SISTEMAS DINÁMICOS

Dr. Francisco Javier Wong Cabanillas
Editor & Compilador

2018

SISTEMAS DINÁMICOS 2018

SISTEMAS DINÁMICOS 2018

Dr. Francisco Javier Wong Cabanillas
EDITOR & COMPILADOR

Sistemas Dinámicos

Editor: Dr. Francisco Javier Wong Cabanillas

Dirección: Av. El Retablo 808 2do. Piso Urb. El Retablo, Comas. Lima-Perú

Correo electrónico: fjavierwongc@yahoo.es

Compilador: Dr. Francisco Javier Wong Cabanillas

Diseño y Redacción: Bach. Carlos Alberto Vega Vidal

ISBN: 978-612-00-4024-9

Primera edición digital: diciembre 2018

Libro electrónico disponible en: http://ctscafe.pe

Criterios de selección de metodologías de desarrollo de software

Mg. Pedro Pablo Rosales Lopez
Universidad Nacional de San Martín
Correo Electrónico: pprosalesl@gmail.com

Mg Julio Alejandro Salas Bacalla
Universidad Nacional de San Martín

Dr. Oscar Rafael Tinoco Gómez
Universidad Nacional de San Martín
Correo Electrónico: otinocog@gmail.com

Resumen: El desarrollo de software no es una tarea sencilla, por mucho tiempo esta labor se ha llevado adelante sin una metodología definida. Al respecto, algunos autores definen una metodología como una colección de procedimientos, técnicas, herramientas y documentos auxiliares que ayudan a los desarrolladores de software en sus esfuerzos por implementar nuevos sistemas de información.
En las dos últimas décadas, respecto a estas metodologías de desarrollo de software se ha entablado un intenso debate entre dos grandes corrientes. Por un lado, las denominadas metodologías tradicionales, centradas en el control del proceso, con un riguroso seguimiento de las actividades involucradas en ellas. Por otro lado, las metodologías ágiles, centradas en el factor humano, en la colaboración y participación del cliente en el proceso de desarrollo y a un incesante incremento de software con iteraciones muy cortas.
El artículo presenta una propuesta, en medio de este debate, para seleccionar una metodología de desarrollo de software.

Palabras claves: Selección/ Metodologías/ Desarrollo/ Software

Abstract: In the last two decades, on these software development methodologies has been engaged in intense debate between two currents. On the one hand, the so-called traditional methods, focusing on process control, with rigorous monitoring of the activities involved in them. On the other hand, agile methods, focusing on human factors in collaboration and

customer participation in the development process and a relentless increase in software with very short iterations.
The article presents a proposal, in the midst of this debate, to select a software development methodology.

Keywords: Selection/ Methodologies/ Development/ Software

1. Introducción

Maida y Pacienzia (2015) resaltan el carácter dinámico de los cambios en la industria del software, particularmente relacionado con las metodologías de desarrollo de software. Destacan, además, la vigencia de la clasificación de estas en robustas y ágiles.
Avison y Fitzgerald (1995) presentan una definición de las metodologías de desarrollo muy clara, destacando sus principales componentes, fases, herramientas y técnicas. "Una metodología es una colección de procedimientos, técnicas, herramientas y documentos auxiliares que ayudan a los desarrolladores de software en sus esfuerzos por implementar nuevos sistemas de información. Una metodología esta formada por fases, cada una de las cuales se puede dividir en sub-fases, que guiarán a los desarrolladores de sistemas a elegir las técnicas más apropiadas en cada momento del proyecto y también a planificarlo, gestionarlo, controlarlo y evaluarlo".

1.1. Modelo de proceso

Según Darniame (1999), un modelo de procesos es una representación del mundo real, que captura el estado de actual de las actividades para guiar, reforzar o automatizar partes de la producción de los procesos.
En el artículo de E. Georgiadou Software Process and Product Improvement: A Historical Perspective, se reseñan los modelos; V-Model, WModel, X-Model, RAD y Orientado a Objetos, sin embargo los más conocidos son:

- **Modelo secuencial**. Representado por metodologías tan famosas como Waterfall. Se inicia con un completo análisis de los requisitos de los usuarios. En el siguiente paso, los programadores implementan el diseño y finalmente, el completado y perfecto sistema es probado y enviado.
- **Desarrollo incremental**. Su principal objetivo es reducir el tiempo de desarrollo, dividiendo el proyecto en intervalos incrementales superpuestos. Del mismo modo que con el modelo waterfall, todos los requisitos se analizan antes de empezar a desarrollar, sin embargo, los requisitos se dividen en "incrementos" independientemente funcionales.
- **Desarrollo iterativo**. A diferencia del modelo incremental se centra más en capturar mejor los requisitos cambiantes y la gestión de los riesgos. En el desarrollo iterativo se rompe el proyecto en iteraciones de diferente longitud, cada una de ellas produciendo un producto completo y entregable.

1.2.Modelo en espiral. Comprende las mejores características de ciclo de vida clásico y el prototipado (desarrollo iterativo). Además, incluye el análisis de alternativas, identificación y reducción de riesgos.

2. Material y métodos

Dijkstra, con un influyente artículo (Go to statement considered harmful), sienta las bases para la creación de las metodologías, como se conocen actualmenter.

Entre otros autores se establecieron unos criterios que marcasen el éxito del desarrollo del software, que hasta ahora están vigentes:

- El coste del desarrollo inicial debe ser relativamente bajo.
- El software debe ser fácil de mantener.
- El software debe de ser portable a nuevo hardware.
- El software debe hacer lo que el cliente quiere.

A partir de que Dijkstra planteará su programación estructurada (a través de su libro A Discipline of Programming), empezaron a surgir lenguajes de programación influenciados por este, que respetaban los siguientes criterios:

- Desarrollo de programas mediante top-down, en contraposición a bottom-up.
- Utilizar un conjunto específico de reglas de programación (Prohibido el uso de Go To).
- Seguir un conjunto de pasos formales para descomponer los problemas grandes (divide y vencerás).

2.1. Metodologías tradicionales y Metodologías ágiles

Diversos autores coinciden en señalar algunos re- quisitos que deben tener las metodologías de de- sarrollo:

- Visión del producto.
- Vinculación con el cliente.
- Establecer un modelo de ciclo de vida.
- Gestión de los requisitos.
- Plan de desarrollo.
- Integración del proyecto.
- Medidas de progreso del proyecto.
- Métricas para evaluar la calidad.
- Maneras de medir el riesgo.
- Como gestionar los cambios.
- Establecer una línea de meta.

En tiempos recientes, han surgido las metodologías ágiles, como una alternativa, una reacción a las metodologías tradicionales y principalmente a su burocracia. Brooks, en su mítico libro *The Mythi- cal Man Month*, expone las primeras ideas que se plantean en las metodologías ágiles, gran parte de ellas, responden al sentido común.
Canós, J. (2005) resume las características de ambas metodologías, en la siguiente tabla:

Tabla N° 01. Comparación de metodologías.

Metodologías ágiles	**Metodologías tradicionales**
Se basan en heurísticas provenientes de prácticas de producción de código	Se basan en normas provenientes de estándares seguidos por el entorno de desarrollo
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 muy controlado, numerosas normas
Contrato flexible e incluso inexistente	Contrato prefijado
El cliente es parte del desarrollo	Cliente interactúa con el equipo de desarrollo mediante reuniones

Grupos pequeños (<10)	Grupos grandes
Pocos artefactos	Más artefactos
Menor énfasis en la arquitectura del software	La arquitectura del software es esencial

Fuente: Canós, J *et al*, 2005. Metodologías Ágiles.

Entre las metodologías ágiles, se puede mencionar a las siguientes:

- Adaptative Software Development
- Agile Modeling
- Agile Model Driven Development
- Agile Project Management
- Agile Unified Process
- Crystal Methods
- Dynamic Systems development methods
- Feature driven development
- Internet Speed Development
- Lean development
- Pragmatic programming
- Scrum
- Test Driven Development
- XBreed
- Extreme Programming
- Win Win Spiral
- Evolutionary Project Management
- Story cards driven development
- Agile Unified Process
- Open Unified Process

2.2. Selección de metodologías ágiles, por criterios de presencia

Los diseñadores de software tienen interés de trabajar con metodologías lo suficientemente documentadas, que nos faciliten la obtención de información, pero también es interesante trabajar con metodologías que dispongan de algún tipo de certificación y training. Según estas condiciones, se han determinado seis clasificaciones que permiten seleccionar una metodología, según se encuentran mejor posicionadas, en el acumulado final.
Las clasificaciones son:

- La metodología con mayor presencia en Internet.
- La metodología mejor documentada.
- Metodologías certificadas y con training.
- Metodologías con comunidades.
- Metodología más utilizada por empresas. Presencia empresarial.
- Metodología más utilizada en proyectos software.

Se considera como metodologías **certificadas** aquellas que emiten un certificado que aseguran el cumplimiento y seguimiento de la metodología, así como sus técnicas y prácticas.
Una metodología dispone de **training**, si se en- cuentra alguna institución, organización o compañía que ofrezca formación de la metodología.
Se considera que una metodología tiene **comunidad**, contemplando si se ha formado una comunidad relevante o si está asociada a la Agile Alliance, soportándola y cumpliendo sus principios.

Se consideran los proyectos realizados, en su mayoría por metodologías que se han aplicado en empresas privadas y, por tal motive, no existe mucha documentación pública al respecto. Por lo tanto, determinar esta clasificación, requiere de una búsqueda exhaustiva.

2.3. Selección de metodologías

Este aspecto no ha sido tratado de manera adecuada, sobre todo en el ámbito de las metodologías tradicionales, y en el caso de las ágiles no existe un criterio unificado. Por ello, el presente artículo se orienta a la formulación inicial, en base a la información existente a la fecha y a la experiencia personal, a la formulación de dos procedimientos al respecto: selección por criterios de presencia y por conocimiento.

2.4. Aplicación del criterio de selección por presencia

A un grupo de programadores profesionales en el medio local (10), se le ha aplicado una encuesta, sobre recordación, conocimiento y uso de metodologías, quedando un grupo de 5 metodologías, que se han evaluado, según este criterio de selección.
Para determinar la presencia, de las metodologías en Internet, se han realizado búsquedas en Google, Yahoo y Live. Sobre el resultado, se asignaron 5 puntos al mayor, y 1 punto al menor.
Para determinar las metodologías de mayor documentación, se han considerado como documentos, los Libros en español, Libros en inglés y Papers que hablen sobre la aplicación de la metodología. Siguiendo el mismo método, se asignó 5 puntos al mayor y 1 punto al menor.
En el caso de la Certificación y Training, se ha buscado si hay instituciones que certifican la implementación de la metodología, así como si hay entrenamiento o capacitación en la misma. Como no es posible hacer diferencias en cuanto a la certificación, habiéndose asignado el mismo puntaje a las metodologías que tienen Certificación y Training (5 puntos) y 3 puntos las metodologías que contienen sólo training.
En cuanto a Comunidades, la mayoría pertenece a la Agile alliance, pero hay algunas que tienen sus propias comunidades, alianzas e intensa actividad a su alrededor. A estas metodologías se le asignaron 5 puntos, porque no es posible diferenciar entre estas comunidades el número de miembro. A las metodologías que solo pertenecen a la Agile alliance, se le asignaron 2 puntos.
En cuanto a proyectos de software y presencia empresarial, se han asignado 5 puntos, a la metodología que presenta más proyectos y un punto a la que presenta menos..

3. Resultado de la aplicación del criterio de selección por presencia

Se ha preparado un cuadro resumen con los resultados de la selección. Para cada metodología evaluada, se ha colocado la puntuación que se ha obtenido de la clasificación. La sumatoria de cada clasificación determina que Scrum, es la metodología que se debería usar, por tener una mejor puntuación.

Tabla N° 2

Metodología	Mayor presencia en Internet	Mejor documentación	Certificadas y con training	Comunidades	Presencia empresarial	Proyectos de software	Total
Agile Project Management (APM)	2	1	3	5	1	1	11
Dynamic Systems development methods (DSDM)	1	3	5	5	4	4	22
Scrum	5	2	5	5	5	5	27
Test Driven Development	3	4	3	2	2	2	16
Extreme Programming (XP)	4	5	3	2	3	3	19
Total	15	15	19	19	15	15	95

Fuente: Elaboración propia

3.1. Selección de metodología, por criterios de conocimientos

En función del grupo de trabajo o de diseño, se consideran los siguientes criterios en función de los conocimientos que el equipo de desarrollo tenga de las metodologías a evaluar. Estos criterios son:

- Grado de conocimiento
- Soporte orientado a objetos
- Adaptable a cambios
- Basado en casos de uso
- Posee documentación adecuada
- Facilita la integración entre las etapas de desarrollo
- Relación con UML
- Permite desarrollo software sobre cualquier tecnología

En función de los conocimientos que el equipo tenga, se establecen los pesos para cada criterio. Por ejemplo, Ríos y Suntaxi [37], en su tesis de grado, proponen las siguientes tablas de pesos:

- 20% para el Grado de conocimiento
- 15% para Adaptable a cambios y Posee documentación adecuada
- 10% para el resto de criterios

Para determinar la metodología a usar, Ríos y Suntaxi, han elaborado un cuadro resumen, evaluando las siguientes metodologías:

- RUP, Rational Unified Process
- MSF, Microsoft Solution Framework
- RAD, Rapid Application Development
- XP, Extreme Programming

En esta evaluación, la metodología RUP es la que recibe un mayor puntaje por parte del equipo de desarrollo.

Tabla N° 03

Criterio	%	RUP	MSF	RAD	XP	Total
Grado de conocimiento	20	15	10	10	10	45
Soporte orientado a objetos	10	10	10	10	10	40
Adaptable a cambios	15	10	15	10	15	50
Basado en casos de uso	10	10	5	10	5	30
Posee documentación adecuada	15	15	15	15	10	55
Facilita la integración entre las etapas de desarrollo	10	10	10	10	10	40
Relación con UML	10	10	8	8	8	34
Permite desarrollo software sobre cualquier tecnología	10	10	10	10	10	40
Total	100	90	83	83	78	

Fuente: Elaboración propia

4. Conclusiones

En algunas tesis revisadas, el criterio de selección de la metodología, es bastante subjetivo. Por ello se ha buscado de establecer una escala ordinal de valoración de la importancia de los criterios de se- lección, hecho que contribuye a una mejora en el proceso de selección pero que no desaparece el nivel de subjetividad inherente al proceso

En tal sentido, los autores se comprometen, en un próximo artículo, a enfocar el problema desde el punto de vista del proceso analítico jerárquico, metodología elaborada por Saaty.

6. Literatura Citada

Avison, D. and G. Fitzgerald, (1995). I*nforma- tion Systems Development: Methodologies, Techniques, and Tools.* McGraw-Hill.

Brooks, Fred (1995). *The Mythical Man-Month: Essays on Software Engineering, Anniversary Edition.* Ed. Addison Wesley. 2da edition.

Canós, Joseph (2005). *Métodologías Ágiles en el Desarrollo de Software.* Universidad Politéc- nica de Valencia.

Derniame, J (1999). *Software Process: Prin- ciples, Methodology, Technology.* Lecture Notes in Computer Science 1500 Springer 1999, ISBN 3-540-65516-6 BibTeX

Dijkstra E (1976). *A Discipline of Programming.*Prentice Hall, Universidad de Texas.

Georgiadou, E. (2003) "Software Proces and Product Improvement: A Historical Perspective". *Cybernetics and Systems Analysis.* Vol. 39, N.° 1 2003.

Maida, Esteban & Pacienzia, Julian (2015) Metodología de desarrollo de software. Tesis de Licenciatura Universidad Católica Argentina

Ríos, Edgar y Wilson Suntaxi (2008). Desarrollo de un sistema informático para los procesos de cosecha y post cosecha de la camaronera Pampas de Cayanca. Tesis de grado, Facultad de ingeniería de sistemas, Escuela Politécnica Nacional, Quito.