# Metodología híbrida para la estimación y tratamiento de la Deuda Técnica de defectos en el Desarrollo Ágil de Software

## Hybrid methodology for estimating and treating Technical Debt of defects in Agile Software Development

María Guadalupe Medina Barrera[1,*] y José Juan Hernández Mora[1]

[1]División de Estudios de Posgrado e Investigación, Tecnológico Nacional de México,Instituto Tecnológico de Apizaco
Av. Instituto Tecnológico 418, San Andrés Ahuashuatepec, 90491 Tzompantepec, Tlaxcala, México
[*]guadalupe.mb@apizaco.tecnm.mx

**PALABRAS CLAVE:**

Estimación de Esfuerzo,
Deuda Técnica,
Defectos,
Desarrollo Ágil

**RESUMEN**

En el Desarrollo Ágil de Software (DAS) se realizan entregas parciales al cliente programadas en plazos muy cortos. Con el afán de cumplir los compromisos, los desarrolladores emplean diversas prácticas para acelerar el desarrollo. Sin embargo, las presiones de tiempo pueden propiciar la creación de errores que si no son corregidos antes de la entrega, acumulan Deuda Técnica (DT) de defectos en el producto. La DT representa el esfuerzo extra que debe invertirse para corregir los problemas causados por la adopción de soluciones rápidas. El problema de la DT es que si no se paga lo más pronto posible, puede llevar al punto de quiebre a un proyecto de software. Por lo tanto, es necesario Estimar el Esfuerzo (EE) que se requiere para pagar la DT de defectos y poder así, gestionarla en entornos de DAS. En este trabajo, se presentan los enfoques de EE reportados en la literatura y se propone una metodología híbrida para estimar la DT de defectos en el DAS. Esta propuesta aprovecha las bondades de los enfoques existentes de EE para obtener estimados realistas que faciliten el pago de la DT de defectos.

**KEYWORDS:**

Effort Estimation,
Technical Debt,
Defects,
Agile Development

**ABSTRACT**

Agile Software Development (ASD) seeks to make partial deliveries to the client scheduled in very short periods. Thus, developers employ various practices to accelerate development and to meet commitments. However, time pressures can lead to injecting errors that, if not corrected before the delivery, accumulate Technical Debt (TD) due to product defects. The TD means the extra effort that must be invested to correct problems caused by adopting quick solutions. The problem is that, if the TD is not paid as soon as possible, it can bankrupt a software project. Therefore, it is necessary to Estimate the Effort (EE) required to pay a TD due to defects and thus be able to manage it in ASD environments. This work presents the EE approaches reported in the literature and then, a hybrid methodology is proposed to estimate the TD due to defects in the ASD. This proposal leverages the benefits of existing EE approaches to obtain realistic estimates that facilitate TD due to defects payment.

• **Recibido: 15 de julio de 2024** • **Aceptado: 18 de septiembre de 2024** • **Publicado en línea: 1 de octubre de 2024**

## 1. INTRODUCCIÓN

El Desarrollo Ágil de Software (DAS) es un paradigma que busca satisfacer al cliente mediante la entrega temprana y continua de software que le aporte valor [1]. En efecto, en un esquema de desarrollo ágil se realizan entregas parciales del producto al cliente, las cuales se planean por iteraciones o ciclos cortos de desarrollo. Sin embargo, las presiones en el cumplimiento de los plazos pueden propiciar entregas parciales con defectos, mismos que


DOI: 10.30973/progmat/2024.16.3/6

deben corregirse en las siguientes iteraciones. Si los defectos no se eliminan cuanto antes, se acumula Deuda Técnica en el DAS.

La Deuda Técnica (DT) es una metáfora introducida por Ward Cunningham en 1992 y representa el esfuerzo extra que debe invertirse para corregir los problemas causados por la adopción de soluciones rápidas para conseguir entregas prontas y/o de menor costo. Desde luego, al acumularse grandes cantidades de esfuerzo después de varias iteraciones en el DAS se puede llegar al punto de quiebre en el presupuesto de un proyecto de software. En este sentido, la DT de software es similar al comportamiento de una deuda financiera. Por supuesto, los desarrolladores adoptan diversas prácticas con la finalidad de reducir costos, acelerar el desarrollo y crear productos innovadores. Todos los días emergen una gran cantidad de herramientas que pueden utilizarse para estos propósitos como los conjuntos de herramientas de aprendizaje profundo. No obstante, se debe tener cuidado en no caer en anti-patrones [2], así como adoptar buenas prácticas que preserven la calidad, seguridad y mantenibilidad del producto.

Ahora bien, entre las prácticas que más emplean los desarrolladores de software para eliminar la DT están las actividades de refactorización [3-4]. Seaman y Guo [5] exponen que el principal de una DT es igual al esfuerzo necesario para pagarla, esto es para refactorizar el código en el mismo ciclo en que se incurrió en DT, antes de que se generen intereses que aumenten la deuda. Así que, la cantidad de esfuerzo requerido para cubrir una DT de defectos en el DAS está en proporción de la cantidad de actividades de refactorización necesarias para lograr un producto libre de defectos por iteración. Entonces, para saber a cuánto asciende la DT de defectos en el DAS debe estimarse el esfuerzo necesario para hacer las correcciones correspondientes en cada ciclo de desarrollo. La Estimación de Esfuerzo (EE) es un proceso mediante el cual se calcula el esfuerzo requerido para terminar cada actividad siendo expresado en número de días-hombre o meses-hombre [6]. Pero, en el DAS intervienen diversos factores que dificultan este trabajo [7], como los cambios constantes en el equipo de desarrolladores.

La EE se realiza básicamente desde dos perspectivas, la basada en modelos y la basada en expertos [6]. La primera utiliza formulaciones matemáticas o algorítmicas considerando el tamaño del producto y sus características. La segunda se basa en la experiencia de los desarrolladores. En la siguiente sección se describen las técnicas de EE empleadas en cada enfoque de acuerdo a la literatura; también se realiza un análisis comparativo. En la sección 3, se exponen trabajos relacionados sobre los enfoques para estimar DT. En la sección 4, se presenta una metodología híbrida de EE para la estimación del principal de la DT de defectos en el DAS. La sección 5 muestra los resultados obtenidos en un proyecto académico de DAS. A continuación, se discuten los hallazgos. Finalmente, se exponen las conclusiones y se plantean los trabajos futuros.

## 2. ENFOQUES EE

### 2.1. EE BASADA EN MODELOS

El objetivo de un enfoque de EE basado en modelos es el de encontrar la fórmula matemática para determinar el esfuerzo de desarrollo de un producto de software. Se identifican dos tipos de técnicas, las paramétricas y las de aprendizaje máquina.

El modelo paramétrico COCOMO (*Constructive Cost Model*) permite estimar el esfuerzo y se han diseñado varias versiones [6, 8-9] que buscan obtener una mejor función de ajuste. La versión COCOMO II tiene como premisa que los costos no son lineales con el tamaño del proyecto. Su formulación matemática es [9]:

$$Esfuerzo = a * LOC^{b} * m \quad (1)$$

Donde: $a$ es una constante determinada por factores como prácticas locales y tipo de software, *LOC* es el tamaño expresado en número de líneas de código, $b$ es un valor relacionado con la complejidad del producto, y $m$ es un valor relacionado con atributos de proyecto y proceso.

Sin embargo, la determinación del tamaño del software en LOC se ha vuelto poco realista conforme han evolucionado los sistemas de cómputo y sus requerimientos. Por ello, el tamaño se calcula como Puntos de Función (PF) para que sea independiente del lenguaje de programación o tecnología. En esta práctica se han desarrollado diversos estándares y procedimientos que son administrados por la IFPUG (*International Function Point Users Group*) [10]. Cabe mencionar que además de los métodos de la IFPUG, también está el método denomidado COSMIC. Mientras que IFPUG identifica en los componentes funcionales los datos y las transacciones, COSMIC solo toma en cuenta estas últimas [11]. También, se han propuesto algunas alternativas como los puntos de función simples (SiFP) que pueden reducir los cálculos [12], aunque no logran mejorar la precisión de las estimaciones [13].

En cuanto a los modelos basados en aprendizaje máquina para EE, se encuentran las Redes Neuronales (RN), los Árboles de Decisión (AD), las Redes Bayesianas (RB) y las Máquinas de Vectores de Soporte (MVS). Estos modelos presentan distintas problemáticas, así como virtudes. Por ejemplo, los AD pueden adaptarse al contexto del problema y requieren poco tiempo de entrenamiento, pero fácilmente caen en el sobre-entrenamiento [14]. En cuanto a las RN, tienden a sobre-estimar con un alto costo computacional [14]; no obstante, arquitecturas de tipo perceptrón multicapa y recurrentes han probado mejorar las predicciones de EE [15]. En relación a las RB, se han propuesto para hacer frente a los problemas que causan incertidumbre en la EE. Para esto, combinan datos históricos con la opinión de expertos para incorporar los factores humanos del desarrollo [16], demostrando suficiente precisión de la EE con conjuntos pequeños de datos [17].

Un enfoque para mejorar las EE, consiste en la combinación de las técnicas de aprendizaje máquina y aprendizaje profundo con otras técnicas, creando así modelos híbridos o ensamblados. Algunos ejemplos son los híbridos de perceptrones multicapa que emplean operaciones morfológicas [18], los modelos de regresión de aprendizaje profundo que se combinan con el juicio de expertos [19], así como los ensambles de algoritmos de aprendizaje máquina: RN, MVS y modelos lineales generalizados [20]. En cuanto al desempeño de ensambles, el método de *Random Forest* que acompla solo ADs ha demostrado ser mejor que diversos tipos individuales de RN [21]. Así, los ensambles arrojan resultados más precisos que cuando se utilizan de forma independiente [22].

## 2.2 EE BASADA EN EXPERTOS

Los métodos basados en la experiencia enfrentan algunos cuestionamientos y críticas como su subjetividad, inconsistencia en las mediciones y sesgos [23,24]. Entre los factores que pueden afectar las estimaciones basadas en la experiencia, están los límites presupuestales, las predicciones previas, así como granularidad de la unidad de tiempo de predicción [25,26]. De cualquier forma, también son reconocidas sus bondades. Al respecto, Lenarduzzi, et al. [14] sostienen que estos estimados son más precisos que los obtenidos por los basados en modelos. Esto, posiblemente se debe a que éstos últimos no consideran objetivamente el impacto de un cambio al igual que los factores que contribuyen al esfuerzo adicional [26]. Entre las técnicas utilizadas bajo el enfoque basado en expertos, se encuentran *Planning Poker*, Historias de Usuario, Analogías y Casos de Uso.

*Planning Poker* se basa en el concenso de un equipo de trabajo para asignar estimados,

basados en los números de la serie de Fibonacci. Estos indican los puntos de historia o bien, los días en que se cree se desarrollará una función del producto. Este proceso alienta el involucramiento de todos los miembros del equipo de desarrollo, permitiendo considerar factores relacionados con el equipo y el proyecto más que factores técnicos [27]. En cuanto a las Historias de Usuario, son una forma de expresar los requerimientos de un proyecto y para estimarlas se usa *Planning Poker* [16].

En el caso de las Analogías, se basan en el RBC para seleccionar los casos que más se parecen. Sin embargo, las evaluaciones inconsistentes ponen de manifiesto su problema de inestabilidad [28]. Mientras tanto, el método de Casos de Uso permite la especificación particular de un caso. Sin embargo, Huanca & Oré [29] recomiendan que para mejorar su precisión conviene primero estandarizarlos para realizar una correcta contabilización de transacciones; continuando entonces con la mejora de la clasificación de actores, la evaluación de su complejidad, el cálculo de factores técnicos, ambientales y de productividad. Al respecto, Azzeh & Nassif [7] hallaron que los factores ambientales impactan de forma significativa la productividad necesaria para desarrollar un proyecto.

### 2.3. ANÁLISIS COMPARATIVO

Cada enfoque para EE tiene tanto ventajas como desventajas. En la tabla 1 se presenta resumen.

**Tabla 1.** Comparación de los enfoques para EE

| Basado en modelos | Basado en expertos |
|---|---|
| Objetividad en las estimaciones | Subjetividad en las estimaciones |
| Sensibles a su estructura y parámetros | Sujetos a sesgos |
| Requieren datos históricos (variados) | Requieren de la disponibilidad de expertos |
| Automatización y rapidez de respuesta | Requieren tiempo para dar un resultado |
| Inflexibles a cambios | Flexibles a factores imprevistos |
| Su preparación requiere tiempo | Apropiado para DAS |

Nota: elaboración propia.

Por un lado, la precisión y confiabilidad de las técnicas basadas en la experiencia dependen del conocimiento del experto. Mientras que, las técnicas basadas en modelos requieren de datos históricos, teniendo que su precisión es mayor a medida que se utiliza más variedad, además de mayor cantidad de ejemplos [30]. Así, tenemos que un enfoque para EE basado en modelos no puede emplearse si no existen datos, mientras que el basado en la experiencia requiere forzosamente la disponibilidad del experto.

## 3. TRABAJOS RELACIONADOS EN LA ESTIMACIÓN DE DT

Una vez identificada una DT, debe estimarse a cuánto asciende para estar en posibilidad de gestionarla y prevenir su impago. Así, la estimación de la DT es una actividad fundamental para su gestión [31].

El enfoque que prevalece para estimar la DT se centra en la evaluación de la calidad del código fuente, siendo así la DT de código la más automatizada [32]. No obstante, también existe interés por calcular otros tipos de deuda. Para esto, se han empleado diversas metodologías, técnicas y herramientas. Entre ellas, se encuentran CAST, CodeMRI, TETRA, CBRI y SonarQube [32-34]; siendo esta última la herramienta más utilizada para identificar y medir cuatro tipos de DT mediante un sistema de reglas: arquitectura, diseño, código y pruebas. En cuanto a metodologías, TETRA [33], considera cinco dimensiones para evaluar la DT: 1) calidad del código, 2) usabilidad, interfaz de usuario y documentación, 3) seguridad, 4) desempeño y 5) uso de código abierto.

Mientras predominan los trabajos dirigidos hacia la identificación y cuantificación automática de la DT, otros se enfocan en proponer nuevas métricas y enfoques para mejorar la medición de DT en proyectos de código abierto. Al respecto, Didier [35] presenta una métrica basada en las más empleadas en la literatura. Por su parte, Li et al. [36] proponen un conjunto de métricas ad hoc para proyectos específicos. Con el objeto

de obtener estimados más certeros, Lenarduzzi et al. [37] aprovechan el enfoque de aprendizaje automático dirigido por datos y muestran que es posible mejorar las técnicas y herramientas actuales.

Cualquiera que sea el enfoque, cada tipo de DT tiene sus propias características [38], por lo que pueden requerir diferentes formas de medición. Adicionalmente, los estimados de DT están sujetos a incertidumbre que pueden afectar su precisión [39]; particularmente en el DAS existe gran variedad de factores que impactan la productividad de los equipos [7]. Más aún, la medición exacta de la DT se complica cuando se desea calcularla, a través de varios ciclos de desarrollo. Aquí, la opinión del experto puede integrar con oportunidad las variaciones correspondientes. Por ello, un enfoque híbrido de EE para cuantificar la DT de defectos se postula como la mejor alternativa para obtener estimaciones más precisas.

## 4. METODOLOGÍA

A continuación, se describe la metodología desarrollada para estimar el principal de la DT de defectos en proyectos de DAS.

### 4.1. ESTIMACIÓN DEL PRINCIPAL DE DT DE DEFECTOS EN DAS

En este trabajo se propone emplear un enfoque híbrido de EE para saber a cuánto a asciende la DT de defectos por iteración en un entorno de DAS. En la Fig. 1 se muestra el proceso y los elementos que se involucran.

El monto total del esfuerzo requerido por una DT por defectos se compone de dos partes: el principal y el interés [5]. El principal de la deuda corresponde al esfuerzo necesario para corregir los defectos en la iteración en la que se originaron. El interés de la deuda se genera por el retardo en la eliminación de los defectos. Así que, a medida que transcurren las iteraciones el interés aumenta debido a que el producto crece y las dependencias aumentan. Por lo tanto, una corrección tardía implica un mayor número de cambios o refactorizaciones. En este documento se abordan los elementos que deben considerarse para estimar el principal de la DT de defectos en el DAS.

En el DAS, la DT de defectos se detecta al finalizar cada iteración durante las reuniones de revisión con el cliente, o incluso, en periodos de prueba de las entregas parciales. Por supuesto, los defectos se pueden originar desde las primeras fases del desarrollo. Wang et al. [40] exponen que existe una relación lineal entre el porcentaje de defectos inyectados en las etapas de requerimientos, de diseño y de codificación, con el porcentaje de esfuerzo necesario para corregirlos. Esto significa que los errores en las etapas de análisis de requerimientos, de diseño y codificación, pueden convertirse en defectos en un producto entregado. Entonces, para estimar el principal de la DT de defectos al finalizar una iteración se requiere consultar el historial del equipo de desarrollo, para indagar cuántas refactorizaciones han realizado para corregir los problemas de requerimientos ($Rr$), diseño ($Rd$) y código ($Rc$). La suma de estas refactorizaciones representa el esfuerzo ($f(x)$) que el equipo de desarrollo ha empleado para eliminar este tipo de defectos. Se tiene entonces que:

$$f(x) = Rr + Rd + Rc \tag{2}$$

Sin embargo, los estimados deben ajustarse según la complejidad ($c1$) de cada problema y los cambios en las capacidades ($c2$) de los desarrolladores que se encargarán de hacer las transformaciones necesarias [41]. Este proceso debe basarse en la experiencia de los miembros del equipo. Así, el equipo debe determinar el factor siguiente:

$$a = c1 \times c2 \tag{3}$$

De esta manera, el principal de la DT de defectos resulta de la composición siguiente:

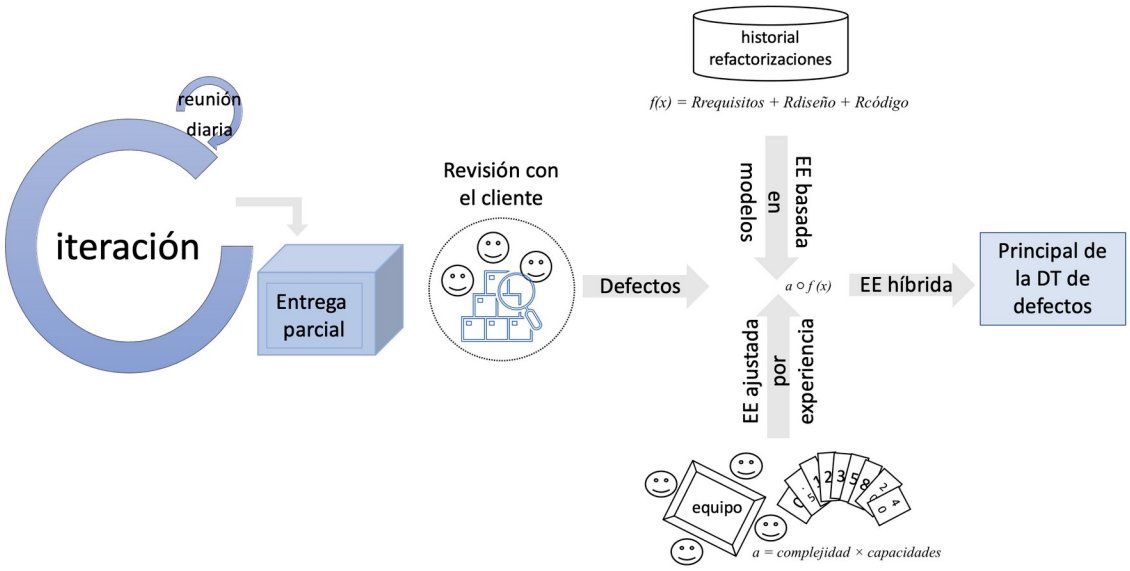


**Figura 1.** Estimación del principal de DT de defectos en el DAS.
Nota: *elaboración propia.*

$$DTdef = a \circ f(x) \quad (4)$$

Considerando lo anterior, el principal de la DT de defectos al finalizar la iteración en que fueron insertados los problemas durante el DAS se obtiene mediante un proceso de EE híbrido. Este proceso recupera los ejemplos en el historial de refactorizaciones y actualiza los estimados de esfuerzo para precisarlos según las capacidades y experiencia de quienes se encargarán de liberar al producto de defectos.

## 5. RESULTADOS

### 5.1 CASO DE ESTUDIO

El proceso de EE híbrido descrito se aplicó en un proyecto escolar de DAS. El periodo de desarrollo fue de tres meses y consistió en una aplicación para el control de asistencia. Se utilizó el marco de trabajo Scrum [42], para lo que se integró un equipo de desarrollo con seis miembros asignando los roles indicados en la Tabla 2. Después de analizar la pila con los requerimientos del producto, se definieron cuatro *sprint* o ciclos de desarrollo ágil. El cliente participó tanto en las reuniones de planeación previas a cada *sprint*, como en las reuniones de revisión del producto al finalizar cada *sprint*.

**Tabla 2.** Equipo scrum

| Miembro | Rol asignado |
|---|---|
| 1 | Scrum master |
| 2 | Arquitecto de software |
| 3 | Analista |
| 4 | Programador |
| 5 | Documentador |
| 6 | Tester |

Nota: elaboración propia.

### 5.2 ESTIMACIÓN

Al finalizar cada *sprint*, el equipo se reunió con el cliente para llevar a cabo las reuniones de revisión del producto. En la Tabla 3 se muestran los defectos detectados en cada *sprint*. Se indica la problemática que generó cada defecto, así como el impacto que puede tener en la calidad del producto final de no ser removido el defecto en cuestión. Ahora bien, para planear la estrategia de remoción de defectos del *sprint* en turno, es necesario estimar primeramente el principal de la DT de defectos correspondiente. Los resultados pueden verse en la Tabla 4. Siguiendo el proceso híbrido descrito en la sección anterior, se consultó el historial de refactorizaciones y esfuerzo ejercido por el equipo en problemáticas similares a las que generaron los defectos detectados. De esta

manera, se obtuvo el principal de la DT de defectos sin ajustar (Eq. 1). En seguida, se empleó la técnica de *Planning Poker* para obtener el factor de ajuste (Eq. 2). Finalmente, mediante la operación de composición $a \circ f$ se obtuvo el principal de la DT de defectos (Eq. 3).

**Tabla 3.** Defectos detectados por sprint.

| S[1] | Def [2] | Problema |
|---|---|---|
| 1 | S1.1 | Especificación deficiente |
| 2 | S2.1 | Rol de usuario no incluído |
| | S2.2 | Atributos de entidad no incluídos |
| | S2.3 | Inconsistencias en diseño conceptual |
| 3 | S3.1 | Diseño de BD deficiente |
| | S3.2 | Protocolocos de acceso ausentes |
| | S3.3 | Elementos de interfaz confusos |
| | S3.4 | Elementos de interfaz no inclusivos |
| 4 | ninguno | |

[1]número de sprint
[2]identificador de defecto

Nota: elaboración propia.

En la Fig. 2 se presenta el conjunto de tarjetas utilizadas para determinar los niveles de complejidad $c1$ y capacidades $c2$ del equipo. Cabe mencionar que los valores de la escala de complejidades van en aumento, a medida que el equipo consensa un mayor nivel de complejidad de los problemas a resolver. Mientras tanto, los valores de la escala de capacidades van en decremento, a medida que el equipo determina que es mayor su experiencia y dominio de las técnicas y herramientas necesarias para eliminar los defectos.

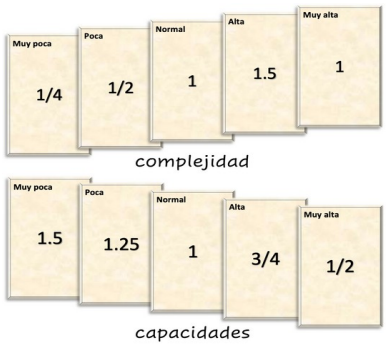


**Figura 2.** Escala de estimación del factor de ajuste.
Nota: Elaboración propia

## 6. DISCUSIÓN

El enfoque híbrido propuesto para estimar la DT de defectos permite tener una gestión eficiente de la DT, ya que estos incorporan tanto los factores técnicos, como los relacionados con el factor humano y el ambiente de desarrollo. Esto solo se puede conseguir mediante la participación del equipo de desarrollo, para ajustar el esfuerzo y refactorizaciones en sus históricos a las circunstancias particulares que prevalecen en el *sprint* en turno. Así, la deteccción y estimación de la DT de defectos, permite planear el pago oportuno de los impactos a la calidad del producto final.

**Tabla 4.** Estimación híbrida del principal de la DT de defectos.

| S[1] | Def [2] | EH | Rr | Rd | Rc | f(x) | c1 | c2 | a | DTdef | EE | EJ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | S1-1 | 50 | 80 | 120 | - | 200 | Normal | Alta | 0.75 | 150 | 37.5 | 42 |
| | | | **Principal de la DT de defectos en Sprint 1** | | | | | | | **150** | **37.5** | **42** |
| 2 | S2-1 | 25 | 50 | 75 | - | 125 | Alta | Muy alta | 0.75 | 93.75 | 18.75 | 16 |
| | S2-2 | 5 | 5 | 5 | - | 10 | Normal | Normal | 1 | 10 | 5 | 5 |
| | S2-3 | 15 | 10 | 33 | - | 43 | Alta | Muy alta | 0.75 | 32.25 | 11.25 | 10 |
| | | | **Principal de la DT de defectos en Sprint 2** | | | | | | | **136** | **35** | **31** |
| 3 | S3-1 | 25 | 3 | 18 | 68 | 89 | Normal | Muy alta | 0.5 | 44.5 | 12.5 | 12.5 |
| | S3-2 | 15 | 1 | 12 | 24 | 37 | Normal | Normal | 1 | 37 | 15 | 15 |
| | S3-3 | 20 | 6 | 24 | 55 | 85 | Normal | Normal | 1 | 85 | 20 | 20 |
| | S3-4 | 15 | 2 | 8 | 50 | 60 | Poca | Alta | 0.375 | 22.5 | 5.625 | 5.5 |
| | | | **Principal de la DT de defectos en Sprint 3** | | | | | | | **189** | **53.125** | **53** |

[1] número de sprint
[2] identificador de defecto
[EH] Esfuerzo en históricos (hrs x equipo)
[EE] Esfuerzo estimado para pagar DTdef (hrs x equipo)
[EJ] Esfuerzo ejercido (hrs x equipo)

Nota: elaboración propia.

Cabe hacer notar que, de los ocho defectos estimados, tres requieren 253.75 refactorizaciones para no afectar su funcionalidad, otros tres requieren 152 rectorizaciones para no demeritar su usabilidad, uno necesita 37 refactorizaciones para no vulnerar su seguridad y el último defecto requiere 32.25 para no impactar en su confiabilidad.

La cantidad de refactorizaciones, así como el esfuerzo requerido por el equipo en horas de trabajo podría incrementarse dramáticamente, de no ser corregidos los problemas al finalizar el *sprint* en que fueron insertados. Es decir, el interés de la DT de defectos se acumularía ciclo a ciclo, aumentando la DT total. Entonces, además del esfuerzo de corrección, ahora debe sumarse el esfuerzo de remediación de daños. Se pretende ampliar este trabajo para cuantificar la DT total de defectos en el DAS.

Debe hacerse notar, que al tratarse de un proyecto escolar, el historial de refactorizaciones y esfuerzo es limitado a los ejemplos que el equipo ha acumulado en proyectos académicos anteriores. No obstante, se observa que el equipo fue mejorando la precisión de sus estimaciones conforme se avanzaba en los ciclos de desarrollo. En el primer *sprint* el equipo subestimó en 4.5 horas el esfuerzo requerido para mejorar la especificación de requerimientos; mientras que en el segundo *sprint* la tendencia fue hacia la sobre-estimación en 4 horas para incluir funcionalidades no consideradas y corregir inconsistencias. Finalmente, en el tercer *sprint* las EE fueron prácticamente exactas al compararlas con lo ejercido.

## CONCLUSIONES Y TRABAJO FUTURO

Para llevar a cabo una EE se requiere de una planeación que permita realizarla de la mejor forma posible, enfocándose siempre en el objetivo de obtener estimados lo más apegados a la realidad. En esta tarea, siempre deben tomarse en cuenta las condiciones que se tengan en el momento de llevar a cabo este proceso. Por ello, resulta valioso conocer en qué consisten los enfoques para EE, tomando conciencia acerca de sus ventajas, pero también de sus desventajas.

La respuesta de un enfoque basado en modelos suele ser muy rápida, aunque para obtener los mejores estimados se requiere de gran cantidad de datos además de mucho trabajo previo como de habilidades técnicas y conocimiento para procesarlos. Debido a esto, se considera que su implementación es un poco engorrosa. De cualquier manera, esta forma de EE es más objetiva que cuando se utiliza el enfoque basado en la experiencia.

En realidad, las técnicas que requieren de la colaboración del experto suelen estar sujetas al sesgo, siendo ésta su mayor crítica por los resultados inconsistentes además de imprecisos, que se pueden obtener. Después de todo, la sensibilidad de este enfoque le permite ser flexible en cuanto que puede incorporar con facilidad factores imprevistos que respondan al cambio.

Finalmente, ambos enfoques pueden ser complementarios sin perder de vista que lo que se busca es la agilidad en el proceso de EE. Es decir, el uso de diversas técnicas puede requerir más tiempo del necesario para llegar a un resultado final, por lo que se debe evaluar de qué manera se mejoran los resultados al integrar varios métodos.

Como trabajos futuros se realizará una metodología para estimar el total de la DT de defectos en un entorno de DAS, calculando también la parte correspondiente a los intereses de la deuda. También, se pretende aplicar este enfoque híbrido de EE a proyectos de software en desarrollo para estimar su DT de defectos. Este trabajo puede aportar conocimiento últil para los desarrolladores de software y practicantes sobre cómo estimar la DT de defectos en el DAS.

## AGRADECIMIENTOS

Los autores agradecen al Tecnológico Nacional de México, IT de Apizaco, por el apoyo otorgado al proyecto de investigación del que forma parte este trabajo.

## REFERENCIAS

[1] Agile Manifesto. Manifesto for Agile Software Development [en línea]. Agile Manifesto, 2001 [recuperado el 5 de junio de 2024] de: https://agilemanifesto.org.

[2] Bogner J, Verdecchia R, Gerostathopoulos I. Characterizing Technical Debt and Antipatterns in AI-Based Systems: A Systematic Mapping Study. In: 2021 IEEE/ACM Int. Conf. on Technical Debt (TechDebt), 2021, 64-73. doi: 10.1109/TechDebt52882.2021.00016.

[3] Freire S, Rios N, Gutierrez B, Torres D, Mendonca M, Izurieta C, Seaman C, Spínola RO. Surveying Software Practitioners on Technical Debt Payment Practices and Reasons for not Paying off Debt Items. In: Proc. of the 24th Int. Conf. on Evaluation and Assessment in Software Engineering, April 15-17, 2020, 210-219. doi: 10.1145/3383219.3383241.

[4] Tang Y, Khatchadourian R, Bagherzadeh M, Singh R, Stewart A, Raja A. An Empirical Study of Refactorings and Technical Debt in Machine Learning Systems. In: 2021 IEEE/ACM 43rd Int. Conf. on Software Engineering (ICSE), Madrid, España, 2021, 238-250. doi: 10.1109/ICSE43902.2021.00033.

[5] Seaman C, Guo Y. Measuring and Monitoring Technical Debt. Advances in Computers, 2011, 82, 25-46. doi: 10.16/B978-0-12-385512-1.00002-5.

[6] Trendowicz A, Jeffery R. Software Project Effort Estimation: Foundations and Best Practice Guidelines for Success. Springer, 2014. doi: 10.1007/978-3-319-03629-8.

[7] Azzeh M, Nassif AB. Analyzing the relationship between project productivity and environment factors in the case points method. Journal of Software: Evolution and Process, 2017, 29, 1-19. doi: 10.1002/smr.1882.

[8] Boehm B. Software Engineering Economics. Prentice Hall, 1981.

[9] Boehm B. COCOMO II Model Definition Manual. Center for Software Engineering, University of Southern California, 2000. [recuperado el 12 de abril de 2024] de: http://csse.usc.edu/csse/research/COCOMOII/cocomo2000.0/CII_modelman2000.0.pdf

[10] Dekkers C. IFPUG 30 years - International Year of Software Measurement Timeline. Metric Views: Celebrating IFPUG's 30th Anniversary, 11(2), 4-5, August 2017 [recuperado el 10 de abril de 2024] de: https://www.ifpug.org.

[11] Commeyne C, Abran A, Djouab R. Effort Estimation with Story Points and COSMIC Function Points - An Industry Case Study [Position Paper]. Software Measurement News, 2016, 21(1), 25-36.

[12] Lavazza L, Meli R. An Evaluation of Simple Function Point as a Replacement of IFPUG Function Point. In: 2014 Joint Conf. of the Int. Workshop on Software Measurement and the Int. Conf. on Software Process and Product Measurement, 2014, 196-206. doi: 10.1109/IWSM.Mensura.2014.28.

[13] Lenarduzzi V, Lunesu I, Matta M, Taibi D. Functional size measures and effort estimation in agile development: A replicated study. In: Aalst W, Mylopoulos J, Rosemann M, Shaw MJ, Szyperski C. (Eds.), LNBIP: Vol. 1. Agile Software Development and Extreme Programming, 2015, 105-116. doi: 10.1007/978-3-319-18612-2_9.

[14] Rodríguez Sánchez E, Vázquez Santacruz E, Cervantes Maceda H. Estimación de esfuerzo en desarrollo de software ágil utilizando redes neuronales artificiales. Research in Computing Science, 2022, 151(7), 77-91.

[15] Nassif AB, Azzeh M, Capretz LF, Ho D. Neural network models for software development effort estimation: a comparative study. Neural Comput & Applic, 2016, 27, 2369–2381. doi: 10.1007/s00521-015-2127-1.

[16] Durán M, Juárez-Ramírez R, Jiménez S, Tona C. User Story Estimation based on the Complexity Decomposition using Bayesian Networks. In: Proc. of the Institute for System Programming of the RAS, 2021, 33(2), 77-92. doi: 10.15514/ISPRAS-2021-33(2)-4.

[17] Turic M, Celar S, Dragicevic S, Vickovic L. Advanced Bayesian Network for Task Effort Estimation in Agile Software Development. Applied Sciences, 2023, 13, 9465. doi: 10.3390/app13169465.

[18] Araújo RA, Oliveira ALI, Meira S. A class of hybrid multilayer perceptrons for software development effort estimation problems. Expert Systems With Applications. 2017, 90, 1–12. doi: 10.1016/j.eswa.2017.07.050.

[19] Kassem H, Mahar K, Saad AA. Story Point Estimation Using Ussue Reports With Deep Attention Neural Network. E-Informatica Software Engineering Journal, 2023, 17(1), 230104. doi: 10.37190/e-Inf230104.

[20] Pospieszny P, Czarnacka-Chrobot B, Kobylinski A. An effective approach for software project effort and duration estimation with machine learning algorithms. The Journal of Systems and Software, 2018, 137, 184-196. doi: 10.1016/j.jss.2017.11.066.

[21] Sharma A, Karambir. Empirical Validation of Random Forest for Agile Software Effort Estimation based on Story Points. International Journal of Engigeering Sciences & Research Technology, 2016, 5(7), 1437-1446.

[22] Thiago HAC, Oliveira ALI, da Silva FQB. Ensemble Effort Estimation: An updated and extended systematic literature review. Journal of Systems and Software, 2023, 195, 111542. doi: 10.1016/j.jss.2022.111542.

[23] Hacaloglu T, Demirors O. Challenges of Using Software Size in Agile Software Development: A Systematic Literature Review. In: Int. Workshop on Software Measurement IWSM Mensure Conference, 2018, 2207, 109-122. Recuperado de: http://ceur-ws.org/Vol-2207/IWSM_Mensura_2018_paper_9.pdf.

[24] Halkjelsvik T, Jørgensen M. Time Predictions Biases. In: A.T. Fornebu et al. (Eds), SIMULA SPRINGERBRIEFS ON COMPUTING Volume 5. Time Predictions: Understanding and Avoiding Unrealism in Project Planning and Everyday Life [eBook], 2018, 55-70. doi: 10.1007/978-3-319-74953-2.

[25] Jørgensen M. Unit effects in software project effort estimation: Work-hours gives lower effort estimates than workdays. The Journal of Systems and Software, 2016, 117, 274-281. doi: 10.1016/j.jss.2016.03.048.

[26] Tanveer B, Guzmán L, Engel UM. Effort estimation in agile software development: Case study and improvement framework. Journal of Software: Evolution and Process, 2017, 29:e1862. doi: 10.1002/smr.1862.

[27] Fernández-Diego M, Méndez ER, González-Ladrón-De-Guevara F, Abrahão S. An update on effort estimation in agile software development: a systematic literature review. IEEE Access, 2020, 8. doi: 10.1109/ACCESS.2020.3021664.

[28] Phannachitta P, Keung J, Monden A, Matsumoto K. A stability assessment of solution adaptation techniques for analogy-based software effort

estimation. Empirical Software Engineering, 2017, 22, 474-504. doi: 10.1007/s10664-016-9434-8.
[29] Huanca LM, Oré SB. Factores que afectan la precisión de la estimación del esfuerzo en proyectos de software usando puntos de casos de uso. Revista Ibérica de Sistemas y Tecnologías de Información, 2016, 21(3), 18-32. doi: 10.17013/risti.21.18-32.
[30] Vyas M, Bohra A, Lamba CS, Vyas A. A Review on Software Cost and Effort Estimation Techniques for Agile Development Process. International Journal of Recent Research Aspects, 2018, 5(1), 1-5.
[31] Freire S, Rios N, Pérez B, et al. Software practitioners' point of view on technical debt payment. The Journal of Systems & Software, 2023, 111554. doi: 10.1016/j.jss.2022.111554.
[32] Biazotto JP, Feitosa D, Avgeriou P, Nakagowa EY. Technical debt management automation: State of the arte and future perspectives. Information and Software Technology, 2024, 167, 107375. doi: 10.1016/j.infsof.2023.107375.
[33] Kontsevoi B, Terekhov S, Velesnitsky A. Practice of Technical Debt Management with TETRATM in Terms of Open-Source Project Assessment. In: 3rd Int. Conf. on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME), Tenerife, Canary Islands, Spain, 2023, 1-9. doi: 10.1109/ICECCME57830.2023.10252809.
[34] Ludwig J, Cline D. CBR Insight: Measure and Visualize Source Code Quality. In: IEEE/ACM Int. Conf. on Technical Debt (TechDebt), Montreal, QC, Canada, 2019, 57-58. doi: 10.1109/TechDebt.2019.00017.
[35] Didier DM. Una métrica para medir deuda técnica basada en el análisis de las más usadas. Caso de estudio del repositorio Square, 2024, [Tesis de grado de maestría], Repositorio institucional de la Universidad Nacional de Colombia. Recuperado de https://repositorio.unal.edu.co/handle/unal/86250
[36] Li Z, Yu Q, Liang P, Mo R, Yang C. Interest of Defect Technical Debt: An Exploratory Study on Apache Projects. In: IEEE Int. Conf. on Software Maintenance and Evolution (ICSME), Adelaide, SA, Australia, 2020, 629-639. doi: 10.1109/ICSME46990.2020.00065.
[37] Lenarduzzi V, Martini A, Taibi D, Tamburri DA. Towards Surgically-Precise Technical Debt Estimation: Early Results and Research Roadmpa. In: Proc. of the 3rd ACM SIGSOFT Int. Workshop on Machine Learning Techniques for Software Quality Evaluation (MaLTeSQuE), 2019, Tallinn, Estonia. New York, USA. doi: 10.1145/3340482.3342747.
[38] Pavlič L, Hliš T, Heričko M, Beranič T. The Gap between the Admitted and the Measured Technical Debt: An Empirical Study. Appied Sciences, 2022, 12, 7482. doi: 10.3390/app12157482.
[39] Melo A, Fagundes R, Lenarduzzi V, Barbosa Santos W. Identification and measurement of Requirements Technical Debt in software development: A systematic literature review. The Journal of Systems & Software, 2022, 194, 111483. doi: 10.1016/j.jss.2022.111483.
[40] Wang Q, Gou L, Jiang N, Che M, Zhang R, Yang Y, Li M. Estimating Fixing Effort and Schedule based on Defect Injection Distribution. Software process improvement and practice, 2008, 13, 35-50. doi: 10.1002/spip.366.
[41] Čelar S, Turić M, Vicković L. Method for personal capability assessment in agile teams using personal points. In: IEEE 22nd Telecommunications Forum Telfor Conference, 2014, 1134-1137. doi: 10.1109/TELFOR.2014.7034607.
[42] Shastri Y, Hoda R, Amor R. The role of the project manager in agile software development projects. Journal of Systems and Software, 173, 110871. doi: 1016/j.jss.2020.110871.

## ACERCA DE LOS AUTORES

María Guadalupe Medina Barrera es Doctora en Planeación Estratégica y Dirección de Tecnología por la Universidad Popular Autonóma del Estado de Puebla (UPAEP). Realizó estudios de Maestría en Ciencias en Ciencias Computacionales en el Centro Nacional de Investigación y Desarrollo Tecnológico (CENIDET) y estudios de Licenciatura en Informática en el Instituto Tecnológico de Tepic, Nayarit, México. Actualmente, cuenta con el reconocimiento como candidata en el Sistema Nacional de Investigadores del CONAHCYT, además del reconocimiento al Perfil Deseable y de ser miembro del cuerpo académico "Sistemas de Información", ambos reconocidos por PRODEP. Sus áreas de trabajo son: Gestión y Desarrollo de Proyectos de Software, Automatización de Procesos, Interfaces Humano Computadora y Reconocimiento de Patrones.

José Juan Hernández Mora es Ingeniero en Computación por la Universidad Autónoma de Tlaxcala. Tiene el grado de Maestro en Ciencias en Ciencias Computacionales por el Centro Nacional de Investigación y Desarrollo Tecnológico (CENIDET), de Cuernavaca, Morelos y Doctor en Excelencia Docente por la Universidad de los Ángeles.

Es Profesor con Perfil Deseable por parte del PRODEP, es líder del cuerpo académico "Sistemas de Información" y nivel de candidato del SNII del CONAHCYT. Sus líneas de investigación incluyen: Ingeniería de Software, Desarrollo de Aplicaciones de Tecnologías de la Información, Procesamiento Digital de Imágenes (PDI), Redes Neuronales Artificiales (RNA).