CALIDAD DE SOFTWARE Y PARADIGMAS ÁGILES

AUTORES:
Copras Maricella C.I. 3-118-287
Noris Lay C.I.8-717-2386
Eric Sánchez 2-129-762

Panamá, Julio 2019

INTRODUCCIÓN

El proceso evolutivo de los sistemas informáticos, involucra la capacidad del talento humano de crear, ingeniar y actualizar de forma permanente la parte blanda de los dispositivos respectivos, denominada software asociada a los lenguajes de programación, lo cual va permitiendo dar dinámica o flujo permanente a las funciones del Hardware o parte dura de tales recursos, requiriendo mayor posibilidades de competencias, capacidades y respuestas del Human Hardware, en su proceso evolutivo interactivo con tales sistemas, la modernización implica asumir metódicas ágiles cuya capacidad transformacional potencie sistemas operativos.

Así el contenido del ensayo intitulado calidad de software y paradigmas agiles, supone la creación y redefinición ingeniosa de programas de lenguajes informáticos superatorios de capacidades de respuestas tradicionales, al asumirse incorporación de propiedades cuya posibilidad de respuesta optimizan los sistemas de operaciones en equipos permitiendo una más fácil disposición a talentos humanos en el manejo de sus significantes bondades, por tanto de seguidas debe apuntalarse la consistencia y fin dinámico de estos importantes herramientas y recursos.

De seguidas se presenta, la relación de elementos referentes al software, ingenieria de software, desarrollo y arquitectura del mismo partiendo de la continuada superación del tradicionalismo tecnológico, asumiéndose el software ágil, como forma transformadora referente a un conjunto de principios significativos orientados hacia el fortalecimiento de referentes técnicos y de carácter operativo, por último se configura un estudio comparativo de tecnologías tradicionales y ágiles, sumándose referentes de evolutividad y tangencialidad conforme a las posibilidades permanentes de lograr configuración de productos y servicios a partir de un software garantista de mayor confort y ergonomía.

CALIDAD DE SOFTWARE Y PARADIGMAS ÁGILES

. El desarrollo ingenioso y perfectibilización de los sistemas informáticos, requiere un permanente mejoramiento, reingenio y

actualización de las propiedades del software permitiendo contribuir en la superación de la capacidad de respuesta en equipos de más alta definición operativa, del significado del mismo O Regan (2011:16) define al software como programas y como estructuración de datos, se entiende así una sumatoria de licencias y códigos cuya integración permite dar optimidad permanente, actualización y mejoramiento de las capacidades de respuestas al complejo dinámico del Human Hardware.

En relación a la capacidad de creación, reingenio y transformación, se cuenta con la denominada ingenieria del software, de su alcance Sommerville (2011: 3) alude parte de un elemento importante: el software, que además de ser el programa, se necesita para hacer que este programa opere de manera correcta, ciertamente los linguismos informáticos, deben transformarse, creando nuevas bandas operativas, estas asumen atributos más dinámicos y efectivos en el manejo de dispositivos, al punto de permitir la proliferación de herramientas y recursos de nueva generación facilitando funciones y dando más óptima capacidad de respuesta a las sociedades digitales emergentes.

Al respecto del particular, Boehm, (2006), indica de tendencias en ingeniería del software, muestra que el mercado actual está caracterizado por el desarrollo rápido de aplicaciones y la reducción de la vida de los productos, por tanto es evidente se acrecientan los software y se desgastan los hardware por tanto deben crearse mecanismos orientados a establecer compensaciones de procesos al reingeniar funciones para satisfacer exigencias evolutivas del particular.

De manera que, se deben superar los atributos y capacidades del software más aún para responder a procesos de mercados de alta definición y de mayor competitividad, asumiéndose como posibles logros rentabilidad funcional y operativa, al punto de configurar sistemas operativos, expuestos a procesos de investigación creativos y evolutivos para fomentar respuestas más inmediatas a una autopista digital cambiante

En atención al proceso de desarrollo de software, Carvajal (2008) lo define como el

conjunto de actividades que se llevan a cabo para construir un producto software, agregando que las metodologías permitirán al proceso marcar un camino común para producir software con éxito. Igualmente, se debe considerar viabilidad, flexibilidad y posibilidad abierta de transformación, disponiéndose ciertamente configurar lenguajes de programación más potenciados, y orientados efectivamente a un éxito en el logro concreto de respuestas inmediatas e integrales a una sociedad de información y emprendimiento transformadora y reinventora, en una interacción digital continuada.

Ahora bien la ingeniería y el desarrollo del software debe compaginarse a la materialidad de una estructura denominada arquitectura, de su naturaleza Fernández (2006: 40) alude parte de tres componentes: elementos, forma y razón, ciertamente este debe condensarse de factores de orden integral, códigos, lenguajes, perspectivas y lógicas secuenciales fortaleciendo el uso dinámico de tradicionalismos y la creatividad ingeniosa en superar tales componentes, con recursos de más óptima aplicación y rendimiento en procesos globalizantes hacia una universalización integradora de dispositivos y sistemas más transformadores y readaptivos a pluralismos técnicos y socioculturales estructurantes.

Por su parte, Bass, Clements y Kazman (2003: 43) definen tal arquitectura partiendo de la importancia que tiene, debido a que permite la comunicación entre las personas involucradas en la construcción del producto y así obtener una comprensión adecuada, además permite tomar las decisiones más convenientes en el diseño, la implementación y el mantenimiento del ciclo de vida del software después de un análisis y un estudio, supone esto un proceso científico de permanente investigación del arcaísmo tecnológico, de las modas actuales operativas y la predefinición de elementos estructurales para redefinir propiedades y concertar nuevos referentes hacia el tecnologicismo informático postmoderno.

Ahora bien, la dinámica socio cultural y el propio proceso de revisión y transformación de necesidades y medios para satisfacerlas, trascienden en una dialéctica superatoria, focalizada en el

reconocimiento de tecnologías tradicionales, comportables de lenguajes programáticos con límites de recursos y herramientas de acción, asumiéndose la imperiosa necesidad de competir y trascender en mercados globales mediante la supuración y perfilamiento de tecnología ágil postmoderna, cuyos atributos supongan atender culturas y procesos de más exigente mecánica de acción orientados a los umbrales del primer mundo rentable y tecnológico.

De esta manera, surge la dinámica del denominado Software Ágil, de su sustrato significante Marco et al. (2010: 194), describe cuando es más importante la interacción entre los miembros que la utilización de técnicas de desarrollo; la construcción de un software útil y de calidad que una documentación adecuada de un sistema; la contribución con el cliente que la realización del contrato; la capacidad de cambio según se presente cualquier situación que el seguimiento de planificación.

Por tanto, los cambios estructurales y funcionales al paso del tiempo, han significado ciertamente dar ocasión a la creación de medios, recursos y herramientas, cuyas propiedades puedan significar facilitar y simplificar la satisfacción de requerimientos en una cultura de uso y aprovechamiento de tecnología electrónica más exigente y cambiante, esto con el fin de fomentar y fortalecer la inclusividad de habitantes cibernautas a una sociedad de información, conocimiento y emprendimiento de más complejas relaciones, en los cuales la eficiencia, efectividad y facilismo de medios consolidan patrones superatorios cada vez más definidos.

Al respecto de los particulares del software ágil, Beck et al. (2012: 1), considera este se encuentra compuesto por 12 principios:

1. Satisfacer al cliente mediante la entrega temprana y continua de software con valor, esto implica crear y disponer lenguajes y programas de inmediata actualización, reingeniable y reinventable ante urgentes necesidades de respuesta a usuarios exigentes y en procesos emergentes.

2. Aceptar que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente, efectivamente se requiere ingeniar

programas cuyos gratificantes resultados permitan mejorar satisfacción de clientes, cuando se optimizan gestiones operativas y acrecientan funciones y aplicaciones más modernas.

3. Entregar software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible, la inmediatez, la eficiencia y la oportunidad caracterizan este tipo de medios, al punto de crear programas más suficientes de superación y perfectibilidad continuada.

4. Los responsables de negocio y los desarrolladores deben trabajar juntos de forma cotidiana durante todo el proyecto, se requieren fomentar estudios técnicos y de mercado, con el fin de complementar estrategias de invención acordes a necesidades y expectativas de posibles y potenciales usuarios.

5. Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo, por ende, los patrones de culturismo tecnológico predominantes pueden inferir en el fortalecimiento de medios cuya inmediatez y eficiencia de respuesta permitan generar modos más avanzados en configuración y capacidad de acción.

6. El método eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara, por tanto implica dialógicas convergentes que permiten superar diagnósticos y asociaciones de patrones para coincidir en configurar medios avanzados hacia perfiles de vida más confortables.

7. El software funcionando es la medida principal de progreso, así mismo, reingeniar, reinventar, componer nuevos códigos, bandas de acción y de eficiente alcance puede permitir sobrepasar un primitivismo reformista a un evolutivismo modernista de alto contenido versátil y adecuatorio.

8. Los procesos ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios deben ser capaces de mantener un ritmo constante de forma diseño para mejorar la agilidad, por tanto mientras más capacidad cuente el talento humano creador y más necesidad de acción requieren individuos y organizaciones se podrá justificar trascender fronteras.

9. La atención continua a la excelencia técnica y al buen diseño mejora la agilidad, significa mientras más diestro sea

el creador de software y más convincente sea el público usuario de las propiedades de productos movidos por programas sofisticados será probable atender modernismos globales cambiantes e inclusivos.

10. La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial, por tanto supone crearse mecánicas simplificadoras y eficientes ante las urgencias organizacionales, dándose formas más ergonómicas y factibles de atender con mayor refinamiento necesidades, gustos y preferencias universales significativos.

11. Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados, así las entidades eficientes, crearán inteligencia artificial electrónica de más capacidad generativa, por tanto, mientras evolucionen medios y programas se tendrán garantismos de ascender civilizaciones a modelos globales referentes.

12. A intervalos regulares el equipo reflexiona sobre cómo ser efectivo para luego perfeccionar su comportamiento, e modo que, la capacidad flexible de reingeniar y profundizar el alcance de los programas podrá permitir establecer una cultura y capacidad de respuestas equiparadas en lo evolutivo, procurando cerrar brechas de simetrías entre culturas interactivas en procesos globales postdinamicos.

Según Canós, Letelier y Penadés (2003: 1), algunos métodos para construcción de software ágil son: extreme Programming, SCRUM, Crystal Methodologies, Dynamic Systems Development Method, Adaptive Software Development, Feature –Driven Development, LeanDevelopment, entre otros, por tanto existen medios técnicos disponibles cuyas herramientas y recursos permiten crear y actualizar de forma permanente las propiedades de software ágil, en general supone apuntalar programas de desarrollo, cuya adaptación y adecuación responde a entes empresariales y organizaciones con medios de investigación y evolución referente.

Por lo expuesto construir software ágil, supondrá fomentar desarrollo de estrategias de laboratorios, cuyas experiencias, prácticas y experticias se traducen en crear movilidades y dinámicas de programaciones orientadas a establecer

equipos y recursos más óptimos, esos que, al superar trivialidades, permitan aflorar referentes culturales digitales inclusivos hacia procesos de definición emergente y focalizada hacia lo multidimensional.

Por último, se da una comparación básica entre metodologías ágiles y tradicionales destacados.

según puede verse en la Tabla 1 (Canós et al 2003). Estas diferencias no solo afectan al proceso de desarrollo de software, sino también al contexto del equipo y a la organización, siendo referentes

Metodologías Ágiles	**Metodologías Tradicionales**
Basadas en heurísticas provenientes de prácticas de producción de código	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo
Especialmente preparados para cambios durante el proyecto	Cierta resistencia a los cambios
Impuestas internamente (por el equipo)	Impuestas externamente
Proceso menos controlado, con pocos principios	Proceso mucho más controlado, con numerosas políticas/normas
No existe contrato tradicional o al menos es bastante flexible	Existe un contrato prefijado
El cliente es parte del equipo de desarrollo	El cliente interactúa con el equipo de desarrollo mediante reuniones
Grupos pequeños (<10 integrantes) y trabajando en el mismo sitio	Grupos grandes y posiblemente distribuidos
Pocos artefactos Más artefactos	Pocos roles Más roles
Menos énfasis en la arquitectura del software	La arquitectura del software es esencial y se expresa mediante modelos

Tabla 1. Diferencias entre metodologías ágiles y metodologías tradicionales.

En la referencia tratada las metodologías ágiles, suponen procesos relacionados a la naturaleza de equipos y de sus programas, admiten por tanto establecer la superación de propiedades atendiendo a posibilidades de ingeniería superatoria, más que un contexto social, asumiéndose por cierto flexibilidad e ingenio en cambios evolutivos.

Por lo tanto las tecnologías tradicionales no suponen crear invenciones sino adecuaciones a espectros temporales y espaciales, lo ágil supone disponer de recursos cuya óptima reingeniación y gama funcional de respuestas se presenta evolutiva permitiendo superar referentes acordes a las propias líneas de acciones vanguardistas y significativas

Conclusiones

El contenido de la producción, se vincula con el estudio de la calidad de software y paradigmas ágiles, asumiéndose la presentación de programas y datos dándose en un proceso continuado de superación, perfectibilidad y optimización de atributos con reingeniable y reinventable carácter, esto con el fin de atender nuevos referentes de ordenación y transformación cientista y de orden estructural funcional.

Se apuntala el contenido del software asumiéndose ciertamente el conjunto de datos y programas significando nuevas formas de operativización de recursos, técnicas y metódicas informáticas electrónicas orientadas a gestar dinámicas de medios con mayor ergonomía y confortabilidad ante exigencias y superaciones de referentes culturales.

En el curso de la ingeniería, desarrollo y arquitectura de software se permiten establecer lineamientos técnicos eficientes cuyos contenidos puedan conducir a superar programas previos orientados a fomentar nuevas ingeniosas mecánicas de funcionalidad de dispositivos especiales, con el valor agregado de transformar medios tradicionales para ordenación de sistemas e interacciones electrónicas.

Por lo expuesto, el software ágil, implica ciertamente crear códigos, lenguajes y programas óptimos, cuya facilidad, versatilidad y eficiencia de respuesta pueda significar atender demandas de usos de servicios acordes a nuevas y complejas realidades sumándose efectivamente medios y dispositivos

cuya ergonómica reingenieria permita transformar epicentros de acción local.

En igual orden, el software ágil, supone integrar principios ciertos e integrados cuyos contenidos permitan atender la incorporación e inclusividad de diversas necesidades y posibilidades de interacción tecnológica contribuyendo esto a sedimentar cierre de brechas y asimetrías en ese orden dentro de la universal sociedad digital.

Por último, el contraste entre tecnología tradicional y ágil, permite comprender procesos transformacionales orientados a superar mecánicas de contextos hacia procesos de superación de los atributos de orden técnico propios del dispositivo electrónico, su programación y gradual actualización según metódicas de acción evolutiva.

Referencias Bibliográficas

Bass, L; Clements, P; & Kazman, R. (2003). Software Architecture in Practice. 2 ed. Boston (EUA): Addison Wesley. 560 p. ISBN: 0-321-15495-9.

Beck, K., et al. (2012). The Agile Manifesto. Manifesto for Agile Software Development. *En*: http://www.agilemanifesto.org

Boehm, B. (2006). A View of 20th and 21st Century Software Engineering. In: 28th international conference on Software engineering, pp. 12--29. Shanghai, China.

Canós, J., Letelier, P., Penadés, M. (2003). Metodologías Ágiles en el Desarrollo de Software.
JISBD 2003. España.

Carvajal, J. (2008). Metodologías Ágiles: Herramientas y Modelo de desarrollo para aplicaciones Java EE como metodología empresarial.

Fernández, L. (2006). Arquitectura de Software. En: Software Guru, Vol. 2, No 3 (may jun). México DF (México). p. 40-45. ISSN: 1870-0888.

Marco, M; Marco, J; Blázquez, J & Segret, R. (2010). Escaneando la Informática. Barcelona (España): Editorial UOC. 260 p. ISBN: 978-8497881104.

O›Regan, G. (2011). Introduction to Software Process Improvement. London (UK): Springer. 266 p. ISBN: 978-0857291714.

Sommerville, I. (2011). Ingeniería de software. 7 ed. México D.F. (México): Person Education. 712 p. ISBN: 8478290745.