RACCIS 5(1), pp. 8-13, 2015

Revista Antioqueña de las
Ciencias Computacionales y la Ingeniería de Software

**ISSN: 2248-7441**

*www.fundacioniai.org/raccis*
*raccis@fundacioniai.org*

# Test-Driven Development as a Medical Treatment

# El Desarrollo Conducido por Pruebas como un Tratamiento Médico

**Gabriel Lowe**
Universidad de Oxford. *gabriel.lowe(AT)cs.ox.uk*

**INFORMACIÓN DEL ARTÍCULO**

*Tipo*
Investigación

*Historia*
Recibido: 20-12-2014
Correcciones: 15-03-2015
Aceptado: 30-03-2015

*Keywords*
Agile practices, productivity, Software Engineering, software quality, software testing.

*Palabras clave*
Calidad del software, Ingeniería de Software, prácticas ágiles, productividad, pruebas del software.

**ABSTRACT**
Test Driven Development (TDD) is one of the most referenced agile practices, however, one of the least used in the industry. In this paper TDD it is considered like an imaginary medical pill, and its effects are described from a pharmacological point of view instead of providing a formal systematic review report. It invites the reader to imagine the rest of this document is a piece of medical information for TDD pill, and continue reading with the following question in mind: if TDD out a pill, would accept to improve my health?

**RESUMEN**
El Desarrollo Conducido por Pruebas (TDD) es una de las prácticas ágiles más referenciadas, sin embargo, una de las menos utilizadas en la industria. En este trabajo se trata a TDD como una píldora médica imaginaria y se describe sus efectos desde un punto de vista farmacológico en lugar de proporcionar un informe de revisión sistemática formal. Se invita al lector a imaginar que el resto de este documento es una hoja de información médica para la píldora TDD, y a continuar la lectura con la siguiente pregunta en mente: ¿si TDD fuera una píldora, la aceptaría para mejorar mi salud?



## 1. Introducción

El Desarrollo Conducido por Pruebas (TDD) [1] es una de las prácticas ágiles más referenciadas, sin embargo, es una de las menos utilizadas en la industria. Su abandono se debe principalmente a la falta de comprensión de sus efectos sobre las personas, los procesos y los productos. Aunque la mayoría de personas están de acuerdo en que la escritura de casos de prueba antes que código promueve una implementación más robusta y un mejor diseño, los costos desconocidos asociados con los efectos TDD y la inversión del programador en el paradigma ubicuo código-luego-pruebas ha impedido su adopción.

Para proporcionar una visión general de la evidencia actual sobre los efectos del TDD, se realizó una revisión sistemática de la literatura a su investigación en bases de datos en línea y publicaciones científicas. La revisión sistemática de la literatura es un método de investigación popularizado en la comunidad médica para reunir y analizar resultados de los ensayos clínicos. Una revisión sistemática pretende contestar a la pregunta general: ¿Qué dice la evidencia publicada acerca de los efectos del uso de la técnica X? En medicina juega un papel crítico en la evaluación a la eficacia de los medicamentos y tratamientos alternativos para las enfermedades. Los investigadores en Ingeniería de Software empírica han adoptado este enfoque para resumir y analizar la evidencia sobre los efectos de las prácticas del desarrollo de software.

En este trabajo se trata a TDD como una píldora médica imaginaria y se describe sus efectos con una narrativa desde un punto de vista farmacológico, en lugar de proporcionar un informe de revisión sistemática formal. Se invita al lector a imaginar que el resto de este documento es una hoja de información médica para la píldora TDD y a continuar la lectura con la siguiente pregunta en mente: ¿si TDD fuera una píldora, lo aceptaría para mejorar mi salud?

## 2. La píldora TDD

Los ingredientes de esta píldora son los siguientes y se debe preparar siguiendo exactamente el orden propuesto: 1) elegir una tarea pequeña, 2) escribir un caso de prueba para esa tarea, 3) ejecutar todas las pruebas para verificar si ese caso falla, 4) escribir el código de producción mínimo para completar la tarea, 5) ejecutar todas las pruebas para verificar si pasan, 6) recodificar según sea necesario, y 7) repetir desde el paso 1.

El ingrediente activo en la píldora es la creación de casos de prueba antes de escribir código. Esto requiere del paciente para considerar el diseño de la solución, cómo

fluirá la información, las posibles salidas del código y los escenarios excepcionales que pudieran ocurrir. Ejecutar el caso de prueba recién estructurado antes de escribir el código ayuda a verificar que la prueba está escrita correctamente y que el sistema compila. Si en este momento el caso de prueba pasa no quiere decir que no haya otros defectos. La píldora TDD también implica escribir suficiente código para ejecutar el caso de prueba, lo que anima a un diseño sobrio y modular. Por otra parte, los usuarios de TDD crean una biblioteca de casos de prueba automatizados creciente, que se puede ejecutar en cualquier momento para verificar la exactitud del sistema cada vez que se hagan cambios.

Al igual que con muchos medicamentos, la píldora TDD tiene algunas variantes genéricas, incluyendo ATDD (Acceptance-Test-Driven Development), BDD (Behavior-Driven Development) y STDD (Story-Test-Driven Development). ATDD reemplaza el paso 1 con *tareas lógicas de nivel funcional empresarial*, mientras que STDD utiliza *especificaciones de comportamiento*. El ordenamiento de las tareas en TDD se diferencia de otros tratamientos, pero las variedades oficiales de la píldora también pueden contener grupos de otros ingredientes, tales como dividir el trabajo en tareas simples, refactorizar, mantener ciclos cortos de código de prueba y pruebas de regresión exigentes. Pero, a excepción de la refactorización, algunos ingredientes clave pueden faltar en muchas píldoras *genéricas*, debido a diferentes interpretaciones en la práctica y al dominio del ingrediente activo.

## 3. Metodología

Aquí hay que considerar que la calidad interna de un sistema está relacionada con la calidad del diseño, generalmente con la interpretación de que los buenos diseños son simples, modulares y fáciles de mantener y entender. Aunque TDD se interpreta principalmente como una práctica de desarrollo, también se considera como una práctica de diseño. Se espera que al usar la píldora TDD surja un diseño gradual y sencillo. El diseño simple es conducido por la modularidad necesaria para hacer que el código sea comprobable, escribiendo el mínimo código de producción para completar tareas sencillas y mediante una constante refactorización. Para evaluar la calidad interna de un sistema las pruebas TDD usan una o más de las siguientes medidas: 1) métricas orientadas a objetos [2], 2) complejidad ciclomática, 3) densidad de código, 4) puntos de función. Por su parte, la calidad externa de un sistema suele medirse por el número de defectos antes y después de la entrega. TDD se asocia con la pretensión de aumentar la calidad externa, porque anima a escribir casos de prueba, los desarrolladores trabajan en tareas simples que son fáciles de comprender, el sistema está bajo frecuentes pruebas de regresión y los errores debidos a los cambios pueden ser fácilmente detectados. En las pruebas de TDD la calidad externa se reporta por una o más de las siguientes métricas: 1) casos de prueba que pasan, 2) número de defectos, 3) densidad de defectos, 4) defectos por prueba, 5) esfuerzo requerido para reparar defectos, 6) cambio de la densidad, 7) porcentaje de cambios preventivos.

El foco de esta investigación fue recopilar evidencias cuantitativas sobre los efectos de la píldora en la *calidad interna* del código, la *calidad externa*, la *productividad* y *calidad de la prueba*. La evaluación a los resultados se basa en datos obtenidos de 32 ensayos clínicos. Para ello se colectaron 325 informes de investigación, reportes técnicos, tesis y literatura gris. Mediante un proceso de selección de dos niveles y a través de filtros el conjunto inicial se redujo a 22 informes. No se tuvo en cuenta los estudios realizados antes del 2000, ni los estudios cualitativos, las encuestas y los análisis totalmente subjetivos. Algunos de esos informes contenían ensayos múltiples o superpuestos, es decir, los mismos resultados se reportaban en varios documentos; en tales casos, el trabajo se contó solamente una vez. Posteriormente, se extrajo la información clave de los informes: el diseño del estudio, el contexto del estudio, los participantes, los tratamientos y los controles y los resultados del estudio. En total se analizaron 22 informes que contenían 32 ensayos únicos.

## 4. Resultados

Estos ensayos se realizaron en los ámbitos académicos o industriales en forma de experimentos controlados, estudios piloto o proyectos comerciales. Los experimentos controlados se llevaron a cabo en laboratorios académicos o entornos industriales controlados y con un protocolo de investigación definido; en los estudios piloto se utilizaron tareas experimentales menos estructuradas; y los proyectos comerciales describen equipos industriales que utilizan TDD como parte de su trabajo diario. Los participantes en estos ensayos tenían diferentes niveles de experiencia, que van desde estudiantes de pregrado a estudiantes de posgrado y profesionales. El número de participantes por ensayo varió desde un individuo hasta equipos de más de 100 personas. El esfuerzo invertido abarca un amplio intervalo, que va desde unas pocas horas-persona hasta miles horas-persona. Cada ensayo compara los efectos de la píldora TDD con respecto a otro tratamiento. Los sujetos en los grupos de tratamiento se componen de varias unidades: individuos, parejas, equipos y proyectos.

Los 32 ensayos se clasificaron en cuatro niveles con base en la experiencia de los participantes, el detalle de la construcción experimental y la escala de la prueba. La experiencia de los participantes se determina: si eran estudiantes universitarios, estudiantes de postgrado o profesionales. Las descripciones de la dinámica de TDD y el tratamiento de control se utilizaron para evaluar la construcción de la prueba, ya fuera *buena*, *adecuada*, *pobre*, o *desconocida*. Una *buena* construcción debe cumplir todos los ingredientes TDD prescritos anteriormente, una *adecuada* prescribe pruebas pero no utiliza todos los ingredientes TDD, una *pobre* no cumple las medidas de TDD y una *desconocida* no especifica si los pasos TDD son forzados o no.

Por último, la escala de un ensayo se registra como *pequeña*, *mediana* o *grande*, dependiendo del esfuerzo reportado o estimado con base a la duración y el número de participantes. En los proyectos pequeños se requiere

menos de 170 horas-persona de esfuerzo total, mientras que los grandes necesitan entre 3.000 y más de 20.000 horas-persona. Se utilizó un algoritmo de agrupamiento sencillo para categorizar la escala de los proyectos, mientras que la experiencia de los participantes y los detalles de la construcción se basó en los datos descriptivos que reportan los informes de los ensayos.

La confianza en que los resultados del uso de la píldora TDD se generalizaran a casos de la vida real aumentó a medida que el nivel de los ensayos se incrementó. El nivel más bajo, *L0*, contiene todos los ensayos a pequeña escala que reportan menos de 170 horas-persona de esfuerzo o menos de 11 participantes. El nivel *L1* consiste en ensayos de mediana o gran escala con construcciones desconocidas o pobres. El nivel *L2* consiste en ensayos de mediana o gran escala con construcciones adecuadas o buenas y estudiantes de pregrado. Por último, el nivel más alto, *L3*, contiene ensayos medianos o de gran escala con construcciones adecuadas o buenas y estudiantes de posgrado o profesionales. En las Tablas 1 y 2 se resumen estos resultados.

**Tabla 1**. Niveles de los ensayos

| Atributo | L0 | L1 | L2 | L3 |
|---|---|---|---|---|
| Experiencia | Alguna | Alguna | Estudiantes pregrado | Estudiantes posgrado o profesionales |
| Construcción | Alguna | Pobre o desconocida | Adecuada o buena | Adecuada o buena |
| Escala | Pequeña | Mediana o grande | Mediana o grande | Mediana o grande |

**Tabla 2**. Tipos y cantidades de ensayos

| Tipos | L0 | L1 | L2 | L3 |
|---|---|---|---|---|
| Experimentos controlados | 2 | 0 | 2 | 4 |
| Estudio piloto | 2 | 0 | 5 | 7 |
| Uso industrial | 1 | 7 | 0 | 2 |

## 5. Análisis de resultados

### 5.1 Eficacia de TDD

Se analizaron los ensayos TDD que reportaron resultados cuantitativos de los efectos de la píldora en la productividad, la calidad interna y externa y la calidad de la prueba. La comparación directa de los resultados cuantitativos era imposible, porque los ensayos midieron la eficacia de TDD de diferentes maneras. En lugar de ello, se asignó a cada ensayo un valor resumen de *mejor*, *peor*, *mixto* o *diferencia no-concluyente*. El valor resumen se determinó por los resultados cuantitativos reportados en comparación con un control. Además, se incorpora la interpretación del informe del autor de los resultados del ensayo. En ensayos con un valor resumen de mejor, la mayoría de las medidas cuantitativas favorecen la píldora TDD en comparación con el tratamiento control; con un valor de peor, la mayoría de medidas favorecen el tratamiento control; con un valor de diferencia no-concluyente, no fueron concluyentes o no reportan diferencias observables. Por último, en los ensayos con un valor resumen de mixto, algunas medidas favorecen a TDD mientras que otras no lo hacen. En todos los casos, la asignación de resumen se guio por la interpretación del reporte del autor de los hallazgos del estudio, ya que, en muchos casos, se omiten detalles de juicio que habrían permitido una evaluación externa objetiva.

### 5.2 Calidad interna

Las evidencias disponibles de los ensayos sugieren que TDD no tiene un efecto consistente en la calidad interna. Aunque parece dar mejores resultados sobre el grupo control para determinados tipos de métricas, tales como complejidad y reutilización, pero para métricas como acoplamiento y cohesión a menudo son peores. Otra observación de los datos de prueba es que TDD produce código menos complejo a nivel método/clase, pero más complejo a nivel paquete/proyecto. Este efecto inconsistente es más visible en los ensayos más rigurosos, es decir, L2 y L3. Las diferencias en la calidad interna se pueden deber a otros factores, tales como la motivación, la habilidad, la experiencia y los efectos de aprendizaje. En la Tabla 3 se muestra la clasificación de los ensayos de acuerdo con las métricas de calidad interna. El primer número es el total de los ensayos y entre paréntesis los ensayos L2 y L3.

**Tabla 3**. Efectos de TDD sobre la calidad interna

| Tipos | Mejor | Peor | Mixto | No-concluyente |
|---|---|---|---|---|
| Experimentos controlados | 1 (0) | 0 (0) | 0 (0) | 2 (2) |
| Estudios piloto | 1 (1) | 1 (1) | 3 (1) | 2 (2) |
| Uso industrial | 3 (1) | 1 (1) | 0 (0) | 0 (0) |

### 5.3 Calidad externa

Existe cierta evidencia que sugiere que TDD mejora la calidad externa, y aunque los resultados de los experimentos controlados en su mayoría son no-concluyentes, los estudios de uso industrial y piloto le favorecen fuertemente. Sin embargo, la evidencia de soporte de los de uso industrial y los experimentos controlados desaparece después de filtrar los menos rigurosos, es decir, ensayos L0 y L1. Además, la evidencia de los estudios piloto y los experimentos controlados es contradictoria cuando se filtran los ensayos L0 y L1. Sin embargo, si se tiene en cuenta por igual a todos los estudios, la evidencia sugiere que la píldora TDD puede mejorar la calidad externa. En la Tabla 4 se clasifican los ensayos de acuerdo con las métricas de calidad externa.

**Tabla 4**. Efectos de TDD sobre la calidad externa

| Tipos | Mejor | Peor | Mixto | No-concluyente |
|---|---|---|---|---|
| Experimentos controlados | 1 (0) | 2 (2) | 0 (0) | 3 (3) |
| Estudios piloto | 6 (5) | 1 (1) | 0 (0) | 2 (2) |
| Uso industrial | 6 (0) | 0 (0) | 0 (0) | 1 (1) |

### 5.4 Productividad

La dimensión de la productividad genera el debate más polémico de TDD. Aunque muchos admiten que su adopción requiere una empinada curva de aprendizaje, que puede disminuir la productividad en un principio, no hay consenso sobre los efectos a largo plazo. Una línea de argumentación argumenta que la productividad se aumenta con TDD, especialmente por razones como que ofrece un contexto fácil para cambiar de una tarea sencilla a otra, mejora la calidad externa (hay pocos errores y pueden ser detectados rápidamente), mejora la calidad interna (la fijación de los errores es más fácil debido al diseño más simple), y mejora de calidad de la prueba (las posibilidades de introducir nuevos errores es baja debido a las pruebas automatizadas). La línea opuesta sostiene que TDD genera demasiada sobrecarga que tiene un impacto negativo en la productividad, porque toma demasiado tiempo y el enfoque se puede orientar a pruebas de autor en lugar de añadir nuevas funcionalidades. Las diferentes medidas utilizadas en los ensayos de TDD para evaluar la productividad incluyen al desarrollo y el mantenimiento del esfuerzo, la cantidad de código o características producidas en el tiempo y la cantidad de código o características producidas por unidad de esfuerzo de desarrollo.

Las evidencias disponibles de los ensayos sugieren que TDD no tiene un efecto consistente en la productividad. Aunque las de los experimentos controlados sugieren una mejora cuando se utiliza TDD, los estudios piloto proporcionan evidencias mixtas, algunas a favor y otras en contra. En los estudios industriales la evidencia sugiere que la productividad es peor. Incluso cuando se consideran solamente los estudios más rigurosos (L2 y L3), la evidencia está igualmente dividida en favor y en contra de un efecto positivo en la productividad. En la Tabla 5 se clasifican los ensayos de acuerdo con los efectos de TDD sobre la productividad.

**Tabla 5**. Efectos de TDD sobre la productividad

| Tipos | Mejor | Peor | Mixto | No-concluyente |
|---|---|---|---|---|
| Experimentos controlados | 3 (1) | 0 (0) | 0 (0) | 1 (1) |
| Estudios piloto | 6 (5) | 4 (4) | 0 (0) | 4 (3) |
| Uso industrial | 1 (0) | 5 (1) | 0 (0) | 1 (0) |

### 5.5 Calidad de la prueba

Debido a que con TDD los casos de prueba preceden a todas las actividades de desarrollo, se espera que la exactitud de la prueba de un sistema en evolución se facilite con una creciente biblioteca de pruebas automatizadas. Además, y debido a la fina granularidad de las pruebas producidas, que el proceso de prueba sea de alta calidad. En los ensayos, la calidad de la prueba es capturada por la densidad de la prueba y su cobertura, productividad, o esfuerzo. En los resultados existe cierta evidencia que sugiere que TDD mejora la calidad de la prueba, aunque la mayor parte es la que proviene de los estudios piloto. Los experimentos controlados sugieren que los valores de TDD son por lo menos tan buenos como en los tratamientos control. No hay pruebas suficientes del uso industrial para llegar a una conclusión. Por lo tanto, la calidad de la prueba asociada con TDD parece al menos no peor, y a menudo mejor que los enfoques alternativos. Aquí se esperaba resultados más fuertes, debido a que el fomento del desarrollo de casos de prueba es uno de los ingredientes activos principales de TDD, pero la evidencia general reportada en estos estudios no favorece la promoción de las medidas de calidad de la prueba. En la Tabla 6 se observan los resultados acerca de la calidad de la prueba.

**Tabla 6**. Clasificación de los ensayos de acuerdo con la calidad de la prueba

| Tipos | Mejor | Peor | Mixto | No-concluyente |
|---|---|---|---|---|
| Experimentos controlados | 2 (1) | 0 (0) | 0 (0) | 3 (3) |
| Estudios piloto | 7 (5) | 1 (1) | 0 (0) | 1 (1) |
| Uso industrial | 1 (0) | 1 (1) | 0 (0) | 1 (0) |

## 6. Análisis de resultados

Aunque en la mayoría de ensayos no se mide ni controla la cantidad de la píldora TDD tomada (que en lenguaje software se traduce en una falta de atención a los procesos de conformidad), esta dosis terminó siendo una variable a través de ensayos y casos. Los ensayos con construcciones pobres o desconocidas pueden no haber utilizado estrictamente TDD, y es muy probable que los participantes personalizaran la píldora con una selección de ingredientes en lugar de seguir la definición estricta de los libros de texto. Esta cuestión plantea una seria amenaza para sacar conclusiones generalizadas. No hacer cumplir o no medir el uso de TDD es análogo a no asegurarse que los pacientes tomen una píldora para algún tratamiento, o no saber qué dosis tomó el paciente en un contexto médico. Por lo tanto, los efectos observados de la píldora TDD pueden deberse a procesos de conformidad u otros factores que no se describen o controlan adecuadamente.

En los ensayos futuros la conformidad con el tratamiento y el control deben ser monitoreados cuidadosamente.

Independientemente de la calidad de la información de los ensayos TDD, en el proceso surgió un interrogante relacionado: ¿la definición de TDD en los libros se puede aplicar en todos los casos de la vida real? A veces los pacientes mejoran incluso con una píldora de tamaño medio o un cuarto de tamaño modificado para su contexto específico de trabajo y estilo personal. Existen herramientas de registro a nivel micro para la actividad de desarrollo que están disponibles y pueden ser utilizadas para investigar estos temas. Tales herramientas de registro pueden ser útiles para el control de conformidad con los procesos de TDD, como para comprender las implementaciones prácticas en la vida real.

### 6.1 Precauciones y efectos secundarios

1. ¿Cuál es el mejor contexto de uso? No hay un mejor contexto recomendado para el uso de TDD. No se sabe si es aplicable a todos los ámbitos, a todo tipo de tareas dentro de un dominio, o para proyectos de todos los tamaños y complejidades. Por ejemplo, los ensayos no dejan claro si es una práctica aplicable para el desarrollo de sistemas empotrados o sistemas altamente descentralizados, donde las pruebas incrementales puede no ser factibles. Además, a menudo se considera un desafío utilizar TDD para sistemas heredados que pueden requerir una considerable refactorización del código existente.

2. ¿Cualquier persona lo puede utilizar? Un hecho básico sobre el que casi todo el mundo está de acuerdo es que TDD es difícil de aprender. Se trata de una curva de aprendizaje empinada que requiere habilidad, madurez y tiempo, sobre todo cuando los desarrolladores están atrincherados en el paradigma código-entonces-pruebas. Para fomentar la adopción de TDD se requieren mejores herramienta de soporte para generar casos de prueba y una exposición temprana en el aula para cambiar la mentalidad código-entonces-pruebas.

3. ¿Podría ser adictivo? Las comunicaciones personales con los desarrolladores TDD sugieren que se trata de una práctica adictiva. Cambia la forma de pensar y su enfoque para la codificación es difícil de deshacer. Por lo tanto, dejar esta práctica puede ser tan difícil como adoptarla.

4. ¿Puede interactuar con otros medicamentos? No se encuentran estudios centrados específicamente en si TDD funciona mejor o peor cuando se utiliza con otros medicamentos. En un ensayo se sugiere que, cuando se combina con el diseño adelantado, los resultados mejoran en un 40% la calidad externa [3]. Otro compara el desempeño de los desarrolladores individuales y por pares que practican TDD con los resultados de las pruebas incrementales [4]. El resultado es que en los informes no hay diferencia en la calidad externa del software producido, por lo que no se sabe qué prácticas son mejores con TDD. Aunque puede haber prácticas que estimulen efectos deseables, también puede haber algunas que los inhiben.

## 7. Conclusiones

Los efectos de TDD todavía involucran muchas incógnitas. De hecho, la evidencia no es indiscutiblemente coherente respecto a los efectos sobre ninguna de las medidas aplicadas: calidad interna y externa, productividad o calidad de la prueba. Gran parte de la probable inconsistencia puede atribuirse a factores internos que no se describen completamente en los ensayos. Por lo tanto, TDD está obligado a permanecer como un tema controvertido de debate e investigación.

Para los profesionales que buscan un consejo, el equipo de investigación recomienda tomar la píldora TDD, pero cuidando de vigilar sus interacciones y efectos secundarios, y en consecuencia aumentar o disminuir la dosis.

En este trabajo se recopilaron y analizaron algunas evidencias, pero cada lector tiene que tomar sus propias decisiones. En primer lugar, decidir qué cualidades son la de mayor importancia. Por ejemplo, ¿la prioridad es la productividad o la calidad externa? ¿Se puede justificar la inversión de más esfuerzo para crear pruebas de mayor calidad? La evidencia en esta investigación es útil para tomar decisiones con base en los objetivos específicos que cada lector busque. Algunos profesionales de la industria opinan que al tomarse la píldora TDD se engancharon al tratamiento, porque mejora la productividad, aunque la evidencia de este estudio demuestra su deficiencia. Tal vez sea simplemente una percepción, porque con base en estos resultados, sobre todo los relacionados con la evidencia de su impacto positivo en la calidad externa, si no estuvieran usando TDD les gustaría empezar a tomarla en pequeñas dosis y ver si encuentran una mejora en la productividad a largo plazo. Si no hay reacciones adversas aumentarían la dosis poco a poco y seguirían observando.

Otros opinan que aunque TDD es prometedor su adopción puede verse obstaculizada por la incertidumbre acerca de su eficacia, y por el alto costo. Aún así, sus ingredientes parecen fomentar buenos hábitos de programación y desarrollo, brindando en el largo plazo mejor calidad a los programadores y las pruebas. Pero manifiestan que sabe mal al principio. A muchos les gustan más las cosas viejas. Después de todo, es difícil sentirse productivos cuando se pasa gran cantidad de tiempo sin escribir casos de prueba. Por otro lado, nunca han escrito código más limpio y se sienten bien al hacerle cambios al viejo código, pulsar el botón *Run Tests* y estar seguro que todo puede funcionar.

Las evidencias incluidas en este trabajo demuestran que TDD podría ser una cura para cualquiera, sin embargo, no todos deben tratar de usarla como panacea. Su aventura con TDD tiende a variar con ciertos factores, incluyendo la experiencia y el contexto en que se está trabajando. Para los profesionales sería un activo valioso desarrollar una idea acerca de cuándo esperar mejoras de TDD.

## Referencias


[1] Beck, K. (2002). Test-Driven Development: By example. Boston: Addison-Wesley.

[2] Chidamber, S. & Kemerer, C. (1994). A metrics suite for object oriented design. IEEE Transactions on Software Engineering 20(6), pp. 476–493.

[3] Williams, L., Maximilien, M. & Vouk, M. (2003). Test-Driven Development as a defect-reduction practice. Proceedings of the 14th IEEE International Symposium on Software Reliability Engineering, (34). November 17-20, Denver, USA.

[4] Madeyski, L. (2005). Preliminary analysis of the effects of pair programming and Test-Driven Development on the external code quality. Proceedings 2005 Conference on Software Engineering: Evolution and Emerging Technologies (113–123).