URKUND

Urkund Analysis Result

Analysed Document:	propuesta_version_urkund20181016231329_Taipe.docx (D42897818)
Submitted:	10/22/2018 7:45:00 PM
Submitted By:	dverap@unemi.edu.ec
Significance:	0 %

Sources included in the report:

Instances where selected sources appear:

0

CAPÍTULO 1

PROBLEMA

1.1 Planteamiento del problema

En el instante de efectuar el desarrollo de un sistema se necesita ejecutar un primer paso, realizar el proceso de levantamiento de la información, lo cual radica en la definición del sistema que el usuario necesita, esto resulta ser una molestia para los clientes o posibles usuarios hacia los desarrolladores, ya que éstos últimos no poseen sus necesidades claras, lo que involucra inconvenientes al instante de modelar sus ideales en la adquisición de un sistema, por varios factores influyentes como poca comunicación, problemas en la asignación de las responsabilidades, variaciones en la planificación, así como también molestias en la financiación de un proyecto de desarrollo de software lo que incurre a un desgaste de tiempo. Es por eso que, para prevenir variaciones de argumentos, en los posteriores períodos del ciclo de vida, surgen las metodologías agiles como un adjunto que no pertenece a las etapas del desarrollo del software, que tiene como función crear la documentación al momento del desarrollo de software, de esta manera se reduce el tiempo y se apoya a los ciclos para la creación del software. En cuanto a la comunicación, se estima como un punto decisivo y trascendental en cualquier proyecto y con mayor razón en un software, la correlación entre desarrollador y usuario debe ser inmediata, cuando esto no funciona genera un enorme desequilibrio; a pesar de aquello el resultado final requerido es la valoración de requerimientos y el trabajo en conjunto. En el tema de la asignación de responsabilidades y de la misma forma colaboración del cliente es de suma importancia al momento de asignar tareas o definir roles de los participantes, así se podrá originar nodos de responsabilidad en escalas menores, con la finalidad de disminuir el tiempo necesario en el progreso del producto final o software. Además, la planificación de un software requiere considerar varios aspectos como recursos, responsables, tiempo y actividades que no solo sirven para establecer el punto de inicio sino también para realizar una comparación entre lo que se ha planificado y lo que ha sido ejecutado, hasta realizar acciones a tiempo para la obtención de un determinado producto en el sistema. Así mismo otro componente que se afecta es la financiación que se encuentra relacionado directamente con una apropiada y oportuna planificación, el cual deberá ser considerado para las etapas del ciclo de vida del software, esto es: la planificación, el desarrollo, la evaluación, medición y control del mismo.

1.2 Objetivo general • Analizar las metodologías ágiles para el desarrollo de software, aplicando el modelo descriptivo en el contexto para su observación y conclusión.

1.3 Objetivos específicos • Delimitar el problema de investigación en relación a las metodologías ágiles para el desarrollo de software • Determinar fuentes documentales sobre las metodologías ágiles • Proponer una herramienta que optimice el tiempo en el desarrollo e implementación de software.

1.4 Justificación

La importancia de esta labor se debe a la elaboración de un estudio documental para la comparación entre resultados de estudios realizados y enlazados al asunto anteriormente propuesto, en el que la comunicación, la planificación, la asignación de responsabilidades y el financiamiento contribuyen a la elaboración de un sistema que es solicitado por el cliente o usuario.

Cabe destacar que hay sistemas con estructuras y diseños de mala calidad generando insatisfacción del usuario que realizó el requerimiento del software y de la misma manera la vulnerabilidad refiriéndose propiamente a las seguridades del sistema, precisamente por la falta de aplicación de metodologías ágiles como requisito fundamental en el desarrollo del software. Todo esto proviene por la escasez de conocimientos acerca de estas nuevas metodologías, además de la falta de experiencia para la aplicación de aquellas y el poco interés por el lado del desarrollador en cuanto al fomento y aplicación de metodologías como parte del transcurso en el desarrollo del software.

Otro de los factores que influyen es la manera lexical en la que el usuario utiliza al momento de expresar los requerimientos del diseño del sistema y la mínima aportación de responsabilidad que ofrece al desarrollador.

CAPÍTULO 2

ANTECEDENTES Y MARCO TEÓRICO

2.1 Antecedentes de la investigación

Avison y Fitzgerald CITATION Avi95 \n \t \l 12298 (1995) establecieron una clasificación temporal sobre la aparición de las diferentes metodologías en el tiempo, que incluye cuatro periodos:

• Periodo de pre-metodologías • Primeras metodologías • La era de las metodologías • Era post – metodológica

El primer periodo es de pre – metodologías. Se posiciona temporalmente durante los años 50 y 60 del siglo XX, caracterizado por la carencia de metodologías en el desarrollo de software. Esta época era el comienzo de los grandes computadores, los cuales fueron vinculados a aplicaciones militares y científicas. Los programadores eran quienes realizaban la implementación de estos sistemas, aunque no eran necesariamente buenos comunicadores, y de esta manera dificultaban la comunicación con los usuarios. Debido a la necesidad de destinar más tiempo al análisis y diseño de la aplicación, surgieron los analistas de sistemas y operadores CITATION Car08 \l 12298 (Carvajal Riola, 2008). Los operadores cumplían con el rol de controlar el funcionamiento de las máquinas, mientras que los analistas de sistemas era un intermediario entre el usuario y programador, en ocasiones se distinguía dos tipos: el de negocio que se encargaba de reunir las necesidades del sistema y hacérselo saber a un segundo analista, el técnico, quien realizaba el diseño de la solución y la trasladaba al programador. La segunda aparición fue de las metodologías iniciales, periodo establecido alrededor de 1970 y 1980, siendo considerado como la etapa de la juventud. En 1971 Yeats y Daniel plantearon la inicial metodología fundamentada en el ciclo de vida de desarrollo de un

software, “el modelo waterfall” CITATION Tin10 \l 12298 (Tinoco Gómez, Rosales López, & Salas Bacalla, 2010). Se encuentra formada por un grupo de etapas que debían ser ejecutadas de manera secuencial, es decir, una fase debe estar culminada para que pueda comenzar la siguiente (razón por la cual se denomina waterfall o cascada) y cada una de las fases poseen unos entregables o salidas para poder ser consideradas como finalizadas (figura 1). Esta fue la primera metodología y que hoy en día se la conoce como metodologías formales o tradicionales, que fundamenta su potencia principalmente en la ingeniería de procesos CITATION Her07 \l 12298 (Herrera Uribe & Valencia Ayala, 2007).

Figura 11 Modelo en cascada Fuente: Elaboración propia

Pese a esto, Systems Development Life Cycle (SDLC), posee limitaciones muy severas como la inestabilidad (por los cambios constantes de los requerimientos del cliente), imposibilidad en deducir las necesidades más importantes del cliente (por centrarse en el avance de la eficiencia en cuestión de tecnología), poca flexibilidad, exceso de documentación, insatisfacción por parte del cliente, e incumplimiento del tiempo previamente planificado (por intentos de cambiar el sistema para incluir nuevos requisitos del sistema). Frente a las limitaciones de SDLC, aparecieron nuevos enfoques y se da el inicio de la era de las metodologías. Se diferencia dos corrientes en este periodo, una que busca la mejora del modelo waterfall y otra que trata de realizar algo distinto. Cabe destacar que el modelo waterfall fue mejorando mediante la incorporación de herramientas y técnicas surgidas en los años 70, como por ejemplo, normalización, modelado de entidades relacionales, diagramas de actividades, de flujo de datos, y estructurados; además se destacan repositorios de modelado y herramientas Computer Assisted Software Engineering o CASE. Las metodologías que pueden ser consideradas como mejores versiones de SDLC son Yourdon System Method, SSADM o Merise, quienes integran técnicas para actualizar y mejorar el modelo del ciclo del software. En el transcurso de los años iniciales de la década de los noventa seguían apareciendo metodologías que pueden ser consideradas en la era metodológica, que se clasifican en grupos: orientadas a objetos (unifica las metodologías de desarrollo de los sistemas de información), desarrollo incremental o evolutivo (desarrolla software siempre a partir de anteriores versiones y jamás desde cero), y específicas (especificadas para un modelo de aplicaciones). Aun así con las metodologías que eran utilizadas, ya sea el modelo waterfall o sus versiones mejoradas, muchos usuarios no se encontraban del todo satisfechos, ya que gran parte de las metodologías fueron diseñadas para ocasiones especiales y eso no existe, cada proyecto, cada ocasión es distinta; por lo tanto, varias herramientas y técnicas podían ser utilizadas o no, según la situación. El fracaso o éxito del desarrollo del software no necesariamente se debe al desuso o uso de las metodologías, sin embargo, a finales de los 90 aparece una corriente que se caracterizó por una estricta reevaluación de los beneficios que brindan las metodologías, e incluso se visualiza una reacción violenta en contra de las metodologías, junto con un grupo de varios enfoques no metodológicos, lo cual da lugar a la denominada era post metodológica. Fueron varias las causas para explicar la adversa reacción al uso de las metodologías: falta de productividad, complejidad, habilidades requeridas, herramientas difíciles de usar, no contingente, enfoque unidimensional, inflexibilidad, desplazamiento del objetivo, poco orientadas al entorno y personas CITATION Avi95 \l 12298 (Avison & Fitzgerald, 1995). Cada vez que no algo funcionaba, surgía una reacción para

solventar los problemas; entre las alternativas que se proponen en función a la mentalidad de los usuarios y necesidades de cada empresa son: • Desarrollo externo: soluciones ERP • Mejora continua: siempre puede ser mejor • Desarrollo ad – hoc: se fundamenta en la habilidad y experiencia de los desarrolladores • Desarrollo flexible: enfoque contingente • Desarrollo ágil: participación de clientes y usuarios Las metodologías ágiles surgen como una alternativa en reacción frente a las metodologías tradicionales. Varios de los factores que influyeron fueron el "plumbing", es decir, la pesadez por la lentitud de reacción, demasiada documentación y sobre todo falta escasez de agilidad; las metodologías existentes no cumplían con las expectativas previamente planeadas; explosión de aplicaciones web y el movimiento open source. Estas metodologías buscan resolver los problemas surgidos, ya que las necesidades y expectativas de parte de los usuarios se volvieron más frecuentes y urgentes. Fue entonces que al comienzo de los 90 nacieron propuestas metodológicas para llegar a resultados con mayor velocidad en el desarrollo de software sin disminuir la calidad CITATION Her07 \l 12298 (Herrera Uribe & Valencia Ayala, 2007).

En febrero del 2001, en EEUU – Utah, se realizó una reunión, y es ahí donde surge el término "ágil" que luego sería adaptad al desarrollo de software. En esta reunión tuvieron participación 17 personas consideradas expertas en el ámbito de la industria del software, ya que entre ellos se encontraban creadores o impulsadores de las

metodologías de software. Se buscaba proponer una opción distinta a los métodos de desarrollo de software tradicionales, los cuales tenían como características ser rígidos además de ser dirigidos por una documentación, la cual se origina en las acciones desarrolladas. CITATION Let \l 12298 (Letelier & Penadés, 2003). Los que integraron dicha reunión resumieron los principios en los que se fundamentan los métodos alternativos en un incorporado de cuatro postulados, al cual denominaron Manifiesto Ágil CITATION Can03 \l 12298 (Canós, Letelier, & Penadés, 2003). 2.2 Marco Teórico Metodologías ágiles Posee como principios los que se muestran a continuación CITATION Can03 \l 12298 (Canós, Letelier, & Penadés, 2003): 1. Lo principal es el contentamiento del cliente mediante continuas y prontas entregas de software que generen un valor 2. Los cambios a los requerimientos son bienvenidos 3. La entrega frecuente de software que funcione, en cortos periodos de tiempo 4. El trabajo en conjunto de las personas negociadoras y de los desarrolladores 5. Elaborar proyectos en función de intereses con motivación 6. La conversación

en persona y en directo como medio efectivo para obtener información 7. El software que funcione correctamente es la manera más notable de progreso 8. Los procesos ágiles ejecutan el proceso de estimular

un desarrollo sostenible 9. La agilidad incrementa mediante la cuidado intensivo hacia a la excelencia 10. La simplicidad es esencial 11. Los mejores diseños, requisitos y arquitecturas nacen de equipos organizados por ellos mismos 12. En periodos de tiempo los integrantes del equipo reflexionan sobre elementos para producir con mayor efectividad. La encuesta del "Estado de Desarrollo Ágil" efectuada por CITATION Col09 \l 12298 (CollabNet VersionOne, 2009), 1.942 respuestas, presenta las metodologías ágiles más usadas (figura 2).

Figura 22 Metodologías ágiles usadas en el 2018 Fuente: CITATION Col18 \l 12298 (CollabNet, 2018) SCRUM

Es un marco de trabajo que controla el estado del software, el cliente identifica y determina las necesidades prioritarias, luego el equipo SCRUM se organiza y determina la mejor manera de entregar los resultados CITATION Abr02 \l 12298 (Abrahamsson, Salo, Ronkainen, & Warsta, 2002). SCRUM fue originado en el año de 1986 por Ikujiro Nonaka y e Hirotaka Takeuchi, ellos describieron una aproximación metodológica muy novedosa que realiza un incremento en la flexibilidad y rapidez en el proceso de nuevos productos comerciales. Esta metodología tuvo sus inicios en el campo de la industria de tecnología y automovilística, pero en el comienzo de los años 90 Ken Schwaber la puso en marcha en Advanced Development Methods CITATION Mou \l 12298 (Mountain Goat Software, 2011) de la misma forma que Extreme Programming, SCRUM enfatiza en la gestión del recurso humano, lo cual se hace visible en los rasgos que este método posee, y serán explicados a continuación.

Características de Scrum

SCRUM brinda preferencia a las interacciones e individuos sobre las tareas y procesos, lo que representa que el éxito del proyecto se origina en la representación como el equipo realice la organización para efectuar el trabajo. Se debe poseer un enlace dinámico de equipo, pues para lograr el triunfo de un hito no debe realizarlo un solo miembro sino el equipo completo de SCRUM; todos ayudan entre sí, y estimulan a los que integran y que no se encuentran a la par con todo el equipo CITATION Bec01 \l 12298 (Beck, 2001). La orientación SCRUM plantea el software funcional por la masiva documentación, diferente a RUP que es muy estricto en dicho tema. Se muestra al cliente las procedimientos operables y no solamente reportes de progreso, así el cliente decide si se continúa o no, mientras que en otras perspectivas el resultado se espera ver hasta el final. De igual manera, SCRUM propone la cooperación con el cliente en vez de las contrataciones de forma rígida.

Valores de Scrum Scrum propone valores CITATION Sut07 \l 12298 (Sutherland & Schwaber, 2007) que brindan ayuda en el esclarecimiento de los medios de la metodología y además ayuda a garantizar la evolución y cumplimiento de SCRUM, que son: Compromiso y empoderamiento de las personas: se trata atribuir y delegar responsabilidades con el fin de que el equipo pueda organizarse entre ellos mismos y tomar medidas acerca del desarrollo del proyecto. Enfoque en desarrollar lo previamente dicho: cada miembro del equipo debe comprometerse con el equipo y desarrollar lo estipulado con el cliente. Visibilidad y transparencia del proyecto: el equipo debe estar informado sobre cualquier anomalía y responder con transparencia, ya que cualquier error o falla que no se converse puede afectar al resto del proceso. Respeto entre personas: debe existir confianza entre los miembros, además de respetar cada una de sus capacidades y conocimientos, ya que las formas de cada uno resultan ser las fortalezas del equipo. Responsabilidad y coraje: se debe poseer auto disciplina y no una asignada, cada miembro del equipo debe encontrarse dispuesto a sortear conflictos y proceder de manera positiva ante los cambios que puedan surgir. Roles de Scrum En los procesos de desarrollo deben existir roles, son los que establecen actividades y comportamientos de gran importancia en el proyecto. SCRUM define su equipo de trabajo

CITATION Ris00 \l 12298 (Rising & Janoff, 2000) en cinco grupos: • Propietario del producto: es quien

realiza lo necesario para que el valor del producto aumente, conoce cuál será el producto final • SCRUM Manager (Scrum Master): su función es liderar el equipo para obtener los objetivos planificados hasta que se llegue al último sprint •

Equipo Scrum (Development Team): profesionales que realizan la entrega del incremento de producto "terminado" • Interesados • Usuarios Artefactos de Scrum Son los varios modelos de información que se generan en el transcurso del desarrollo de software CITATION Mou \l 12298 (Mountain Goat Software, 2011), SCRUM produce los siguientes: • Pila del producto (Product Backlog): el dueño del producto es quien decide y responsable, aun así se encuentra en evolución y accesible a todos los roles. Selo denomina como el corazón de SCRUM. • Pila del sprint (Sprint Backlog): Requisitos implicados con el equipo para el sprint, se realizan con el nivel de referencia suficiente para su ejecución. • Incremento: parte del producto avanzado en un sprint, es viable que sea usado, posee las pruebas, una codificación pura y documentada.

Reuniones de Scrum Es un elemento fundamental de la metodología SCRUM CITATION Ris00 \l 12298 (Rising & Janoff, 2000) realizadas de forma periódica, cada una indica los resultados a generar: • Planificación del sprint: el dueño del producto expone las precedencias e indecisiones del equipo, evalúan las necesidades con prioridad y desde la misma generan la pila de sprint. El objetivo del sprint es definido en una frase por el SCRUM Manager. • Reunión diaria: comprendida entre 15 a 30 minutos de duración, igual lugar y hora de reunión. La dirige el SCRUM Manager y solo interviene en Equipo SCRUM; se realizan preguntas como: ¿Qué hiciste ayer?, ¿Qué trabajo vas a realizar hoy?, ¿Qué necesitas?, así se reforma la pila del sprint y el SCRUM Manager toma decisiones inmediatamente, además de fijar los obstáculos para ser resueltos de manera externa y así no alargar el lapso de reunión. • Revisión del sprint: reunión informativa con una duración de aproximadamente 4 horas, el SCRUM Manager es el moderador de la misma; se presenta el incremento, la planeación de sugerencias y se anuncia el próximo sprint. • Retrospectiva del sprint: luego de cada sprint, por aproximadamente 4 horas se congregan los miembros del equipo y opinan sobre el sprint superado, con el propósito de optimizar los procesos; considerada como reunión de mejoramiento y evaluación Proceso Scrum Ya que SCRUM se enfoca más en la distribución del equipo de trabajo, fragmenta el proyecto en etapas de aproximadamente 4 semanas, cada uno es denominado Sprint y el equipo recoge una lista de pedidos que deben ser ejecutados en un Sprint específico. En el transcurso de desarrollo de software se enlazan las reuniones, valores y artefactos de SCRUM (ver figura 3), se forma de 5 fases que contienen las actividades que serán desarrolladas.

Figura 33 El proceso Scrum Fuente: CITATION Sut07 \l 12298 (Sutherland & Schwaber, 2007) • Revisión de planes de Release

Es la "planificación del Sprint". Se ejecuta luego de establecer la pila de producto del equipo para evaluar factibilidad de requerimientos. • Distribución, ajustes y revisión de estándares de producto Es la "pila de Sprint". Los desarrolladores ajustan los requerimientos y estándares mínimos para iniciar la fase de Sprint. • Sprint Durante aproximadamente 30 días se realiza el

desarrollo del software y se efectúan las reuniones. • Revisión de Sprint Es el “incremento”. Se examina el Sprint y en caso de requerir se agregan ítems a la pila de producto, esto se frecuenta hasta que esté listo el producto. • Cierre Se realiza la corrección de errores y depuración, esto se frecuenta hasta lograr calidad en el producto. Luego de las pruebas y correcciones se efectúa el Marketing y avance del producto y así queda cerrado el proyecto. KANBAN “El Kanban es un sistema de gestión donde se produce exactamente aquella cantidad de trabajo que el sistema es capaz de asumir.” CITATION Ber10 \l 12298 (Bermejo, 2010) Es un sistema de gestión del trabajo en ejecución, que de manera general sirve para el aseguramiento de una continua producción y sin exceso de cargas; genera el mismo total de trabajo de la que el sistema se responsabiliza. “El Kanban es un sistema de trabajo just in time, lo que significa que evita sobrantes innecesarios de stock…” CITATION Ber10 \l 12298 (Bermejo, 2010), esto corresponde a realizar un esfuerzo e inversión que no es prioritario y así se evita demasiada carga para el equipo.

Figura 44 Metodología Kanban Fuente: CITATION Omn15 \l 12298 (Omniasolutions, 2015)

Valores de Kanban Kanban es considerado como la orientación de Lean al desarrollo de software, que es un conjunto de principios determinados a continuación CITATION Rub09 \l 12298 (Rubio Torá, 2009): • De entrada una excelente calidad: todo lo ejecutado, debe ser bien realizado, no de manera rápida, pues hacer algo con rapidez para luego tener que repararlo no resulta, es conveniente hacerlo bien desde el comienzo. • Minimización del desperdicio: ejecutar lo justo y necesario y no desviarse en tareas que no son principales o que no se las necesita realizar. • Mejora progresiva:

perfeccionamiento sucesivo en el desarrollo, en base a los objetivos que se desean alcanzar • Flexibilidad: Aquello que se va a realizar deberá ser concluido del Backlog pendiente, con esto las tareas podrán ser condicionadas y con prioridad Objetivos de Kanban • Elaborar un punto de equilibrio entre la capacidad y la demanda • Delimitar el trabajo que está siendo ejecutado

• Avanzar de manera progresiva el trabajo

• Manifestar a tiempo los inconvenientes con el ritmo sostenible

• Examinar el trabajo (no el personal)

• Ordenar, concordar y exponer toma de decisiones generadoras de valor y cuellos de botella

• Adquirir grupos auto organizados

• Obtener un tipo de cultura de optimización incremental

Beneficios de Kanban

• Adapta los flujos de valor existentes

a medida • Normas sencillas

que ofrecen la optimización del trabajo

• Sobresaliente administración del riesgo

• Pasividad al experimento

• Propagación viral en la organización con mínima oposición

• Aumento de la asistencia dentro y fuera del equipo

• Perfeccionamiento de la eficacia del trabajo

• Consonancia de trabajo sustentable y sostenido

Sistema de Kanban

Se trata

de una perspectiva orientada a la programación del trabajo. Los típicos proyectos ágiles hacen uso de iteraciones de tiempo fijadas con anterioridad. Al inicio el equipo se reúne y selecciona las historias de Backlog que tengan la posibilidad de ser entregadas, tal como lo hace Scrum. Durante la elaboración, aquellas

historias son desarrolladas y al final son entregadas, EXTREME PROGRAMMING XP La programación extrema es una de las metodologías

que posee mayor rapidez y eficiencia, el cliente está presente en cada fase; el trabajo en grupo ayuda a reducir los errores al momento de ejecutar las pruebas correspondientes que han sido

establecidas para cada parte de código del proyecto que está en proceso de desarrollo. En 1999 fue publicado por Kent Beck y se lo designó como Programación Extrema debido a las reconocidas prácticas en el desarrollo de software además de la intervención del cliente en los niveles más extremos CITATION Wel18 \l 12298 (Wells, 2013).

Figura 55 Ciclo de entrega de XP Fuente: CITATION Wel18 \l 12298 (Wells D. , 2013)

Proceso de XP Al no existir más ciclos se entiende que el sistema ha alcanzado su objetivo, caso contrario se debe continuar con el desarrollo de ciclos para añadir la funcionalidad esperada. Cada fase del ciclo establece lo siguiente:

Figura 66 El proceso XP Fuente: CITATION Pre10 \l 12298 (Pressman, 2010) • Fase de planeación: comienza con las historias de usuario que representan las funcionalidades y características del software. • Fase de diseño: Debe procurarse diseños sencillos y simples para que el desarrollo pueda ser realizado de una manera más fácil. • Fase de codificación: Los desarrolladores deben plantear las pruebas de unidad que serán ejercitadas en las historias de usuario. • Fase de pruebas: Las pruebas son implementadas con un marco de trabajo que puedan ser automatizados, con el fin de elaborar pruebas de integración y realizar validaciones diarias.

GENEXUS Es un sistema que posibilita una administración buena y automática sobre el conocimiento de los sistemas de negocios. Está fundamentado en un prototipo muy diferente a los comunes empleados para el desarrollo de sistemas: no nace de un modelo de datos existente anteriormente ni tampoco de ideas abstractas sobre lo importante de la empresa y lo que no resulta ser. Genexus es desarrollado en Uruguay, en Artech, un laboratorio que permanece en dicho país porque de esa manera se beneficia en gran manera de los Ingenieros en Sistemas de la localidad y por la concordancia que produce dentro de la industria de software local. Es desarrollado mediante tecnología de punta que existe a nivel mundial y una fundamental parte de su programación, de manera particular las pruebas y prototipos, son desarrolladas a través de “predicates logic” y Prolog, su lenguaje. Genexus parte de las distintas perspectivas y enfoca el modelo de datos ideal derivado de las mismas; además sistematiza todo aquello en una Base de Conocimiento, y también la descripción de los enfoques de los usuarios.

CAPÍTULO 3

ANÁLISIS DE ALTERNATIVAS DE SOLUCIÓN

PROPUESTA 1: SCRUM COMO METODOLOGÍA ÁGIL PARA DESARROLLO DE SOFTWARE

Fundamentación teórica Scrum es una metodología ágil que establece el desarrollo incremental de los requerimientos del proyecto en pequeños y temporales bloques o en ocasiones fijos (es decir en iteraciones de por lo general un mes o dos semanas, de así requerirlo). Aquí se vuelve a priorizar los requisitos de forma regular, en cada iteración, dependiendo del valor que se le proporcionan al cliente en ese instante y el coste que se estima por desarrollo, con el propósito de obtener un mejor resultado y como base fundamental al control empírico de dicho proyecto; y se obtiene como resultado la actualización de la Product Backlog (lista de requisitos priorizada). Al terminar cada iteración se expone al cliente lo que se ha logrado obtener, de tal forma que éste pueda pensar en las decisiones correspondientes por lo observado del proyecto en ese instante. Diariamente el equipo se encuentra en constante sincronización y adaptaciones necesarias. El equipo se compromete a realizar la entrega de los requisitos establecidos y para esto se les otorga la autoridad que necesitan para desempeñar con las ocupaciones de sus roles y para organizar sus trabajos. La coordinación de la colaboración y comunicación entre el cliente y equipo es otro de los fundamentos de esta metodología, además de mencionar el timeboxing (técnica que consiste en establecer el tiempo máximo para lograr objetivos, tomar decisiones o realizar tareas) de las acciones del proyecto que son necesarias para la toma de decisiones y de esa manera alcanzar los objetivos previamente establecidos. Análisis técnico Scrum hace uso de un enfoque incremental e iterativo para la optimización de predictibilidad y el control de riesgos. Son tres los pilares que soportan la completa implementación del control de procesos, los cuales se mencionan a continuación: 1. Transparencia Aquellos aspectos característicos del proceso

desarrollo necesitan ser evidenciados para las personas que están a cargo de los resultados finales. Además, estos aspectos deben ser delimitados

por un estándar general, de tal forma que los observadores tengan participación con

un entendimiento en común de lo que está siendo visualizado. Por ejemplo: todas las personas que intervienen deben hacer uso de un lenguaje en común

al referirse de los procesos y además deben poseer una definición en común sobre "terminado". 2. Inspección Los usuarios deben realizar inspección de manera frecuente

a los artefactos de Scrum y el avance de los objetivos, para encontrar cambios inesperados; la misma no debe ser tan seguida para no interrumpir con

el trabajo, además, éstas resultan más beneficiosas al momento de ejecutarlas de manera diligente por inspectores expertos en el tema y dentro del mismo lugar de trabajo. 3. Adaptación En el caso de que un inspector compruebe que uno o varios aspectos del proceso se desalinean de los

límites aceptables y que el producto obtenido será no aceptable, el material o proceso que está siendo ejecutado deberá ser modificado. Aquel ajuste debe ser realizado

de la manera más pronta posible para disminuir mayores desviaciones. Scrum determina cuatro eventos formales, dentro de cada Sprint, para la adaptación e inspección, los cuales se presentan a continuación y son conocidos como eventos de Scrum:

Sprint Planning (Planificación del Sprint), Daily Scrum (Scrum Diario), Sprint Review (Revisión del Sprint)

y Sprint Retrospective (Retrospectiva del Sprint). PROPUESTA 2: KANBAN COMO METODOLOGÍA ÁGIL PARA EL DESARROLLO DE SOFTWARE

Fundamentación teórica Kanban es una metodología para optimizar, gestionar y delimitar servicios en los cuales se entrega trabajo del conocimiento, como por ejemplo servicios profesionales, actividades o trabajos en los que interactúan el diseño y la creatividad de productos de software y de la misma manera productos físicos. Su primordial particularidad es aquel principio de "empieza por donde estés", el cual logra impulsar el rápido cambio dentro de las organizaciones, además de reducir la resistencia ante una variación favorable enfocada y apegada a los objetivos de dicha organización. Kanban transforma el trabajo del conocimiento intangible en algo visible, de tal modo que asegura la correcta función del servicio y con la cantidad exacta de trabajo, el mismo que es pedido y solicitado por el cliente y que además el servicio tiene la capacidad de otorgar. Para esto hace uso de un sistema de flujo de entrega que condiciona la proporción de Work in Progress o WiP (trabajo en proceso) mediante la utilización de señales visuales. La herramienta usada como mecanismo de señalización, algunas veces llamado kanbans, es el tablero Kanban, que representa los márgenes del trabajo en progreso o ejecución, y que notifican la cantidad de trabajo que ingresa en el sistema, así se obtiene una mejora en el flujo de valor a los clientes. Las políticas para establecer un WiP elaboran un sistema de arrastre: el trabajo se "arrastra" al sistema en el momento que otro de los trabajos es culminado y se obtiene capacidad disponible, en vez de "empujar" aquellos trabajos al sistema cuando existe trabajo nuevo requerido. Kanban se

visualiza en la entrega de servicios de una organización, en el que una o más personas cooperan para generar productos de trabajo. Un cliente posee un servicio, y dicho cliente identifica las necesidades o trabajo para aquel para luego aceptar y dar el visto bueno a las entregas de trabajo finalizado o completado.

Análisis técnico Kanban posee tres principios directores que incluyen llamadas a la acción, y que se fundamentan en las necesidades de la organización: 1. Principio de sostenibilidad referente a obtener un foco en la mejora y un ritmo sostenible Enfocado dentro de la organización, su finalidad es construir servicios evadiendo la sobrecarga, en los cuales la demanda se encuentre en equilibrio con el contenido del sistema, así lograr la optimización del funcionamiento de los servicios teniendo presente el coste, la colaboración, el compromiso personal y sobre todo la satisfacción del cliente. Se trata de un punto de salida natural para el cambio, pues en ocasiones donde la capacidad es superada por la demanda, transforma el trabajo intangible en visible, minimiza la sobrecarga y posiblemente tenga un impacto positivo en la cantidad de trabajo ejecutado, en el tiempo establecido para finalizar cada tarea. 2. Principio de orientación al servicio orientado a lograr rendimiento y de la misma manera satisfacción del cliente Se enfoca al exterior de la organización, desde los objetivos de la misma, hacia sus clientes, su objetivo es otorgar servicios a los clientes adecuados para la finalidad que cumplen y que sobrepasan las expectativas y necesidades de los clientes. En el momento en que todos en la organización se enfocan en la prestación de servicios para sus clientes, se obtiene resultados sobresalientes. Kanban define la entrega y mejora de valor en los servicios. Emplear este principio es fundamental para el éxito de Kanban. 3. Principio de supervivencia referente al mantenimiento de la adaptabilidad y competitividad Se enfoca hacia el futuro, su objetivo es comprometerse a que una organización sobreviva y sea próspera aun en tiempos donde existen cambios e incertidumbre. El enfoque evolutivo de Kanban hacia el cambio, su acento en la tolerancia al fallo y en la mejora continua, el compromiso y respeto de los actores involucrados y el impulso de la diversidad en tecnología y procesos es una forma adecuada de objetar a constantes retos. PROPUESTA 3: EXTREME PROGRAMMING XP COMO METODOLOGÍA ÁGIL PARA EL DESARROLLO DE SOFTWARE Fundamentación teórica La programación extrema es uno de varias metodologías ágiles populares, se ha demostrado que resulta ser muy exitosa en compañías de varias industrias y tamaños en el mundo; su éxito radica en el bienestar del cliente. En vez de realizar una entrega de todo lo que se desea para una fecha lejana en el futuro, este procedimiento realiza la entrega del software en medida de lo que se necesita; permite a los desarrolladores argumentar con confianza a las variaciones requeridas por el cliente, incluso en la etapa final del proyecto. El equipo se auto organiza alrededor del problema y de esa manera encontrar solución de la forma más eficiente que pueda ser posible. Análisis técnico Por motivos de constantes cambios, XP plantea valores, para afrontar de alguna manera la solución más efectiva para el proyecto que se dirigen al equipo de trabajo de la forma que se detalla a continuación: • Comunicación: Todos

conformamos el equipo, debemos comunicarnos todos los días personalmente. Se hace el trabajo en conjunto en todo y para todo, comenzando con los requerimientos hasta la parte del código. • Sencillez: Todo aquello que fuese necesario y pedido debe ser realizado, sin más ni menos; de esta manera aumentará el valor estimulado para la inversión realizada hasta

aquella fecha. Se ejecutan los pasos por más pequeños que sean para lograr alcanzar el objetivo y minimizar las fallas en el transcurso del proyecto. Se crea algo da orgullo y que se mantiene a un costo razonable. • Retroalimentación: Se toma en cuenta de manera seria todos aquellos compromisos de iteración, realizando la entrega del software funcionando. El software es demostrado temprano, y en algunas ocasiones, se escucha fijamente para realizar las variaciones que fueran necesarias. Se habla del proyecto y se adapta el proceso necesario, nunca en sentido contrario. • Valentía: Se dice la verdad acerca del avance del proyecto, sus estimaciones y los progresos realizados. En la documentación no se incluyen los fracasos ya que se realiza una planificación para conseguir el éxito. Se adaptan cambios.

DELIBERACIÓN Y ELECCIÓN DE LA SOLUCIÓN

Tabla 11 Postulados del Manifiesto Ágil Postulados del Manifiesto Ágil Individuos e interacciones en lugar de herramientas y procesos Software de trabajo sobre documentación terminada Asistencia al cliente sobre la negociación del contrato Responder a variaciones sobre perseguir un plan Fuente: CITATION Cun01 \l 12298 (Cunningham, 2001) La aplicación de los criterios de evaluación de metodologías ágiles se muestra en la tabla 3 y la explicación se la realiza posteriormente. Tabla 22 Aplicación de criterios de evaluación a Scrum, Kanban y XP Postulado

Postulado 1 Postulado 2 Postulado 3 Postulado 4 Metodología 1.1 1.2 Total 2.1 2.2 Total 3.1 3.2 Total 4.1 4.2 Total Scrum 5 -1 4 5 -2 3 5 0 5 5 -2 3 Kanban 3 -2 1 3 -2 1 3 -2 1 4 -2 2 XP 5 -3 2 5 -2 3 5 0 5 5 -5 0 Fuente: Elaboración propia Postulado 1: Scrum y Kanban obtuvieron 5 en P1 ya que las dos metodologías valoran al individuo, establecen responsabilidades y roles, a diferencia que Scrum define actividades para cada iteración. Kanban por su parte obtuvo un total de 1. Postulado 2: Las 3 metodologías evaluadas requieren solo documentación requerida al inicio de cada iteración, con el detalle que Kanban genera entregable con testing cómodo al término de cada iteración, mientras las otras lo hacen constituido con el resto de las ocupaciones. Postulado 3: En Kanban el cliente es un integrante más del equipo, planifica iteraciones, responde consultas, mientras que en Scrum y XP colabora con pruebas y requerimientos además de que el contrato no contribuye valor al producto. Postulado 4: Tanto Scrum como XP permiten introducir cambios en la iteración en ejecución, además de no definir planificación alguna; mientras que Kanban está permitida la evolución mas no es recomendable mientras la iteración se encuentra en proceso. CAPÍTULO 4

DESARROLLO DE LA PROPUESTA TECNOLÓGICA TÍTULO Y DESCRIPCIÓN DE LA PROPUESTA TECNOLÓGICA “Scrum como metodología ágil para el desarrollo de software” Para la poner en marcha la alternativa propuesta, se decidió hacer uso de Genexus, una herramienta que a partir de las perspectivas del usuario encapsula el conocimiento y lo estructura en una base de conocimiento; esta herramienta tiene la facultad para diseñar, originar y sostener automáticamente la estructura de los programas de la aplicación y de la base de datos Al momento de desarrollar una aplicación en Genexus primero se realiza el Diseño haciendo un registro de las perspectivas del usuario, parte en la que el sistema encapsula y estructura el conocimiento, cumpliendo con el primer postulado del Manifiesto que indica que se debe valorar con mayor énfasis a los individuos y la interacción entre ellos por sobre los procesos y

herramientas. Luego se traslada a la fase de Prototipado en el que Genexus elabora la base de datos, con su estructura y sus datos, además de los programas necesarios para el ambiente del prototipo; y luego de generarse éste es enviado a realizar pruebas por los usuarios y el analista, punto en el que se cumple el segundo postulado que indica valorar al software principalmente por sobre una exhaustiva documentación. Si cuando están siendo ejecutadas las Pruebas se descubren errores o mejoras se regresa a la etapa de Diseño, para las modificaciones respectivas y se continúa con el Prototipo. A este ciclo se lo denomina Diseño/ Prototipo. Después de ser aprobado el Prototipo, viene la fase de Implementación, donde Genexus automáticamente produce los programas y base de datos. En resumidas palabras, el desarrollo empieza por el Diseño, para después hacer el Prototipo y ser Implementado o producido; resaltando que en cualquiera de las fases anteriores se puede retornar a la etapa de Diseño para modificaciones, cumpliendo con el cuarto postulado, donde se valora la respuesta ante un cambio más que a un seguimiento planificado. DESARROLLO EN DETALLE DE LA PROPUESTA TECNOLÓGICA Para la demostración del funcionamiento de esta herramienta, se usará la versión Genexus 15 Trial.

Figura 77 Genexus 15 Trial Fuente: Elaboración propia

En este caso se hará un ejemplo del desarrollo de una aplicación para una agencia de viajes, para lo cual se deberá desarrollar el Backend (aplicación interna que será usada solamente por el personal de dicha agencia para el registro de países, ciudades, y demás información de clientes, vuelos, etc.); y que en este caso se elegirá generar el código fuente en .NET; además de generar un módulo destinado para usuarios finales orientado a dispositivos móviles con el sistema Android; haciendo uso de Genexus Server para la administración de la aplicación, orientado al trabajo en equipo tal como lo menciona Scrum, siendo sus beneficios el contar con un respaldo de la información así como registro histórico de cambios y poder integrar de manera fácil a un nuevo desarrollador.

Figura 88 Ambiente de desarrollo integrado de Genexus Fuente: Elaboración propia Aquí se define el nombre de la base de conocimientos, en este caso será TravelAgency y se debe asignar la carpeta que contendrá dicha aplicación.

Figura 99 Creación de la base de conocimiento Fuente: Elaboración propia A continuación, se realiza el procedimiento de subir la aplicación al servidor mencionado anteriormente, Genexus Server, en la interfaz que se presenta a continuación.

Figura 1010 Conexión de la base de conocimiento con Genexus Server Fuente: Elaboración propia

Así, ha sido subido la base de conocimientos al servidor, luego de esto se debe crear los sujetos a los cuales el cliente se refiere cuando hace los requerimientos del proyecto, por lo tanto, se lo debe escuchar atentamente para no perder ningún detalle e información. En este caso, el cliente menciona cuatro objetos de la realidad que debemos describir en la base: clientes, atracciones turísticas, países y ciudades; para los cuales se creará un objeto Genexus de tipo transacción, para describir los objetos o actores de la realidad.

Se ha indicado que la metodología Scrum se basa en valorar al cliente y atender cada uno de sus requerimientos; es por eso que se elige esta herramienta, Genexus, ya que facilita el trabajo del desarrollador, pues luego de crear la base de conocimientos del proyecto automáticamente genera el formulario con el que se va a interactuar, de esta manera se disminuye tiempo en procesos y se dedica más atención a las necesidades del cliente.

Figura 1111 Generación automática del formulario Fuente: Elaboración propia

Conforme se va avanzando en el proyecto, se puede hacer verificaciones de los cambios que se ejecutan gracias a que se encuentra en el servidor de Genexus.

Figura 1212 Seguimiento del proyecto mediante Genexus Server Fuente: Elaboración propia

En Scrum se menciona que se pueden realizar modificaciones en el transcurso del ciclo de vida del proyecto, de tal manera que Genexus permite realizar variaciones sin alterar espacios en el tiempo o reducir el nivel de calidad del producto.

Figura 1313 Genexus permite realizar modificaciones en el software Fuente: Elaboración propia

Y para el desarrollo del módulo para usuarios finales mediante una aplicación Android, Genexus facilita traspasar el código ya ejecutado a un nuevo ambiente.

Figura 1414 Conversión de ambiente web a móvil en Genexus Fuente: Elaboración propia

CAPÍTULO 5

ANÁLISIS TÉCNICO ECONÓMICO DE LA PROPUESTA TECNOLÓGICA Para que la metodología Scrum sea usada mediante Genexus, es necesario invertir los valores estimados que se muestran en la siguiente tabla: Tabla 33 Análisis de costos anual Recursos Humanos

Salario Anual Scrum Master 1740 Product Owner 2050 Licencias de Software Total Genexus 460 Recursos Financieros Total Anual 2 Laptops 2400 Internet 240 Gastos de transporte y viáticos 1000 Gastos administrativos Total Materiales de oficina 110 Otros gastos 150 TOTAL INVERSIÓN 8150 Fuente: Elaboración propia Cabe aclarar que los costos presentados en este trabajo son estimados, además resulta más factible ya que la herramienta propuesta cumple las funciones que deberían ser asignadas a más recursos humanos.

CONCLUSIONES

En un mundo en el que existen constantes cambios en el ámbito tecnológico es necesario hacer conciencia que la realidad de las empresas se torna mucho más compleja, por lo tanto, se necesita contar con efectivas soluciones que permitan a la empresa desarrollar aplicaciones con mayor flexibilidad y que estén mejor preparadas para los cambios que puedan surgir, por eso: • La metodología Scrum es la más apropiada para el desarrollo de software, puesto que los enfoques son aquellos que determinan las prioridades exactas para que el proyecto genere un producto de calidad. • Genexus es una herramienta efectiva para ser combinada con la metodología antes mencionada, ya que comparten las filosofías y

estiman al cliente como la parte más importante del proyecto. • Los valores y principios que rigen a cada una de las metodologías ágiles deben ser aplicados para obtener resultados positivos.

RECOMENDACIONES

• Seguir los principios reflejados en el Manifiesto Ágil • Cumplir con los valores de Scrum • Verificar que las funciones desempeñadas por los integrantes del equipo sean las correctas • Ejecutar pruebas cada vez que se realicen cambios en el producto

hdphoto1.wdp

hdphoto2.wdp

Hit and source - focused comparison, Side by Side:

Left side: As student entered the text in the submitted document.
Right side: As the text appears in the source.
