

**UNIVERSIDAD POLITÉCNICA DE MADRID**

**Escuela Técnica Superior de Ingeniería de Sistemas Informáticos**

Máster Universitario en Ingeniería Web

Trabajo Fin de Máster

Estado del arte de las metodologías de desarrollo ágiles

**Autor**

**Manuel Puchades Rodríguez**

**Tutor**

**Luis Fernández Muñoz**

**julio de 2019**

## 

## Agradecimientos

Gracias a …

## Resumen

Amparadas bajo el manifiesto ágil, cohabitan diferentes herramientas y prácticas para la gestión y elaboración de proyectos de software. Estas son conocidas como metodologías ágiles. Sin embargo, el término ágil ha sufrido un gran desgaste en los últimos años y varios de los creadores y firmantes del manifiesto sostienen que los principios y valores originales que llevaron a su redacción se han diluido con el tiempo en los detalles de las implementaciones.

En el presente proyecto exploraremos la historia de los preceptos ágiles, estudiaremos las metodologías más extendidas, y cómo ha evolucionado el concepto de agilidad a través de estas hasta nuestros días.

Palabras clave

manifiesto ágil, metodologías ágiles, SCRUM, XP, Lean, Kanban.

## Abstract

*[El resumen anterior en inglés]*

Keywords

*[Palabras clave relacionadas con el trabajo]*

## Tabla de Contenidos

Contents

[Agradecimientos 4](#_Toc10391107)

[Resumen 6](#_Toc10391108)

[Palabras clave 6](#_Toc10391109)

[Abstract 8](#_Toc10391110)

[Keywords 8](#_Toc10391111)

[Tabla de Contenidos 10](#_Toc10391112)

[Objetivos 1](#_Toc10391113)

[Capítulo 1. Introducción 3](#_Toc10391114)

[Capítulo 2. Entendiendo la agilidad 5](#_Toc10391115)

[1. Dynamic Systems Development Method (DSDM) 5](#_Toc10391116)

[2. Adaptive Software Development 5](#_Toc10391117)

[3. Scrum 5](#_Toc10391118)

[4. Crystal 5](#_Toc10391119)

[5. Extreme Programming (XP) 5](#_Toc10391120)

[6. Pragmatic programming 5](#_Toc10391121)

[7. Lean and Kanban 5](#_Toc10391122)

[8. El manifiesto ágil 5](#_Toc10391123)

[Capítulo 3. El discurso Agile 7](#_Toc10391124)

[1. Catastrofismo y la falsa dicotomía: Ágil o Cascada 7](#_Toc10391125)

[2. Un conjunto de casos de éxito, pocas estadísticas reales. Confundiendo causalidad y casualidad. 10](#_Toc10391126)

[3. Todo o nada. No estas siendo lo suficientemente ágil 10](#_Toc10391127)

[Capítulo 4. La agilidad hoy 12](#_Toc10391128)

[1. Cómo Agile se convirtió en sustantivo y perdió sus valores con el paso del tiempo. 12](#_Toc10391129)

[2. Hecho para desarrolladores impuesto por las empresas. 14](#_Toc10391130)

[3. El problema de las certificaciones 16](#_Toc10391131)

[4. La agilidad, ese gran desconocido: prácticas vs. Principios 18](#_Toc10391132)

[Conclusiones y Posibles Ampliaciones 20](#_Toc10391133)

[1. Conclusiones 20](#_Toc10391134)

[2. Lineas futuras 21](#_Toc10391135)

[**Referencias** 24](#_Toc10391136)

[Anexo I 26](#_Toc10391137)

## Objetivos

## Introducción

## Entendiendo la agilidad

### Dynamic Systems Development Method (DSDM)

### Adaptive Software Development

### Scrum

### Crystal

### Extreme Programming (XP)

### Pragmatic programming

### Lean and Kanban

### El manifiesto ágil

## El discurso Agile

### Catastrofismo y la falsa dicotomía: Ágil o Cascada

El origen del término de “Ingeniería del Software” se le atribuye a Margaret Hamilton, quien desarrollaba el sistema de guía y navegación para la nave espacial Apollo como jefe de la División de Ingeniería de Software del Laboratorio de Instrumentación MIT.

Hamilton explica por qué eligió llamarlo ingeniería de software [1]: “luché para legitimar el software, de modo que tanto esta ingeniería como los que la construían recibieran el respeto que merecían, por lo que empecé a usar el término “ingeniería de software” para diferenciarlo del hardware y de otras formas de ingeniería. Cuando empecé a usar estas palabras, se consideraban graciosas. Fue una broma recurrente durante mucho tiempo. Les gustaba bromear con mis ideas radicales. El software acabó ganándose el mismo respeto que cualquier otra disciplina.”

La expresión se popularizó a mediados de la década de los 60 apareciendo incluso como reclamo comercial [2]. Aunque no fue hasta 1968, en la conferencia NATO de nombre homónimo, cuando se oficializó [3]. En ese evento, Friedrich L. Bauer habló por primera vez de la crisis del software, apuntando a la complejidad del software como principal responsable de la cantidad de proyectos fallidos, retrasados o con sobrecostes.

Curiosamente durante aquel acontecimiento Andy Kinslow ya anunciaba que el proceso de diseño del software es iterativo. Lo que Douglas Ross completaba criticando el proceso de desarrollo basado en el concepto de especificar lo que vas a hacer, y luego hacerlo. Según sus palabras, los proyectos que se denominan exitosos son aquellos que han cumplido con sus especificaciones, aun cuando estas se basaron en la ignorancia de los diseñadores antes de comenzar el trabajo. El proceso que Kinslow y Ross criticaban se conocería más tarde como metodología en cascada.

La icónica representación que ilustra el modelo de cascada apareció por primera vez en un artículo de Winston Royce en 1970 sobre problemas en la gestión de proyectos de TI de grandes sistemas de software [4]. Royce escribe justo después de introducir el patrón: “Creo en este concepto, pero la implementación descrita anteriormente es arriesgada e invita al fracaso”.

Royce no define el modelo de cascada y lo recomienda, todo lo contrario: identifica el patrón y expone que, si la fase de prueba se realiza una vez finalizado todo el proceso de desarrollo, los fallos encontradas allí probablemente resultarán en un importante rediseño del software. Royce propone en su artículo mejoras, y desaconseja este proceso de desarrollo.

Además, describe los efectos devastadores de este proceso que 30 años más tarde recordaría la literatura Agile: “Es probable que los cambios de diseño requeridos sean tan disruptivos que los requisitos de software en los que se basa el diseño y que proporcionan la razón para todo se violan. Se deben modificar los requisitos o se requiere un cambio sustancial en el diseño. En efecto, el proceso de desarrollo ha regresado al origen y se puede esperar hasta un 100 por ciento de exceso en el cronograma y / o los costos”.

Sin embargo, Royce nunca usó la palabra cascada ya que no existía un método de cascada en ese momento. El término cascada hasta el artículo "Requisitos del software: ¿son realmente un problema?" Por T.E. Bell y T.A. Thayer en 1976 [5].



Ilustración 1: Modelo en cascada [9]

Esta ilustración de Winston Royce se hizo famosa como el modelo de cascada. Sin embargo, no usó el término cascada en su artículo y lo usó para mostrar cómo el proceso de desarrollo no está funcionando.

No solo eso, sino que ya en 1970, Royce explicó que no se puede crear un buen software sin iteraciones, realizar pruebas exhaustivas y recopilar comentarios de los usuarios lo antes posible.

Años más tarde, las publicaciones y textos de la literatura en defensa de la agilidad aludían al mal estado de la industria para proponer nuevas ideas y herramientas que solucionasen esta situación. Mencionando de forma explícita la metodología en cascada como el causante de la crisis del software.

Dicha literatura se apoya en el Chaos Report de 1994 [6] de la compañía Standish Group. El informe una visión del éxito o fracaso de los proyectos de software:



Ilustración 2: Chaos Report 1994. Apuntes de MDW – MiW [7]

Podemos encontrar como se justifica la agilidad como solución a los malos resultados obtenidos con las metodologías empleadas hasta la fecha. La literatura ágil insiste en plantear la dicotomía de elegir una metodología ágil enfrentándola a la metodología de cascada [8] [9].

Incluso el último informe de Standish Group sigue haciendo referencia a este enfrentamiento. Ofreciendo unas cifras claramente mejores para los proyecto llevados con metodologías ágiles.



Ilustración 3: Ágil vs. Cascada [10]

Tras 40 años de evolución de ingeniería del software, dónde desde 1968 se anuncian los problemas de la cascada, las metodologías agiles dilapidan o se apropian de todo este progreso y confrontan una metodología con conocidas deficiencias al nuevo modelo.



Ilustración 4: Estadísticas por tipo de proyecto [10]

En este ejercicio de confrontación, las metodologías formales como RAD, RUP, OO,… son olvidadas o agrupadas junto con cascada.

Curiosamente, el mismo informe del Standish Group de 2015 expone una tasa de éxito similar a proyectos desarrollados con metodologías modernas frente a aquellos desarrollados con métodos tradicionales. Teniendo los primeros un porcentaje de proyectos fallidos sensiblemente mayor a las difamadas metodologías pesadas.

Cabe recordar además que estos informes citados en la literatura ágil fueron desacreditados [11] [12]. En resumen, estos resultados son inconsistentes, no están confirmados por otros estudios y se basan en datos de propiedad exclusiva que los investigadores independientes no pueden contrastar. Sin embargo, hasta el día de hoy continúan siendo citados como una justificación para los procesos ágiles.

### Un conjunto de casos de éxito, pocas estadísticas reales. Confundiendo causalidad y casualidad.

### Todo o nada. No estas siendo lo suficientemente ágil

## La agilidad hoy

### Cómo Agile se convirtió en sustantivo y perdió sus valores con el paso del tiempo.

De acuerdo con Martin Fowler [13], la agilidad se ha convertido en algo comúnmente establecido, ya no es exótico, ni está mal visto como pudiera ser en los primeros momentos. Efectivamente así lo demuestran diferentes estudios y encuestas realizadas [14] [15] [16] [17].



Ilustración 5: Método de desarrollo primario utilizado en la organización de proyectos. [14]

Los resultados de diferentes investigaciones coinciden en que la gran mayoría de las empresas – por encima del 80% – se apoyan en mayor o menor medida las metodologías agiles dentro de sus procesos y que su adopción va, además, en aumento.



Ilustración 6: Experiencia y adopción de las metodologías Agiles en la empresa. [16]

Más de un 85% de los desarrolladores que respondieron a la encuesta de StackOverflow [17] señalan que utilizan metodologías agiles en la gestión de sus proyectos, lo que concuerda con los estudios realizados por diferentes agencias privadas. Curiosamente, estas empresas consideran que su organización está por debajo de un alto nivel de competencia con prácticas ágiles [16].

Este aparente éxito en la adopción de las metodologías ágiles de desarrollo permanecería a nivel superficial en opinión de Martin Fowler [13]. Un número creciente de “agilistas”, incluidos los signatarios destacados del Manifiesto Agile original, han hecho pública su preocupación con el estado de cosas.

Distintos autores han bautizado de diferente manera matizando esta inquietud con respecto a la implementación de la agilidad que se está llevando a cabo hoy día: Martin Fowler lo llama “Dark Agile” [18], “Faux Agile” según Ron Jeffries [19], “Flacid agile” en palabras de Andy Hunt [20], o el anti patrón “Methodology Façade” para Ken Schwaber [21].

En palabras de Dave Thomas, la famosa reunión de Utah se realizó con el propósito de compartir el conocimiento adquirido por una serie de personas que tenían ideas comunes en cuanto al desarrollo de software se refiere [22]. Durante el encuentro trataron de describir aquellas ideas, lo que finalmente resultó en el Manifiesto para el Desarrollo Ágil de Software [23].

Sin embargo, para Andy Hunt [25] tras la publicación del manifiesto, el resultado no fue el esperado. El documento fue creado con la idea de cambiar la cara del desarrollo de software para mejor. En 2001, cuando se redactó el manifiesto y se lanzó el movimiento de desarrollo de software ágil, los autores esperaban ver una oleada de nuevos métodos, nuevas prácticas y formas de agilidad. Estas nuevas ideas seguirían las pautas del manifiesto, pero introducirían nuevas ideas y enfoques y ayudarían a avanzar en el desarrollo de software.

Sin embargo, para varios firmantes consideran que eso no es lo que finalmente está sucediendo. En cambio, hemos visto una adopción a mayor escala de prácticas individuales que favorecen el desarrollo ágil de software, incluso anteriormente prácticas controvertidas, como la programación en pares, e incluso prácticas de higiene básicas, como el control de versiones, que antes no siempre se seguían. En general, esto parece haber tenido un efecto muy positivo en muchas organizaciones de desarrollo, pero seguramente es menos de lo que muchos de nosotros habíamos esperado.

Dave Thomas ilustra el deterioro de los principios de la agilidad en el contexto de la gramática. La palabra ágil o “Agile” se usa como un nombre cuando es un adjetivo. Dave recuerda como el titulo original del manifiesto es “The Manifesto for Agile Software Development” aunque finalmente se ha popularizado como “The Agile Manifesto”. como Eso es simplemente un error. “Haz Agile Right” y “Agile for Dummies” son solo dos de los innumerables ataques contra el idioma inglés que presenta la palabra. No tienen sentido. Agile no es un sustantivo, es un adjetivo, y debe calificar algo más. "Hacer ágil" es como decir "Hacer naranja".

Pero, más allá del problema de la gramática, hay un problema mayor. Una vez que el Manifiesto se hizo popular, la palabra ágil se convirtió en un imán para cualquier persona con puntos a proponer, horas para facturar o productos para vender. Se convirtió en un término de mercadeo, cooptado para mejorar las ventas de la misma manera que palabras como eco y natural. Una palabra que se abusa de esta manera se vuelve inútil: deja de tener significado cuando se convierte en una marca.

Esto duele a todos, pero soy particularmente sensible al daño que causa a los desarrolladores. No es fácil escribir código, y los desarrolladores, naturalmente, están atentos a las cosas que les ayudarán a entregar valor de manera más efectiva. Todavía creo firmemente que adherirse a los valores y prácticas del manifiesto les ayudará en este esfuerzo.

Pero una vez que la palabra ágil deja de tener sentido, los desarrolladores ya no pueden usarla como una guía de lo que es útil en su práctica.

Parece que el enfoque inicial del desarrollo de software ágil se ha olvidado a lo largo del tiempo, y el objetivo ha pasado de convertirse en profesionales adaptables, flexibles y ágiles a simplemente seguir un subconjunto de prácticas ágiles prescritas y canónicas. Parece que la gente ha olvidado por qué surgió este movimiento.

*Nosotros, como pueblo, habíamos olvidado por qué estábamos haciendo lo que estábamos haciendo. Si nuestra memoria masiva colectiva puede olvidar eventos tan importantes y fundamentales en nuestra historia global compartida, entonces recordar las sutilezas de un tema de nicho como "formas efectivas de desarrollar software" probablemente esté condenado desde el principio. Por lo tanto, no es demasiado sorprendente considerar que tal vez la intención original detrás del movimiento de desarrollo de software ágil y sus objetivos se hayan diluido con el tiempo, y tal vez no se haya transferido a nuestros colegas, recién llegados a nuestros equipos o nuestros sucesores. No es sorprendente que muchos adoptantes y sus organizaciones ahora afirman estar "cómodos" con la idea del desarrollo ágil de software y su propio uso. Si te consideras cómodo con "ágil", entonces eso es muy malo, porque probablemente significa que lo estás haciendo mal. Incómodo con Agile Software, los equipos de proyectos parecen mostrar una tendencia hacia uno de los dos estados extremos:*

*1. Una rutina cómoda y repetitiva (algunos pueden decir "somnambulant") que termina en desastre cuando el equipo está cegado por los nuevos desarrollos (ya sea de mercado, la tecnología, o la organización).*

*2. Desorden caótico, donde la fuerza motriz principal es el pánico, alimentado por las horas extraordinarias implacables, los objetivos y prioridades en conflicto, y las suficientes decisiones deficientes para alimentar un calendario Dilbert.*

*Sin embargo, dada la elección, no desea apuntar solo a la mitad de estos extremos caricaturescos. En lugar de eso, debes arrastrarte hasta el borde de ese aterrador y salvaje desastre de desastre de desarrollo disfuncional: hasta el borde del abismo, pero sin caerte. Piensa en esa sensación incómoda de volverse en tu asiento, justo antes de ti. perder el equilibrio y atraparte a ti mismo. Ahí es donde debe ser una práctica ágil para prosperar: equilibrada en un borde estrecho que requiere pequeños ajustes constantes para mantener el equilibrio.*

*Creo que la mejor definición que he visto que capta este espíritu ágil proviene de la Dra. Patricia Benner, autora de From Novice to Expert. Hablando sobre la naturaleza de la experiencia y cómo capacitar a las personas en las prácticas del mundo real (enfermería clínica en este caso), dijo: “Las prácticas nunca pueden objetivarse o formalizarse por completo, ya que siempre deben elaborarse de nuevo en relaciones particulares y en situaciones reales. tiempo”* [26]*. Es decir, nunca se puede definir completamente ágil, o sus prácticas, porque están en constante evolución para satisfacer necesidades específicas en circunstancias específicas. Agile debe ser siempre cambiante, siempre cambiante, siempre respondiendo al cambio en el contexto. Como profesional de desarrollo de software ágil, debe seguir pensando y ajustándose. Sugiero humildemente que esta idea fundamental de desarrollo ágil es lo que (colectivamente) hemos olvidado.*

*Esto se debe a que los principiantes de una nueva habilidad tienden a seguir reglas libres de contexto y es probable que se conviertan en fanáticos ágiles que ponen demasiado énfasis en algunos métodos formales vinculados a ágil. Haciendo cosas en las que ya se sienten cómodos, en lugar de pensar por sí mismos. En lugar de seguir la metodología ágil a ciegas, debe ajustarse a las necesidades de cada uno* [27]*.*

*Esa fue siempre la intención con lo que llamamos "ágil". Nunca fue la intención de ser un conjunto fijo y estático de nada. Recuerda las primeras palabras del Manifiesto Ágil, que escribimos hace diez años:*

*"Estamos descubriendo mejores formas de desarrollar software ..." - agilemanifesto.org*

*Todavía estamos descubriendo, aun descubriendo. Y tú también deberías estarlo. Lo que me funciona bien no necesariamente funcionará bien para ti; lo que funciona bien para usted ahora no necesariamente funcionará bien la próxima vez* [28]*.*

*Según él, los valores ágiles se han corrompido principalmente por tres razones: primero, un complejo agilindustrial en constante crecimiento⁴. Este complejo ágil-industrial simplemente capacita a las personas brevemente, proporciona certificados brillantes de pocos días y los empuja a ocupar puestos de consultoría o gestión. En segundo lugar, un enfoque en la metodología. Esto ha llevado a un énfasis excesivo en las reglas formales a costa de la falta de reconocimiento de la excelencia técnica por parte de ingenieros capaces y experimentados. Y en tercer lugar, centrarse en proyectos en lugar de productos. En lugar de conectar a los desarrolladores con los clientes y centrarse en la calidad, se valora el plazo y la finalización del proyecto*.

*Robert C. Martin señaló recientemente, el enfoque Agile en sí mismo fue una reacción a una crisis en el desarrollo de software* [24]*.*

### Hecho para desarrolladores impuesto por las empresas.

*Un síntoma sorprendente de la crisis de Agile son las imposiciones de Agile en los equipos, que parece ser una práctica común en la actualidad. Si Agile es tan grande y realmente otorga más poder y autonomía a los desarrolladores, ¿por qué es comúnmente impuesto por la alta gerencia? En mi experiencia, esto es bastante común y esta impresión está respaldada por un número cada vez mayor de líderes de pensamiento ágil.*

*Imponer un proceso en un equipo es completamente contrario a los principios del software ágil, y lo ha sido desde su inicio. el conflicto entre los valores de Agile y un enfoque en los procesos de negocios (impuestos) relacionados con Agile.*

*Las verdades del manifiesto ágil parecen sostenerse a sí mismas. Pero las deficiencias en el análisis de por qué exactamente no funcionan según lo planeado son desconcertantes. ¿Somos todos demasiado suaves en ágil? ¿Está el emperador desnudo y nadie se atreve a decir la verdad abiertamente?*

*Un gran error, al bloquear nuestra comprensión de la naturaleza de Agile, es la idea de que Agile está superando el taylorismo como lo sugirió Fowler. Al igual que el mito de la cascada, esto no es cierto, si se examina más detenidamente:*

*El software de escritura e ingeniería es (quizás pronto: ¿solía ser?) Un trabajo de conocimiento altamente calificado. Es intrínsecamente diferente de los bienes de fabricación y no se puede racionalizar fácilmente. Pero trabajar en los procesos ágiles comunes como lo sugieren Scrum o Kanban es lo más parecido a trabajar en una línea de ensamblaje, ya que el software de escritura puede ser:*

*El trabajo se divide en los pasos más pequeños y más fáciles posibles.*

*El ritmo del trabajo es controlado, medido y gestionado.*

*El objetivo final es hacer que el trabajador de software sea desechable por el proceso e incluso la brecha entre los ingenieros altamente experimentados y los miembros menos capacitados del equipo. Esto se hace en beneficio de la productividad y la calidad predecible del producto resultante, de una manera que pretende ser lo más reproducible posible. Sin embargo, al hacer esto, Pioneros y Genios en un extremo y Spaghetti-Script-Cowboys en el otro extremo del espectro ya no son indispensables y están claramente desactualizados.*

*Cortar el trabajo y eliminar la habilidad requerida tanto como sea posible aprovechando procesos predefinidos es el corazón del taylorismo. Pero la industria del software no ensambla en masa el mismo modelo de automóvil una y otra vez como lo hizo el Fordismo del siglo 20 y no puede simplemente copiar la línea de ensamblaje. E incluso la manufactura se alejó de este enfoque. Si observamos algunos otros aspectos de los procesos de producción ágil, vemos más aspectos que también surgieron en la producción en masa moderna, tales como:*

*Controlar y mantener una alta calidad está en el centro de atención.*

*Control total de la productividad, con el objetivo de eliminar las fallas y maximizar el uso de recursos.*

*Integración del personal en producción por un énfasis en el trabajo en equipo.*

*Más autonomía en la toma de decisiones en el sitio de producción real.*

*Jerarquías planas y reducción de énfasis de los símbolos de estado.*

*Todos estos también son componentes elementales de los métodos de gestión modernos, desarrollados en Japón y, a menudo, etiquetados como "Toyotismus". Algunos sociólogos identifican el toyotismo como el posfordismo. Algunos como ni pre- y post-fordismo²⁰. Al igual que con el taylorismo, el toyotismo también es un proceso altamente racionalizado y tienen mucho en común.*

*La cercanía de los procesos ágiles al toyotismo también se puede ejemplificar examinando Kanban, que es la segunda metodología ágil más popular hoy en día. "Kanban" es la palabra japonesa para Signboard, un elemento central en el estilo japonés de producción justo a tiempo y el Kanban aplicado en la producción de Software es un descendiente directo del Kanban que surgió en el Toyotismo. Se aplicó por primera vez al software en Seattle 2007²¹. Pero también Scrum muestra un alto grado de primo con este estilo de gestión.*

*Esos enfoques modernos y magros en la gestión son las claves para el auge de la industria moderna japonesa y su alta competitividad en la industria electrónica y automotriz. El enfoque ágil \* no \* superó el taylorismo, simplemente implementó algo parecido a su forma moderna: manufactura esbelta: el modo Toyota.*

* *Agile is something you become (james Coplien)*

### El problema de las certificaciones

Actualmente, las tres más importantes son Professional Scrum Master (PSM) de Scrum.org, Certified Scrum Master (CSM) de Scrum Alliance y Agile Certified Professional (PMI-ACP) de PMI (los mismos de la certificación PMP).

Las dos primeras tienen su origen en la misma persona, Ken Schwaber. Ken es uno de los creadores de Scrum, que, junto con Jeff Sutherland, definió las versiones iniciales de Scrum que presentaron juntos formalmente en la OOPSLA del 95.

Juntos crearon también la organización Scrum Alliance en la que comenzaron a certificar profesionales de Scrum con la certificación CSM.

En 2010, Ken decide dejar la Scrum Alliance y fundar el instituto scrum.org para intentar orientarlo más hacia el objetivo de divulgar Scrum. Desde este nuevo instituto (scrum.org) se comenzaron a entregar las certificaciones PSM. Desde 2012 hay también un nuevo competidor en liza, la certificación PMI-ACP que está sonando bastante fuerte desde entonces.



Ilustración 7: Carreras y tipos de certificaciones Scrum [29]

*Los programas de certificación son comunes en la industria del software, pero los desarrolladores más capaces que conozco tienen poco respeto por ellos. La opinión general es que la certificación tiene poca correlación con la competencia. Esto se agrava en la comunidad ágil con la asociación de certificación con el CMM, que históricamente es todo menos ágil.*

*Parte del problema con la certificación en general es que los programas de certificación han sido muy débiles. Muchos esquemas de certificación orientados al desarrollador son poco más que preguntas de opción múltiple que son fáciles de configurar y marcar, pero no hacen un buen trabajo de sondear más que la memorización de memoria. También es un problema común, no solo en la industria del software. esa certificación se convierte en una industria en sí misma, lo que fomenta la proliferación de pruebas y actualizaciones que cada vez más están ahí para ayudar a los márgenes de ganancia de las empresas de certificación y pruebas.*

*Ninguno de estos problemas es necesariamente fallas con la certificación en general. Puede ser posible crear un programa de certificación que realmente se relacione con la competencia. Pero la certificación todavía tiene problemas particulares para los métodos ágiles.*

*En un proceso impulsado por un plan, el punto es que la conformidad con el proceso es esencial. Por lo tanto, un esquema de certificación puede probar que un equipo o una organización hace un buen trabajo de conformidad con un proceso definido. Sin embargo, en un mundo ágil, cada proceso sigue la autoadaptación, es decir, se espera que el equipo modifique el proceso para adaptarlo a sus condiciones locales. Esto hace que cualquier certificación sea mucho más difícil de diseñar.* [30]

*Recuerdo una conversación sobre cerveza después de XP Universe 2002. Nos preguntamos qué se necesitaría para una certificación de XP. Consideramos que implicaría varias semanas de observación, ver a las personas lidiar con las distintas etapas de un proyecto de software, ver cómo utilizan diversas habilidades, incluida la optimización del proceso. Tal prueba sería costosa de realizar. ¿Estaría la gente dispuesta a pagar por este tipo de programa?*

*A pesar de todos estos recelos, me gustaría que la industria del software encontrara la manera de idear un esquema de certificación significativo. Ayudaría a separar a las personas más capaces y permitiría que las personas más competentes sean mejor recompensadas por su habilidad. No creo que la industria esté en condiciones de idear un solo esquema, de ahí mi escepticismo sobre Swebok. Pero puede ser posible que una escuela particular de desarrollo de software, como XP, encuentre algo. Pero ese aspecto será muy diferente al tipo de programas de certificación que vemos actualmente.*

Curiosamente Kent Beck también ha expresado su malestar con las certificaciones tal y como las conocemos [31], dudando de los conocimientos o aptitudes que se pueden llegar a adquirir en un par de días y haciendo una prueba online. Kent alude además a la responsabilidad de la entidad certificadora que, para mantener la credibilidad y reputación de la acreditación, debería asegurarse de que cada candidato es apto y merece dicho certificado [32].

Kent propone además como modelo de certificación el utilizado por La Leche League [33]. Según este modelo, tanto el acreditado como la entidad son responsables de su comportamiento. La acreditación sería una forma de informar que aquello que dice en el curriculum es cierto, y que la persona en cuestión está alineada con los principios y prácticas de la organización. Para formar parte de la organización debes en primer lugar ser invitado por un miembro.

En este caso concreto, tanto el anfitrión como una tercera persona evaluadora deben asegurar que que el candidato posee los conocimientos y habilidades necesarias. El proceso de evalución incluye:

* Evaluación de los conocimientos técnicos, sociales y organizacionales del candidato para desempeñar su labor.
* Acompañamiento del candidato en su desempeño y critica de sus actuaciones.
* Revisión de un documento
* Interaccion social con otros miembros.
* Ser introducido públicamente al resto del grupo en una conferencia regional.
* Acompañamiento y soporte.
* Dave Snowden Cynefin

### La agilidad, ese gran desconocido: prácticas vs. Principios

En sus inicios, James Highsmith definió la agilidad como “la capacidad de una organización para reaccionar o responder ante cambios en su entorno más rápido que la tasa de estos cambios” [34]. Esta definición sintetiza el propósito último del desarrollo ágil para una empresa. En su lugar, la agilidad se explica como una serie sintética de prácticas (véase Scrum, XP, Lean), o bien como un conjunto de propiedades en oposición a otras (véase el manifiesto).

Lo que denuncian los impulsores y firmantes del manifiesto es que la tendencia actual es adoptar alguna de estas metodologías, o un conjunto de ellas para ajustarse perfectamente a aquello que predica el manifiesto, sin por ello ser ágil. Este fenómeno se conoce como “do agile”, sin ser ágil en absoluto.

Para ilustrar este fenómeno P. Kruchten utiliza como analogía la definición de una carretera [35]. Según sus palabras podríamos tratar de definir una carretera como algo compuesto de rocas trituradas y alquitrán, o como una superficie que es más negra que blanca, más lisa que ondulada… O bien se podría definir una carretera como como un componente de un sistema de transporte que permite a los vehículos circular de un punto a otro. Según sus palabas, las propiedades y los componentes de dicha carretera deberían emanar de esta tercera definición, permitiendo así nuevas y novedosas soluciones en el diseño y ejecución.

## Conclusiones y Posibles Ampliaciones

### Conclusiones

El movimiento ágil ha sido sobre-idealizado. En parte debido al protagonista original que se mostró entusiasmado con los valores ágiles. Este entusiasmo fue amplificado por el Complejo Industrial Agile, que rápidamente se convirtió en un negocio. Considerado sobrio, el cambio hacia Agile cambió los caminos de la industria del software. Pero al final del día, es solo una racionalización en la producción de software comparable a otros procesos de racionalización en otras industrias. Modernizó la forma en que estamos haciendo nuestro trabajo, pero está más impulsado por la búsqueda de la eficiencia que por los valores. Irónicamente, la eficiencia no puede desplegarse completamente cuando los valores no se tienen en cuenta.

Agile parece haber llegado a una meseta. Si bien en muchos lugares aumentó la eficiencia y la productividad², la división entre trabajadores de administración y de software parece continuar. Mucha gente intuitivamente siente que se rompe la promesa, pero es difícil argumentar contra la racionalización en un mundo competitivo. Las antiguas Jerarquías y las estructuras de poder a menudo se esconden detrás de Jerarquías planas y el trabajo en equipo, pero ahora son más difíciles de definir y criticar.

Algunos problemas de Agile pueden deberse a una mala comprensión de algunos practicantes y al desconocimiento de sus valores. Pero suponer que Agile es el final de la historia en el desarrollo de software es increíblemente miope. Asumir que la insatisfacción con Agile solo surge de "no hacer las cosas bien" nos impide preguntar por sus defectos inherentes.

Ir post-ágil no es solo buscar un nuevo nombre para la comercialización y reinventarlo o una Difusión semántica²³ como argumentó Fowler. El posmodernismo no es una difusión semántica del modernismo. Es algo que siguió y evolucionó desde el modernismo, al comprender mejor la naturaleza del modernismo y sus limitaciones inherentes.

Agile comenzó como una especie de crisis de gestión (proyecto) y la gestión todavía parece ser un problema difícil en TI. Tenemos que profundizar en la crisis de gestión y la crisis de software y volver a visitarla. Por el momento, al parecer, los ideales de la comunidad ágil son difíciles de defender en la dura realidad del mundo de los negocios. Llamar a la responsabilidad individual de cada profesional de TI solo no es suficiente. Martin demostró que la industria se está expandiendo rápidamente y que los desarrolladores tienen cada vez menos experiencia o en muchos otros campos. En muchos casos pueden carecer de la madurez para hacerlo.

La desconfianza en desarrolladores, que supuestamente tiene que ser disciplinado y controlado por la administración, es otra cuestión que debemos abordar. Es inhumano, tóxicos y prolonga el nerd-cliché y no ayudará en la búsqueda de personas que trabajan en ella, que tienen una perspectiva más amplia y no sólo son buenos con los ordenadores, sus idiomas y algunos algoritmos y lógica formal.

La confianza es la base de cualquier buena comunicación. Pero la confianza no puede ser exigida. Necesita ser ganado. Este problema está altamente relacionado con Agile, ya que la confianza es esencial para cualquier equipo Agile. Pero nunca se puede imponer.

Y el problema de la confianza no se puede abordar sin mirar el problema del poder. Ágil, especialmente Scrum, es más acerca de la eficiencia de aproximadamente el empoderamiento de los desarrolladores y no es un alejamiento de taylorismo. En una inspección más cercana, esto será visible en todos los conflictos dentro de las compañías que intentan transformarse hacia Agile. Todo lo contrario es cierto: hace que la gente más reemplazable y controlable y es una forma moderna y competitiva de Gestión. Pero cuando los poseedores del poder se vuelven invencibles y los equipos comienzan a imponer el control sobre sí mismos, se evitan los conflictos y la responsabilidad queda oculta. La promesa humano de Agile está roto.

Los colaboradores originales enfatizan que los métodos ágiles deben ser seleccionados cuidadosamente y que los ágiles no deben verse como una bala de plata. Subrayan la importancia de considerar la variedad de prácticas y métodos diferentes que influyeron en el desarrollo del manifiesto. Además, mencionan que las personas deberían cuestionar su comprensión actual de ágil y recomiendan reconsiderar las ideas centrales del manifiesto.

Esta publicación proporciona impresiones del manifiesto. Revela la incomodidad de algunos de los colaboradores con las tendencias actuales, como la certificación o la comercialización. Además, se identifican direcciones interesantes para una mayor evolución del desarrollo ágil y más allá.

Con este trabajo, tratamos de retener este conocimiento para las personas que desean aprender más sobre la opinión del colaborador original sobre el manifiesto para el desarrollo ágil de software. La comprensión y el aprendizaje continuo es un paso importante para dominar los conceptos básicos de ágile.

### Líneas futuras

El presente trabajo revela varias vías para futuras investigaciones. Entre las cuales destacarían dos direcciones potenciales:

* Por un lado, se podría investigar la influencia de la adopción de la mentalidad ágil en el grado de éxito de los proyectos a nivel internacional. Esta dirección viene motivada del hecho de que las encuestas y estudios encontrados provienen todos empresas de consultoría privadas.

No se ha encontrado ningún análisis realizado por una organización independiente, véase una universidad, que ratifique el grado de adopción de unas metodologías frente a otras y que mida el impacto de estas en el éxito de los proyectos.

Sería necesario investigar, solicitando datos de empresas del sector, el grado de adopción de las metodologías, la satisfacción de los clientes y de todos los actores involucrados en el proceso de desarrollo.

* Por otro lado, asumiendo los beneficios de las prácticas del desarrollo ágil, se propone desarrollar un plan de estudios para acercar la enseñanza actual de ingeniería del software a la filosofía de la agilidad. Este trabajo vendría motivado por la constatación de varios de los firmantes del manifiesto de que la agilidad se está aplicando de forma incorrecta.

Con la idea de subsanar en la medida de lo posible esta situación se pretendería incorporar los preceptos de la agilidad desde el comienzo de la carrera del programador, en este caso en los estudios de ingeniería informática.

Se buscaría que el estudiante pudiera interiorizar durante su periodo formativo conceptos agiles en la forma en la se imparten y evalúan todas las asignaturas y no como una asignatura de metodologías de desarrollo del software.

El plan de estudios debería establecer los mecanismos para que la formación y evaluación recibida fuese iterativa e incremental, revisable y adaptable y donde el trabajo del alumno es realizado mediante la colaboración en equipos autoorganizados, e inmersos en un proceso compartido de toma de decisiones.

# Referencias

|  |  |
| --- | --- |
| [1] | J. R. Hancock, «Entrevista a Margaret Hamilton, la pionera de la programación que llevó el Apolo a la Luna,» *Verne, El Pais,* p. https://verne.elpais.com/verne/2014/12/11/articulo/1418314336\_993353.html, 25 December 4014. |
| [2] | A. A. Oettinger, «letter to the ACM membership,» *Communications of the ACM,* vol. 9, nº 8, 1966. |
| [3] | P. Naur y B. Randell, «Software Engineering,» de *NATO Conference on Software Engineering*, Garmisch, 1968. |
| [4] | W. W. Royce, «Managing the development of large Software Systems,» *IEEE WESCON Proceedings,* pp. 1-9, 1970. |
| [5] | T. E. Bell y T. A. Thayer, «Software requirements: Are they really a problem?,» *ICSE '76 Proceedings of the 2nd international conference on Software engineering, IEEE,* pp. 61-68, 13-15 October 1976. |
| [6] | *The CHAOS Report,* The Standish Group, 1994. |
| [7] | A. Yagüe y E. G. Pardo, *Apuntes de la asignatura de metodologías de desarrollo web,* Madrid: Universidad Politécnica de Madrid, 2019. |
| [8] | K. Schwaber y J. Suthrland, Software in 30 days: How Agile Managers Beat the Odds, Delight their Customers, And Leave Competitors in Dust, New York: John Wiley & Sons, Inc., 2012. |
| [9] | J. Sutherland, Scrum: The Art of Doing Twice the Work in Half the Time, London: Random House Business, 2014. |
| [10] | *The CHAOS Report,* The Standish Group, 2015. |
| [11] | R. L. Glass, «The Standish Report: Does it Really Describe a Software Crisis?,» *Communications of the ACM,* vol. 49, nº 49, pp. 15-16, 2006. |
| [12] | J. L. Eveleens y C. Verhoef, «The Rise and Fall of the Chaos Report Figures,» *IEEE Software,* vol. 26, nº 1, pp. 30-36, 2010. |
| [13] | M. Fowler, *Agile in 2018,* Agile Australia: https://www.youtube.com/watch?v=G\_y2pNj0zZg, 2018. |
| [14] | H. P. Enterprise, «Agile is the new normal, adopting Agile project management,» Hewlett Packard Enterprise Development LP, 2017. |
| [15] | F. Dynamics y C. technologies, «How Agile and DevOps enable digital readiness and transformation,» Freefrim Dynamics, 2018. |
| [16] | C. VersionOne, «13th annual State of Agile Report,» CollabNet VersionOne, 2019. |
| [17] | S. Overflow, «Developer Survey Results,» Stack Overflow, https://insights.stackoverflow.com/survey/2018/#work-which-methodologies-do-developers-use, 2018. |
| [18] | M. Fowler, *The State of Agile Software in 2018,* https://martinfowler.com/articles/agile-aus-2018.html, 2018. |
| [19] | R. Jeffries, *Developers Should Abandon Agile,* https://ronjeffries.com/articles/018-01ff/abandon-1/, 2018. |
| [20] | A. Hunt, *The Failure of Agile,* Toolshed Technologies: https://toolshed.com/2015/05/the-failure-of-agile.html, 2015. |
| [21] | K. Schwaber, *Methodology Façade Pattern,* Ken Schwaber's Blog: Telling It Like It Is: https://kenschwaber.wordpress.com/2010/10/20/methodology-facade-pattern/, 2010. |
| [22] | D. Thomas, «The Coding Gnome,» 4 March 2014. [En línea]. Available: https://pragdave.me/blog/2014/03/04/time-to-kill-agile.html. |
| [23] | K. Beck, M. Beedle, A. v. Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. Mellor, K. Schwaber, J. Sutherland y D. Thomas, «Manifesto for Agile Software Development,» 2001. [En línea]. Available: https://agilemanifesto.org/. |
| [24] | R. C. Martin, *The Future of Programming,* Expert Talks Mobile: https://www.youtube.com/watch?v=ecIWPzGEbFc, 2016. |
| [25] | A. Hunt, *Uncomfortable with Agile,* CrossTalk, The Journal of Defense Software Engineering, 2016. |
| [26] | P. E. Benner, From novice to expert: excellence and power clinical nursing practice, Menlo Park, California: Addison-Wesley, 1984. |
| [27] | A. Hunt, *Stop Practicing and Start Growing,* Toolshed Technologies: https://toolshed.com/articles/2016-07-11-stop\_practicing\_and\_start\_growing.html, 2016. |
| [28] | A. Hunt, *The End of Agile,* Toolshed Technologies: https://toolshed.com/articles/2011-08-01-TheEndOfAgile.html, 2011. |
| [29] | S. Alliance, *Certification Types & Tracks,* https://www.scrumalliance.org, 2019. |
| [30] | M. Fowler, *Should there be a certification program for agile methods?,* https://martinfowler.com/bliki/AgileCertification.html, 2004. |
| [31] | K. Beck, *Leaving Facebook,* Being Human: https://www.youtube.com/watch?v=fH4gqsIYzyE, 2018. |
| [32] | K. Beck y C. Andres, Extreme Programming Explained, Boston: Addison-Wesley, 2005. |
| [33] | «La Leche League,» [En línea]. Available: https://www.llli.org. |
| [34] | J. A. Highsmith, *Agile software development ecosystems,* Boston: Addison-Wesley, 2002. |
| [35] | P. Kruchten, «Contextualizing Agile Software Development,» de *EuroSPI 2010 conference*, Grenoble, 2010. |
| [36] | K. Schwaber, *The State of Agile,* https://www.youtube.com/watch?v=8WXT7\_cHsXI: The Central Ohio Agile Association (COHAA), 2014. |

## Anexo I