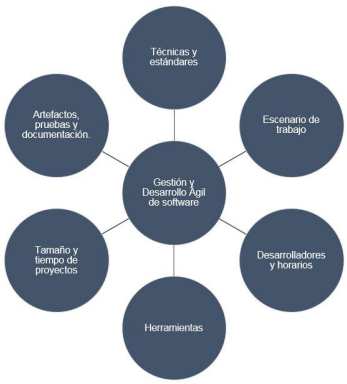

Fig. 3. Elementos para integración de la metodología ágil. Creada en la investigación.

3.1 TÉCNICAS Y ESTANDARES DE PROGRAMACIÓN

Con el fin de estandarizar y contemplar tiempos cortos para el desarrollo, se debe integrar un conjunto de módulos, librerías y técnicas de programación, que en conjunto con las herramientas permitan agilizar el proceso de armado y despliegue de los sistemas de información.

a)Específicamente podrá integrar módulos para generar modelos, controladores, vistas, plantillas, reportes y estadísticas.

b)Se deberá mantener un esqueleto o plantilla de diseño, la cual se utilizará para la mayoría de los sistemas de información, homologando los mismos.

c)Se sugiere a falta de los módulos y librerías, implementar un Framework (marco de trabajo) robusto pero con una curva de aprendizaje corta, dependiendo de la plataforma y lenguaje a utilizar.

d)Se sugiere la programación en pares, para los casos en que algún participante sea de nuevo ingreso o los módulos tengan mayor complejidad.

3.2 ESCENARIO DE TRABAJO

a)Se debe establecer un lugar específico para el grupo de desarrolladores, un espacio para al menos 4 y no más de 10 personas.

b)Se debe colocar una pizarra en la sala para establecer calendarios y metas., así como el análisis iterativo.

c)Debe haber mesas de trabajo en posición a la dirección de la pizarra (no obligatoriamente) con el fin de centrar la atención en la etapa de análisis, reuniones de trabajo continuo y explicación de temas sobre los proyectos.

d)De preferencia es conveniente tener un proyector para la etapa de capacitación y análisis de los proyectos.

3.3 DESARROLLADORES Y HORARIOS DE TRABAJO

a)Se debe reclutar practicantes específicamente de carreras afines a la programación, con o sin conocimiento de las tecnologías utilizadas.

b)Se debe reclutar residentes específicamente de carreras afines a la programación, con o sin conocimiento de las tecnologías utilizadas.

c)El horario de actividad debe ser de al menos 4 horas diarias, 8 horas son óptimas.

d)Los horarios no deben ser quebrados sino continuos.

e)Se debe considerar que las prácticas se realizan antes que las residencias, por lo que el reclutar practicantes trae como beneficio poder mantenerlos más tiempo considerando en un futuro su residencia en el mismo lugar.

3.4 HERRAMIENTAS

a)Deberá definirse un IDE para todos los desarrolladores, aunque no se niega el uso de alguna otra herramienta que los desarrolladores deseen, siempre y cuando mejore el proceso en el que se utiliza.

b)Deberá integrar una herramienta de Modelado de Datos, que permita la iteración con la base de datos física de tal manera que sea rápida la exportación o importación de los modelos.

c)Se sugiere utilizar un Sistema de Control de Versiones, que permita la gestión de cambios. Es importante mencionar que para evitar problemas con el control de versiones se tengan equipos de cómputo de mesa exclusivos para el desarrollo y evitar equipos móviles propiedad de los desarrolladores.

d)Implementar un Sistema de Gestión de Proyectos con características de la metodología ágil, donde se registren participantes y proyectos. El equipo debe subir archivos y avances del proyecto.

e)Se sugiere mantener un servidor (o pc) de pruebas para los proyectos en desarrollo.

3.5 TAMAÑO Y TIEMPO DE PROYECTOS

Los sistemas de información serán divididos entre transaccionales, de gestión y toma de decisiones, de tipo standalone o multiusuarios; de escritorio o web.

Significativamente para determinar el tiempo de finalización, deberá tomarse en cuenta el número y complejidad de módulos, el número de programadores, sus habilidades y experiencias anteriores.

Se recomienda que el tiempo sea establecido por todos los del equipo en consenso de la siguiente manera:

a)El líder de proyecto (Scrum master), junto al equipo (Team) y él ó los interesados (stakeholder) determinan los módulos (product backlog) que contendrá el sistema.

b)El Scrum Master, presenta los módulos uno a uno y cada integrante del equipo propone (con la técnica de planning póker) el tiempo aproximado para terminarlo. Este valor se obtiene de la media de las propuestas seleccionadas y se asigna el tiempo.

c)El líder de proyecto da la palabra a los integrantes más experimentados o a quienes hayan realizado esos módulos en sistemas anteriores.

d)El tiempo también estará determinado por el número de módulos que contendrá el sistema y su complejidad.

e)Generalmente se especificarán tiempos por horas para cada módulo.

Para solicitudes de nuevos proyectos, concretamente deberá tomarse en cuenta el wips (work in progress), la capacidad que tiene el personal para trabajar simultáneamente.

3.6 ARTEFACTOS, PRUEBAS Y DOCUMENTACIÓN

Se sugiere elaborar un formato de requisitos con al menos los siguientes campos:

1.Nombre del proyecto.
2.Departamento o área solicitante.
3.Nombre (s) de (los) solicitante (s).
4.Asignado a.
5.Fecha de levantamiento.
6.Fecha probable de terminación del proyecto.
7.Tiempo Estimado del requerimiento.

8.Características del requerimiento.

9.Requisitos funcionales y Requisitos no funcionales.

10.Notas y base de conocimientos.

11.Firmas del solicitante y de autorización

Por cada proyecto es necesario identificar los actores y sus roles en el sistema y con ello realizar los casos de uso [9]. Además se recomienda mantener impreso y en digital el modelo lógico de la base de datos y el diccionario de datos, el cual es entregado a cada participante del proyecto.

Se recomienda, realizar pruebas funcionales y/o unitarias para garantizar el funcionamiento correcto de los módulos.

La Fig. 4 ejemplifica un formato de requisitos, que contiene los elementos anteriormente mencionados y entre sus particularidades incluye el campo notas y base de conocimientos.

INSTITUTO TECNOLÓGICO DE VILLAHERMOSA

REQUERIMIENTOS DE SISTEMAS DE INFORMACIÓN DEPARTAMENTO DE CENTRO DE COMPUTO ÁREA DE DESARROLLO DE SOFTWARE						
Nombre del proyecto:	Aplicación móvil para la visualización de noticias, convocatorias y avisos.					
Departamento:	**Comunicación y difusión**		Solicitante:	**Lic. Beatriz Cañas**		
Asignado a:	**MTC. José Manuel Gómez Zea**					
ID: **ITVH0020**	Fecha de levantamiento	**10/08/2015**	Fecha de Proyecto:	**28/08/2015**	Tiempo estimado	**26 horas (3.5 días)**
Requisitos Funcionales	**Inicio de sesión de los usuarios** 1. La aplicación debe solicitar un nombre usuario (que será el correo electrónico registrado del usuario) y una contraseña (debe aparecer oculto al escribir la clave); y aparece vínculo para registrarse en caso de no tener una cuenta. 2. En caso de escribir mal un dato, se enviara mensaje de error. 3. En caso de acceder, el usuario podrá visualizar la lista de noticias, una pequeña imagen y el resumen. 4. Además el usuario podrá elegir de un menú, cambiarse a visualizar convocatorias, noticias y avisos. 5. Además el usuario podrá ver un menú de opciones donde presentará el cerrar sesión en caso que lo desee.					
Notas y base de conocimiento	• Diseñar la pantalla de inicio de sesión (8 hrs) • Realizar el método para enviar el nombre de usuario y contraseña (javascript encriptado) (16 hrs.) • El método debe recibir los siguientes datos, estatus de inicio (entro o no entro), nombre de usuario y correo (estas se considerarían variables de sesión) • Pruebas y correcciones (2 hrs.)					
Requisitos No Funcionales	1. El mensaje de respuesta debe ser lo más rápido posible 2. El servidor de servicios web y base de datos debe estar activo					
Notas y base de conocimiento	• El retorno de datos de los métodos deben ser xml • Checar que el servicio web login funcione correctamente con software alterno SOAPUI o JSON					

JEFE DE DEPARTAMENTO

NOMBRE Y FIRMA DEL SOLICITANTE

Fig. 4. Formato de requisitos. Creada en la investigación.

4 INTEGRACIÓN DE LOS ELEMENTOS

a)La elección y elaboración del método de programación, estandarización o marco de trabajo y/o aprendizaje del mismo se debe contemplar de 3 a 5 meses.

b)La integración y configuración de las herramientas, control de versiones y servidor de pruebas se debe contemplar a 1 mes.

c)El número de personal practicantes y residentes debe ser al principio de 4 para elegir de entre ellos a un líder de proyectos (scrum master), considerando que no se cuenta con un desarrollador de base.

d)Inicialmente deberá capacitarse al personal en un periodo de 1 mes, considerando que ya se han probado y ejecutado los módulos o códigos establecidos al principio. En este sentido se recomienda tener tutoriales o manuales paso a paso para que los desarrolladores realicen capacitación autodidacta.

e)En el transcurso de la capacitación (en caso de no haber una solicitud de desarrollo de un sistema), realizar la simulación del diseño de un prototipo funcional, en el cual se realicen las actividades siguientes: reunión de planeación (spring planning) y listado de requisitos (spring backlog), establecer tiempos de desarrollo de módulos utilizando el Planning Póker, asignar los módulos a cada desarrollador, reunir al equipo al menos cada dos días por 15 minutos para verificar el avance (spring daily meeting), revisar los pendientes y ayudar a quienes tienen dudas, reunirse los viernes o fechas establecidas para verificar los módulos terminados o el producto terminado (spring review). Al finalizar un producto o un módulo analizar los obstáculos que tuvieron y proponer mejoras (spring restrospective).

f)Contemplar en cada reunión la técnica TIMEBOX, estableciendo el tiempo de reunión y evitando salirse de la plática o discusión acordada.

g)El inicio de un proyecto, debe partir de la reunión de planeación (spring planning) y participan el usuario o usuarios solicitantes junto al equipo de desarrollo, continuar con la identificación de módulos (llenar el formato de requisitos o historias de usuarios), determinar tiempos y asignar módulos a los programadores, a continuación listar los roles del sistema y elaborar los casos de uso; posteriormente elaborar el modelado lógico y físico, imprimirlo y proporcionarlo al equipo de desarrollo.

h)Cada nuevo proyecto debe registrarse en el sistema de gestión de proyectos y los involucrados en su desarrollo deberán subir los artefactos que se vayan elaborando, incluyendo la lista de requisitos, los casos de uso, los roles del sistema. el modelo lógico, el modelo físico, un diccionario de datos, los módulos de programación y un manual de usuario (este último solo en caso de ser necesario)

i)En el transcurso del desarrollo deberán habilitarse pruebas de funcionalidad o unitarias, para cada módulo nuevo y registrarse en un formato que debe ser anexado en el Sistema de Gestión del Proyecto.

j)Al finalizar un proyecto se debe mejorar la metodología, buscando optimizar los procesos de desarrollo, así como las herramientas utilizadas.

k)Es importante cambiar de líder de proyecto por cada uno o dos proyectos con la finalidad de dividir equipos cuando se tenga más de un proyecto en elaboración.

5 DEMOSTRACIÓN DEL MÉTODO

La implementación se realizó en el departamento de centro de cómputo del Instituto Tecnológico de Villahermosa, en el estado de tabasco, México. En este período se contaba con un jefe de departamento y un apoyo de soporte técnico. Dentro de sus funciones están establecidas actividades de: Administración de servicios de red, diseño y administración de sistemas de información, soporte técnico y de mantenimiento a los equipos tecnológicos, administración del centro de datos, entre otros.

Las solicitudes de nuevos sistemas de información fueron en aumento de 2 a 3 sistemas por semestre, determinado por los eventos académicos, sociales y culturales de la institución. Los sistemas solicitados requieren al menos procesamiento de información a una base de datos, envío de emails, administración de usuarios, difusión de registros del evento, reportes estadísticos, impresión y funcionalidad para su acceso desde el internet.

5.1 DISEÑO EXPERIMENTAL

La tabla 1 muestra los métodos y resultados obtenidos en el escenario inicial a través de las variables analizadas, obtenidos en un periodo de 8 meses.

La tabla 2 muestra los métodos y resultados obtenidos en el segundo escenario a través de las variables analizadas, obtenidos en un período de 6 meses.

La tabla 3 muestra los métodos y resultados obtenidos en el tercer escenario a través de las variables analizadas, obtenidos en un periodo de 6 meses.

TABLA I ANALISIS DE VARIABLES EN EL ESCENARIO INICIAL

	ESCENARIO INICIAL	RESULTADOS OBSERVADOS
Tipo de codificación	1.-Utilización de código misceláneo. 2.- Creación de nuevos diseños de interfaz de usuario por cada sistema. 3.- Programación individual. 4.- Se realizan funciones nuevas con códigos encontrados en la web	-Constantes errores en los sistemas por el uso de múltiples técnicas de programación. -Proyectos fuera de tiempos. -Resultados no deseados por implementar código sin fundamento.
Lugar de trabajo	1.- Los participantes se ubicaron en lugares alejados de los demás. 2.- Los participantes usaban sus laptops personales para programar.	-Problemas de comunicación y seguimiento. -Perdidas de archivos, desorganización y tiempos perdidos.
Participantes y Horarios	1.- Se reclutaron 2 residentes, con conocimientos básicos en el lenguaje de programación php y java.	- Al surgir la solicitud de otro proyecto no se podía atender.
Horarios de participantes	1.- Se establecieron horarios de 4 horas sin condición de ser continuas.	- Algunos participantes pierden la intencionalidad.
Herramientas	1.- Utilizar cualquier IDE o editor de código y cualquier Modelador de datos 2.- Las modificaciones y nuevos archivos eran copiados al servidor a través de pen drive. 3.- Las pruebas se realizan en cada laptop.	- Problemas al pasar datos de un equipo a otro. - Perdida de tiempo al pasar la información por pendrive - Al subir a producción el proyecto no funcionaba.
Tamaño y tiempo de proyectos	1.- Los tipos de proyectos se manejan por igual, los tiempos acordados con el usuario es establecido por el Coordinador del departamento junto con el usuario que lo solicito.	- Los proyectos se entregan fuera del tiempo establecido y sin todos los módulos comprometidos.
Documentación y pruebas	1.- Los requisitos se anotan en una hoja blanca y se discute entre el coordinador del departamento y el usuario. 2.- Los artefactos que se generan del sistema no son importantes, lo importante es el desarrollo del sistema. 3.- Se realizan pruebas con el usuario para ver el funcionamiento de lo solicitado.	- Al finalizar un sistema las hojas de requisitos desaparecen o extravían. - Para agregar otro modulo a un sistema se requiere el doble del tiempo que se llevó el mismo.

TABLA II
ANALISIS DE VARIABLES EN EL ESCENARIO DOS

	ESCENARIO DOS	RESULTADOS
Tipo de codificación	1.-Utilización de códigos comprobados y homogéneos. 2.- Diseño único para todos los sistemas. 3.- Programación individual 4.- Se realizan funciones nuevas con códigos pero se realizan pruebas del mismo antes de implementarlo.	- Minimizaron errores y optimizo el tiempo. - Mejoro apariencia y tiempo de desarrollo. - Se presentan retrasos al trabajar de manera individual los proyectos - Minimizaron errores en los módulos.
Lugar de trabajo	1.- Se estableció un lugar para el área de desarrollo sin ningún orden en particular. 2.- Los participantes usaban sus laptops personales para programar. 3.- Se incluyó una nueva pc para el desarrollo y respaldos asignado a un participante.	- Mayor comunicación y mejoro el seguimiento de los proyectos. - Perdidas de archivos, desorganización y tiempos perdidos. - Mejoró en cuestión de control de versiones de los archivos y elimino la perdida de los mismos.
Participantes	1.- Se reclutaron 3 residentes 1 servicio social, con conocimientos básicos en el lenguaje de programación php y java.	- Se atendieron otros proyectos y se aminoré el tiempo de entrega por módulos.
Horarios de participantes	1.- Se establecieron horarios de 4 horas pero continúas.	- Mayor concentración de los participantes y mejoró el tiempo de entrega.
Herramientas	1.- Se estableció el *IDE NetBeans* y el *MysqlWorkBeanch* para codificación, depuración y modelado de datos. 2.- Se instaló un equipo con software de control de versiones 3.- Las pruebas se realizan en cada laptop y pc.	- Mayor facilidad para corregir errores - Mayor organización de proyectos - Mejoró los tiempos de modificación. - Mejoró el control de archivos. - Perdida de tiempos al pasar los módulos a producción.
Tamaño y tiempo de proyectos	1.- Se categorizan los sistemas en transaccionales, toma de decisiones, de escritorio y web. 2.- Los tiempos de los proyectos se deciden en conjunto con el equipo y el usuario que lo solicitó.	- Se establecen más adecuadamente los tiempos de entrega.
Documentación y pruebas	1.- Se diseña, utiliza y resguarda una hoja de especificación de requisito para cada modulo o solicitud. 2.- Se diseña, utiliza y resguarda una hoja de modelado de datos. 3.- Se realizan pruebas con el usuario y el equipo para ver el funcionamiento de lo solicitado.	- Fundamentar los módulos permitió mayor organización y control de las modificaciones. - Incremento en el desarrollo. - Mejoró el análisis y los tiempos de entrega.

TABLA III
ANALISIS DE VARIABLES EN EL ESCENARIO TRES

	ESCENARIO TRES	RESULTADOS
Tipo de codificación	1.-Incluyo un marco de trabajo. 2.- Una única plantilla para todos los proyectos. 3.- Programación en pares. 4.- Se implementa un curso de capacitación y entrenamiento para los nuevos integrantes	- Más facilidad en el mantenimiento. - Incremento la seguridad y conocimiento de los programadores. - Minimizaron errores en los módulos.
Lugar de trabajo	1.- Se agregó un pizarrón y proyectos en la sala de desarrollo. 2.- Se incorporaron PCS específicos para el área.	- Mejoro la etapa de análisis y comprensión de temas avanzados. - Mejoró en cuestión de control de versiones de los archivos y elimino la perdida de los mismos.
Participantes	1.- Se reclutaron 2 residentes 2 servicio social, con conocimientos básicos en el lenguaje de programación php y java.	- Se mantienen más tiempo los practicantes de servicio social apoyando al departamento.
Horarios de participantes	1.- Se establecieron horarios de 5 horas continúas.	- Mayor involucramiento, concentración de los participantes y mejoró el tiempo de entrega.
Herramientas	1. Se agregaron herramientas de generación de código y formularios. 2.- Se estableció el *IDE NetBeans* y el *MysqlWorkBeanch* para codificación, depuración y modelado de datos. 3.- Se incorpora escenarios de test unitarios para el marco de trabajo 4.- Se incorpora la gestión de proyectos mediante software jira	- Mejora del tiempo de desarrollo. - Menos errores de diseño. - Mejoró la velocidad y acceso de las aplicaciones. - Identificamos problemas de codificación. - Integra una de base de conocimientos.
Tamaño y tiempo de proyectos	1.- Se categorizan los sistemas en transaccionales, toma de decisiones, de escritorio y web. 2.- Los tiempos de los proyectos se deciden en conjunto con el equipo y el usuario que lo solicitó.	- Se entregan los proyectos en tiempos, por módulo.
Documentación y pruebas	1.- Se utiliza hoja de especificación de requisito. 2.- Hoja de modelado de datos. 3.- Se identifican y diseñan casos de uso	- Mejor control y desarrollo de los módulos. - Incremento el tiempo de entrega de módulos. - Se mantiene una base de conocimientos para los nuevos integrantes.

Las solicitudes de modificación de sistemas de información existentes, fueron de 2 por semestre, sin embargo, el tiempo ocupado para esta actividad era relativamente igual al diseño de un sistema nuevo, ya que la técnica de programación de los sistemas existentes era variada, sin patrones de diseño y sin documentación.

a)Técnicas de programación: Para este escenario se descartó la posibilidad de crear los módulos desde cero, así como utilizar un cms y se optó por la implementación de un framework llamado Yii, porque el lenguaje elegido fue Php.

b)Espacio de Trabajo: Se estableció una oficina con 7 mesas, un pizarrón y un proyector.

c)Numero de programadores y horarios: Se reclutaron 2 practicantes y 2 residentes, con horario de 9 a 5 pm establecido para todos, en una segunda fase se reclutaron 3 practicantes más.

d)Herramientas de apoyo: Se utilizó Netbeans + Plugins Yii Framework, MysqlWorkBeanch para el modelado lógico, Subversión para el control de versiones, entorno de desarrollo lamp en Linux / Centos, jira educativo para la gestión del proyecto.

e)Tamaño y tiempo de proyectos: La tabla IV, muestra los módulos identificados y tiempos de desarrollo para el sistema de eventos deportivos.

TABLA IV
MODULOS Y TIEMPOS DE PROGRAMACIÓN PARA EL SISTEMA DE EVENTOS DEPORTIVOS

Módulo	Tiempo
Casos de uso, modelado lógico y físico.	5 horas
Maquetado y adaptación del diseño de la plantilla	8 horas
Gestión Usuarios	8 horas
Gestión de Roles y Permisos	5 horas
Envío de correo	5 horas
Registro de participantes	8 horas
Generación de hoja de registro en pdf	5 horas
Exportación de datos a excel	5 horas
Backend / administración del sistema	8 horas
Pruebas	10 horas
Total	**67 horas**

Para los módulos crud (créate, read, update, delete), se utilizó una herramienta de scaffolding incluida en el framework yii, su nombre es gii, la cual genera automáticamente los controllers, models, views y forms correspondientes al crud [10].

Después de diseñar los módulos de un sistema, estos son reutilizados en los próximos proyectos, aminorando el tiempo de desarrollo.

f)Artefactos y documentación: Para este proyecto se elaboraron: formato de requisitos, roles y actores, casos de uso, modelo lógico E-R, diccionario de datos y formatos de pruebas.

6 RESULTADOS

Después de tres fases de pruebas en un período de 4 semestres, se mantiene mayor estabilidad en el departamento de desarrollo de software, integrado por el jefe de departamento, practicantes y residentes. Se ha logrado elaborar al menos 7 sistemas de información, para eventos académicos, sociales y culturales, por ejemplo: Sistema de eventos deportivos, sistema de administración de concursos de Innovación, sistema de reuniones regionales, sistema de simposio, congresos y administración de ponencias, Sistema de control escolar de lenguas extranjeras (la tabla V, detalla los módulos y tiempos de desarrollo), así como un Sistema cms del portal institucional. Además se encuentran en fase de desarrollo los siguientes sistemas: Sistema de Gestión del Seguro social (la tabla VI, detalla los módulos y tiempos de desarrollo), Sistema de gestión de inventarios, Sistema de mesa de ayuda y sistema de gestión de metas.

TABLA V MODULOS Y TIEMPOS DE PROGRAMACIÓN PARA EL

SISTEMA DE CONTROL ESCOLAR DE LENGUAS EXTRANJERAS

Módulo	Tiempo
Casos de uso, modelado lógico y físico.	5 horas
Maquetado y adaptación del diseño de la plantilla	8 horas
Gestión Usuarios	8 horas
Gestión de Roles y Permisos	5 horas
Envío de correo	5 horas
Inscripción de alumnos	8 horas
Generación de hoja de registro en pdf	5 horas
Exportación de datos a excel	5 horas
Backend / administración del sistema	30 horas
Vinculación con referencias bancarias	10 horas
Creación de cursos propuestos por alumnos	10 horas
Captura de calificaciones	16 horas
Pruebas	10 horas
Total	**125 horas**

TABLA VI MODULOS Y TIEMPOS DE PROGRAMACIÓN PARA EL

SISTEMA DE CONTROL DEL SEGURO SOCIAL

http://sass.itvillahermosa.edu.mx/

Módulo	Tiempo
Casos de uso, modelado lógico y físico.	5 horas
Maquetado y adaptación del diseño de la plantilla	8 horas
Gestión Usuarios	8 horas
Gestión de Roles y Permisos	5 horas
Envío de correo	5 horas
Registro de estudiantes	8 horas
Generación de constancias en pdf	5 horas
Exportación e Importación de datos de excel	15 horas
Backend / administración del sistema	20 horas
Consumir servicio web de la CURP	20 horas
Algoritmo de conversión para imss	30 horas
Pruebas	10 horas
Total	**139 horas**

7 CONCLUSIÓN

Las metodologías ágiles de gestión y desarrollo de software proponen una filosofía adaptable, flexible y escalable para integrarlas en variados escenarios de trabajo. Además ofrecen beneficios de productividad y eficiencia sin perder calidad en los productos.

Este artículo propone mecanismos, herramientas y técnicas que permiten a las instituciones y empresas con escaso personal o personal eventual, establecer un área de desarrollo de software, apoyado de practicantes y/o residentes, integrando elementos de la metodología ágil scrum combinada con la metodología de desarrollo rud, que le permitan la construcción de sistemas de información de calidad, en periodos cortos de tiempo.

En este contexto, se requiere del esfuerzo inicial del personal de base el cual debe considerar el aprendizaje de un marco de trabajo robusto con una curva de aprendizaje corta, cubriendo al principio los elementos de estandarización de los módulos o códigos, adaptación de un escenario adecuado para la ejecución de la metodología, reclutamiento de practicantes y residentes, dominar un conjunto de herramientas case de desarrollo y conocer la filosofía y técnicas de las metodologías.

Los elementos, actividades, supuestos, herramientas y prácticas mencionadas en este documento, pueden ser utilizados en cualquier área de desarrollo, incluso ser modificadas y adaptadas a diversos entornos.

REFERENCIAS

1. Lledó, P., Gestión Ágil de Proyectos: Lean Project Management. Trafford, 2012, 10-33
2. Priolo, S., Métodos Agiles, Users, 2009.
3. Somerville, F. I., Ingeniería de software, Pearson educación, 2005, 358-361
4. Manifiesto ágil, recuperado en 2015, http://www.agilemanifesto.org/iso/es/.
5. http://www.proyectosagiles.org/que-es-scrum, consultada en 2015, metodología scrum.
6. Laynes, J., Desarrollo de Software Ágil: Extreme Programming and Scrum. Createspace, 2014, 116-119.
7. Somerville, F. I., Ingeniería de software, Pearson educación, 2005, 364-366.
8. Cockburn, A., Crystal Clear: A Human-Powered Methodology for small teams, Addison-Wesley Professional, 2004.
9. Somerville, F. I., Ingeniería de software, Pearson educación, 2005, 123-126
10. Lauren, J., James, R., Yii Rapid Application Development. Pack Publishing Ltd, 2012.

Acerca de los autores:

José Gómez Zea nació en Cd. Pemex, Macuspana, Tabasco, México, el 8 de marzo de 1976. Se graduó en el Instituto Tecnológico de Villahermosa en el año 2001 como Licenciado en informática y recibió el grado de Maestro en Tecnologías de la Computación por la Universidad Mundo Maya en al año 2004. Se ha desempeñado como administrador de proyectos, capacitador, programador independiente y docente a nivel profesional y posgrado.

Ejerció profesionalmente en las empresas Todo Comunicación, Escos S.A de C.V., Colegio Latinoamericano de Tabasco, IEU, CIID, Universidad Mundo Maya. Actualmente es profesor y director del área de T.I. del Instituto Tecnológico de Villahermosa. Entre sus campos de interés se encuentra la programación, las telecomunicaciones y el desarrollo de sistemas.