UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
Posgrado en Ciencia e Ingeniería de la Computación

Estimación de la productividad en equipos de desarrollo de software utilizando curvas de aprendizaje

TESIS
QUE PARA OPTAR POR EL GRADO DE
Maestro en Ciencia e Ingeniería de la Computación

PRESENTA:
Daniel Torres Robledo

DIRECTOR DE TESIS:
Dra. Hanna Jadwiga Oktaba
Facultad de Ciencias

CODIRECTOR DE TESIS:
Dr. Francisco Valdés Souto
Posgrado en Ciencia e Ingeniería de la Computación

Ciudad Universitaria, CDMX, DICIEMBRE, 2021

Agradecimientos

Dedico este espacio para agradecer a todos los profesores que han sido una parte importante de mi formación académica. En particular quiero agradecer de manera especial a mi tutora la Dra. Hanna Jadwiga Oktaba por su guía y apoyo en la realización de este trabajo, así como sus enseñanzas durante la licenciatura. Y a mi tutor el Dr. Francisco Valdés Souto quién me guió y animó a desarrollar artículos durante el Posgrado, además de escuchar mis problemas durante los tiempos de la pandemia.

También quisiera agradecer a mis sinodales Mtra. María Guadalupe Elena Ibargüengoitia González, Mtra. Ana Yuri Ramírez Molina y Mtro. Gustavo Arturo Márquez Flores por sus observaciones y correcciones que ayudaron a hacer de éste un mejor trabajo.

Finalmente quiero agradecer a la *Universidad Nacional Autónoma de México* y al *Posgrado en Ciencia e Ingeniería de la Computación* por brindarme la oportunidad de estudiar la maestría de Ciencia e Ingeniería de la Computación. Y al *CONACYT*, cuyo apoyo económico fue importante para llevar a cabo mis estudios.

Dedicatoria

A mi familia:

Argentina, Ramón y Anaid

que me dieron todo su apoyo durante mis estudios en el Posgrado.

A mis amigos:

Omar y Enrique

que me animaron cuando algunas materias se me hicieron demasiado difíciles.

Gracias a todos por su gran apoyo, sin ustedes no hubiera sido sencillo terminar la maestría.

Índice general

Lista de figuras

Lista de tablas

Glosario

AUC	Costo Promedio por Unidad	Average Unit Cost
CAC	Costo Acumulado Promedio	Cumulative Average Cost
CAT	Teoría del Promedio Acumulado	Cumulative Average Theory
CFP	Puntos Funcionales COSMIC	COSMIC Function Points
CMMI	Integración del modelo de madurez de capacidad	Capability Maturity Model Integration
COSMIC		Common Software Measurement International Consortium
FSM	Medición de Tamaño Funcional	Functional Size Measurement
FSMM	Método de Medición de Tamaño Funcional	Functional Size Measurement Method
FUR	Requerimientos Funcionales de Usuario	Funtional User Requirements
ISO	Organización Internacional para la Estandarización	International Organization for Standardization
LMP	Punto Medio del Lote	Lot Mid Point
ln	Logaritmo natural	Natural logarithm
MMRE	Media de la magnitud del error relativo	Mean Magnitude of the Relative Error
MRE	Magnitud del error relativo	Magnitude of the Relative Error
NFR	Requerimientos No Funcionales	Non-Functional Requirements
PDR	Proporción de Entrega del Producto	Product Delivery Rate
RMS	Raíz de la media del error cuadrático	Root Mean Square error
SWEBOK	Cuerpo de Conocimiento de Ingeniería de Software	Software Engineering Body of Knowledge
UT	Teoría Unitaria	Unit Theory
WH	Horas de trabajo	Work Hours

Introducción

Es sabido que la funcionalidad de un producto de software puede ser medida utilizando algún método de medición de software (*Functional Size Measurement Method, FSMM*), por ejemplo, *COSMIC ISO/IEC 19761*[1]. La posibilidad de medir el tamaño nos permite también determinar la productividad en el desarrollo de software.

Cuando un grupo de desarrollo de software trabaja durante un tiempo en un mismo contexto, se obtiene conocimiento acerca del dominio o contexto de desarrollo y las tecnologías utilizadas. Si bien intuitivamente esto se puede observar en proyectos individuales y/o escolares, también se ha observado este efecto en proyectos reales dentro del ámbito de la industria[25][52]. Cuándo esto ocurre, se generan economías de escala y el costo de producción se reduce, lo que significa que el esfuerzo requerido para desarrollar una unidad de software es menor que en periodos previos debido a la obtención de conocimiento y habilidades en el dominio, lo que significa un incremento en la productividad, que no es lineal y tiende a un límite.

En la literatura revisada referente al desarrollo de software, no se ha identificado una manera formal y bien fundamentada para estimar qué tanto mejora el factor de productividad para períodos posteriores de desarrollo.

Objetivo

El objetivo de esta tesis es determinar el grado de aprendizaje de una empresa desarrolladora de software considerando los periodos previos en los que ha trabajado bajo un mismo o similar contexto utilizando la herramienta de curvas de aprendizaje.

Para lograr esto, se llevó a cabo un caso de estudio que permitió determinar el grado de aprendizaje considerando los periodos previos en los que una empresa desarrolladora de software trabajó bajo el mismo contexto, para lo cual se propone el uso de curvas de aprendizaje. Utilizando el factor de productividad (*PDR*) determinado con base en el tamaño funcional y el esfuerzo en periodos previos, se busca determinar el factor de aprendizaje de manera confiable utilizando un mecanismo formal como es el análisis de curvas de aprendizaje.

Como complemento del primer caso de estudio, se llevó a cabo un segundo caso de estudio, cuyo objetivo es comparar los resultados obtenidos por las principales teorías de curvas de

aprendizaje (Teoría Unitaria y Teoría del Promedio Acumulado) utilizando los criterios de calidad generalmente aceptados dentro del área de estimación de software, para mostrar cuál de estos modelos presenta mejores resultados de manera más práctica.

Durante el desarrollo de los casos de estudio, se tuvo la necesidad de realizar cálculos de manera recurrente, por lo que, como parte de este trabajo se desarrolló una herramienta utilizando el lenguaje de programación *Python* que permita llevar a cabo el análisis de curvas de aprendizaje y mostrar los resultados.

Motivación

Al contratar un proveedor de software por un cierto número de periodos, es natural que el cliente busque tener una mejora en el factor de productividad, que sea reflejado en una disminución de costos o un incremento de la productividad relacionado a un mejor conocimiento del contexto obtenido a través del tiempo. Sin embargo, existe una dificultad al definir este valor de manera formal para los proyectos o desarrollos posteriores, ya que éste debe estar fundamentado y ser consistente, de lo contrario puede llegar a comprometer el éxito de los proyectos.

La teoría de curvas de aprendizaje es una herramienta para estimar costos recurrentes en un proceso de producción y está basada en la observación de que, al repetir una tarea, esta puede ser completada en periodos más cortos de tiempo o requerir menor esfuerzo para ser llevada a cabo, permitiendo determinar el nivel de aprendizaje en actividades, lo cual tiene como propósito realizar estimaciones[31].

Con este trabajo se busca estudiar el comportamiento del factor de productividad en el área de desarrollo de software, lo cuál permitirá mostrar si dentro de ésta área se exhibe un comportamiento similar a otras áreas donde exista producción, por ejemplo micro empresas[16], manufactura[47], construcción[34], aviones[8][10][56], barcos[35], camionetas[15], semiconductores[18], refinación de petróleo[21] y químicos[29].

Metodología

Para llevar a cabo la investigación de esta tesis se utilizó la metodología de caso de estudio propuesta por *Runeson*[38], ya que la calidad de los resultados obtenidos a través de esta metodología dependen, en gran medida, de conducir de manera adecuada y bien fundamentada el caso de estudio.

Se decidió desarrollar el programa utilizando el lenguaje de programación *Python* que permita realizar el análisis de las diferentes teorías de curvas de aprendizaje, el cual permita leer la información obtenida en este caso de estudio y preparar los datos para realizar los cálculos necesarios en ambas teorías, mostrar los resultados y las gráficas necesarias.

Contribución y relevancia

Para las organizaciones, poder estimar el valor del factor de productividad de un equipo de desarrollo de software que trabaja en un mismo contexto por varios periodos de tiempo, de manera formal y bien fundamentada, ayudará a tomar decisiones de negocio más precisas sin comprometer el éxito del proyecto.

Por otra parte, este trabajo busca exhibir la existencia del factor de aprendizaje en el área de desarrollo de software, ya que, como se mencionó anteriormente, no hay estudios acerca de las curvas de aprendizaje aplicadas en esta área.

También, muestra una aplicación del uso de métricas estandarizadas del tamaño funcional del software, dando una razón más para seguir estas prácticas en la industria.

Estructura de la tesis

Esta tesis está organizada con la siguiente estructura:

En el capítulo 1 se desarrolla el marco teórico, el cual incluye la teoría de los temas utilizados en esta tesis, que son:

- Medición de software utilizando el método *COSMIC*
- Productividad y *PDR*
- Estimación y criterios de calidad
- Curvas de aprendizaje y las dos teorías que se consideran para esta tesis: Teoría Unitaria y Teoría del Promedio Acumulado
- La metodología para llevar a cabo un caso de estudio en el área de Ingeniería de Software

En el capítulo 2 se describe cómo funciona el programa desarrollado y las bibliotecas utilizadas para realizar el análisis de las dos teorías de curvas de aprendizaje abordadas.

El capítulo 3 muestra los dos casos de estudio llevados a cabo para esta tesis, el primero tiene como objetivo estimar el *PDR* esperado para un siguiente periodo de desarrollo, mientras que el segundo realiza una comparación de los resultados obtenidos de aplicar la Teoría Unitaria y Teoría del promedio Acumulado y mostrar cuál de estos modelos presenta mejores criterios de calidad.

Por último, se presentan las conclusiones de la tesis, donde se muestra el *PDR* esperado para un siguiente periodo, y cuál de las teorías de curvas de aprendizaje presenta mejores resultados. También, se discuten algunas de las limitaciones de las curvas de aprendizaje y una breve descripción del trabajo futuro que se puede llevar a cabo a partir de los casos de estudio presentados.

En el apéndice A se encuentra el código fuente del programa desarrollado para realizar el análisis de las teorías de curvas de aprendizaje utilizando el lenguaje de programación *Python*.

Capítulo 1

Marco teórico

En este capítulo se desarrollan los elementos que se utilizarán para lograr cumplir los objetivos de esta tesis.

Para ésto se describe qué es la medición de software y el método *COSMIC ISO/IEC 19761*[1], que es un estándar, cuya medida de tamaño funcional de software permite realizar comparaciones entre diferentes proyectos de desarrollo de software.

La posibilidad de medir el tamaño funcional del software nos permite también medir la productividad en el desarrollo de software, por lo que se definen los conceptos de productividad (*SWEBOK*, Sección 12[13]) y factor de productividad[6].

Con el fin de poder evaluar los modelos de estimación es necesario validar su confiabilidad y precisión, por lo que se describen los criterios que se utilizan generalmente para analizar las relaciones de las variables de los modelos de productividad[6].

Posteriormente se desarrollan las dos principales teorías de curvas de aprendizaje (Teoría Unitaria y Teoría del Promedio Acumulado), con el fin de tener una herramienta que permita determinar el grado de aprendizaje y estimar el costo de producción para desarrollos posteriores.

Finalmente, se describe la metodología propuesta por *Runeson* para llevar a cabo un caso de estudio de manera formal y obtener resultados de calidad[38].

Describiendo la teoría de estos elementos tenemos las bases para comprender mejor el comportamiento de una empresa desarrolladora de software llevando a cabo un caso de estudio, en donde utilizando el esfuerzo y el tamaño funcional de software desarrollado en periodos previos para realizar un análisis de curvas de aprendizaje para estimar el *PDR* esperado para un siguiente periodo.

1.1 Medición de software

En el software existen dos tipos de factores que deben ser considerados para su desarrollo, requerimientos funcionales (*FUR*), que describen lo que el software debe hacer en términos de tareas y servicios, y requerimientos no funcionales (*NFR*), que describen atributos de calidad, el ambiente donde se ejecuta y los procesos y tecnologías utilizadas para desarrollar y mantener el software[44][46].

Así mismo, en un software existen dos tipos de factores que pueden ser medidos o cuantificados, factores técnicos y factores funcionales[24].

Los factores técnicos son útiles para los desarrolladores, sin embargo, debido a que utilizan un lenguaje técnico para ser descritos dificultan su entendimiento para usuarios y administrativos y se conocen de manera precisa en etapas avanzadas del proyecto, además de no existir un estándar internacional de cómo medirlos.

Por otro lado, los factores funcionales están descritos en un lenguaje de más alto nivel y no dependen de las tecnologías utilizadas, por lo que resultan útiles tanto para los usuarios y administrativos, como para el equipo de desarrollo, además de existir un estándar internacional que permite medirlos.

Históricamente se consideran dos generaciones de métodos de medición del tamaño funcional[7] (*Functional Size Measurement Method, FSMM*) (figura 1.1).

Actualmente solo existe un método de medición del tamaño funcional de segunda generación, el cual es *COSMIC ISO/IEC 19761*[1], que se generó a partir del estándar *ISO/IEC 14143* y la experiencia obtenida de los métodos de primera generación[55][40]. De la primera generación los cuatro métodos existentes son: FISMA (ISO/IEC 29881)[5], NESMA (ISO/IEC 24570)[4], IFPUG (ISO/IEC 20926)[2], MKII (ISO/IEC 20968)[3].

El método *COSMIC*, además de haber sido diseñado para su aplicación en la industria, busca resolver algunos de los problemas que presentaban los métodos de primera generación, como[7][22]:

- El manejo de conceptos existentes cuando fueron creados y aquellos que ya no se encuentran en uso
- El uso de escalas de medición poco convenientes
- Se basa en la representación de cualquier funcionalidad del software[46], por lo que no depende de análisis estadísticos basados en el esfuerzo como los métodos de primera generación[7], que no necesariamente son datos representativos de cualquier software a desarrollar.

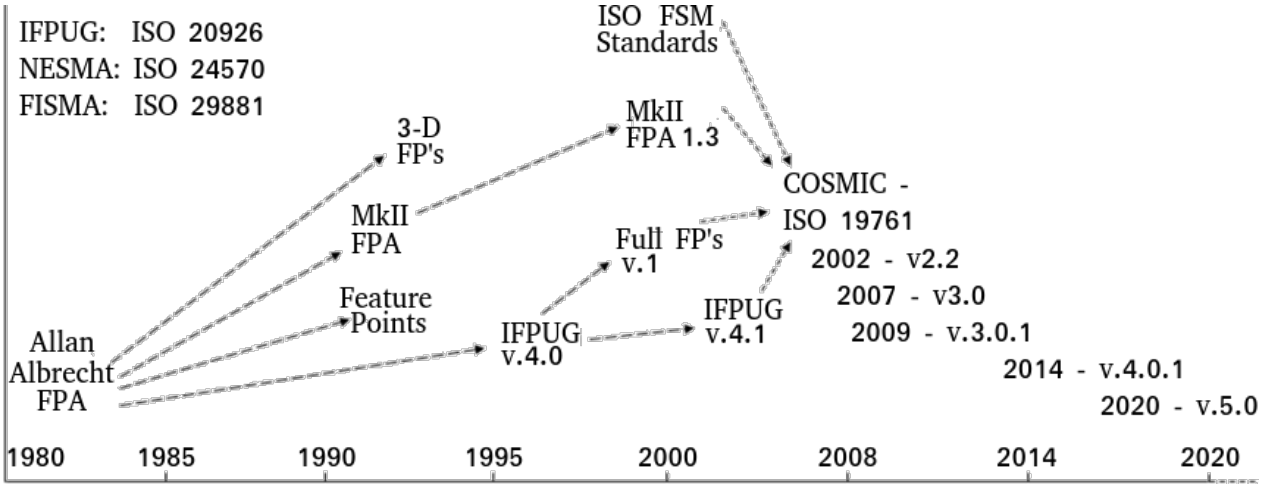

FIGURA 1.1: Evolución de los métodos de medición de tamaño funcional, imagen adaptada de [50].

1.1.1 Principios fundamentales del método *COSMIC*

El método *COSMIC* se conforma de 4 principios fundamentales que se han mantenido intactos desde su creación, lo que permite hacer comparación entre las mediciones a pesar de que se tengan varias versiones en la evolución del método.

Los principios son:

- La funcionalidad del software se conforma de procesos funcionales
- Los procesos funcionales consisten en subprocesos que mueven o manipulan datos
- Cada movimiento de datos mueve un grupo de datos que describe una cosa
- La manipulación de datos se considera incluida para cada movimiento de datos[1]

Conceptos clave de *COSMIC*

Con el fin de poder realizar una medición utilizando el método *COSMIC*, es necesario definir los siguientes conceptos mínimos[46]:

Propósito: Definir el por qué se requiere realizar una medición y para qué será utilizado el resultado.

Alcance: Es el conjunto de requerimientos funcionales que serán incluidos para realizar la medición.

[1] Toda manipulación de los datos realizada dentro del proceso funcional así como el procesamiento lógico relacionado para la entrada, salida, escritura o salida de éstos.

Usuarios funcionales: Son todos los usuarios que inician, proveen información o reciben información de cualquier proceso funcional identificado en los *FUR* dentro del alcance.

Proceso funcional: Es un componente elemental de un conjunto de *FUR*s que integran un conjunto único, cohesivo e independientemente ejecutable de movimientos de datos.

Cada proceso funcional identificado en el alcance del *FSM* debe ser derivado de al menos un *FUR* y ser iniciado por un movimiento de datos de entrada desde un usuario funcional a partir de un eventos desencadenante.

Objeto de interés: Un objeto de interés puede ser algo físico, conceptual o parte de algo conceptual dentro del universo de un usuario funcional.

Grupo de datos: Cada grupo de datos identificados dentro del *FSM* debe ser único y distinguible a través de su colección única de atributos de datos y está directamente relacionado a un objeto de interés descrito en los *FUR*.

Movimientos de datos: Cada proceso funcional identificado debe ser descompuesto en componentes de movimientos de datos (figura 1.2).

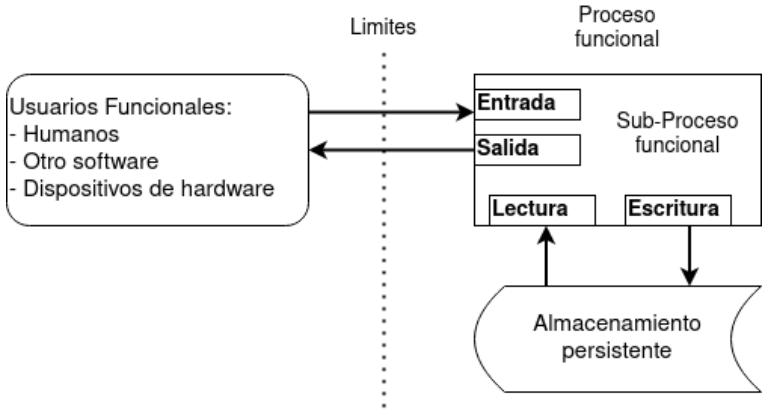

FIGURA 1.2: Los cuatro tipos de movimientos de datos dentro de un proceso funcional.

Los cuáles se clasifican en cuatro subtipos:

- Entrada (E): Recibe un solo grupo de datos que se origina desde un usuario funcional
- Salida (X): Manda atributos de un único grupo de datos a un usuario funcional
- Lectura (R): Recupera un único grupo de datos del almacenamiento persistente
- Escritura (W): Mueve un grupo de datos al almacenamiento persistente

CFP: es la unidad de medida estandarizada del método *COSMIC*, definida como el tamaño de un movimiento de datos, y permite la comparación de tamaño entre diferentes proyectos o procesos funcionales.

Conocer de manera cuantitativa el tamaño funcional del software permite estimar el esfuerzo, tiempo y costo para su desarrollo desde etapas iniciales del proyecto, conocer el alcance y manejar los riesgos derivados de los cambios[51], medir la productividad y velocidad con la que se desarrolló el producto o llegar a una etapa de madurez administrada cuantitativamente dentro del modelo CMMI (*(Capability Maturity Model Integration*)[45].

Además permite el desarrollo y aplicación de modelos de predicción para estimar el comportamiento de un proyecto o proyectos nuevos utilizando datos históricos relacionados al tamaño del software, ya que el tamaño, al estar basado en un estándar, puede ser comparado.

1.2 Economía de la Ingeniería de Software

La economía en la Ingeniería de Software consiste en la toma de decisiones en un contexto de negocio, donde el éxito de un proyecto de desarrollo de software consiste en una apropiada gestión de negocio (Capítulo 12: Economía de la Ingeniería de Software)[13].

Para esto, es necesario estudiar el valor, costos, recursos y las relaciones entre ellos dentro de un contexto o situación[13].

Por lo que para esta tesis se abordarán las relaciones entre producción y recursos, que son la productividad y el factor de productividad, con el fin de tomar decisiones con base en información estandarizada.

1.2.1 Productividad

De manera general, la Real Academia Española[36] define la productividad como: *"Relación entre lo producido y los medios empleados, tales como mano de obra, materiales energía, etc."*

De acuerdo con la guía del cuerpo de conocimiento de la Ingeniería de Software (*SWEBOK*, Sección 12-1-13 [13]), se define la productividad como la relación de la producción y los recursos necesarios para producirla desde una perspectiva económica[13][6]. Esta relación se representar con la siguiente ecuación:

$$Productividad = \frac{producción}{recursos} \tag{1.1}$$

En la Guía para la Medición de Software[44] definen la productividad en términos del tamaño funcional del software, cuya unidad esta dada en *CFP* y su relación con las horas de trabajo para producir ese tamaño funcional, como se muestra en la siguiente ecuación:

$$Productividad = \frac{tamanoFuncional}{horasDeTrabajo} \tag{1.2}$$

Analizando la ecuación de la productividad (ecuación 1.1) podemos observar que una productividad mayor a 1 significa que el valor de la producción es mayor que los recursos utilizados para desarrollarla, mientras que una productividad con valor mayor a 0 y menor a 1 muestra que se gastan más recursos contra lo que se produce.

1.2.2 Factor de productividad

El factor de productividad o proporción de entrega del producto (*Product Delivery Ratio, PDR*), es inverso a la producción por lo que representa cuántos recursos son necesarios para producir una unidad[6] y está representada por la siguiente ecuación:

$$PDR = \frac{recursos}{producción} \tag{1.3}$$

De las definiciones de productividad y *PDR*, de manera particular se puede observar que la producción, dentro del área de Ingeniería de Software, puede estar dada por el tamaño funcional del software a desarrollar, cuyas unidades son *CFP*, y el recurso necesario para desarrollar la funcionalidad puede ser el esfuerzo, cuyas unidades son generalmente horas-hombre.

1.3 Estimación

La estimación es importante en etapas tempranas de un proyecto de desarrollo de software, cuando la información no está completa, es ambigua y se espera que haya cambios, para poder tomar decisiones de negocio.

Una estimación es es una evaluación, realizada de manera formal, de los recursos que serán necesarios para producir una cierta cantidad de unidades[13], por ejemplo *CFP*, y se utilizan para tomar decisiones de negocio en etapas tempranas de un proyecto, cuando la información disponible es incompleta o ambigua.

De acuerdo al *SWEBOK*, existen cinco tipos de técnicas de estimación[13]:

- juicio de experto
- analogía
- estimación por partes
- métodos paramétricos
- métodos estadísticos

1.3.1 Estimación en el software

En el área de Ingeniería de Software, generalmente el costo para desarrollar un proyecto de software esta dado por el esfuerzo requerido para desarrollarlo[20].

Por otro lado, uno de los factores principales que determinan la precisión de la predicción del esfuerzo es tener un método adecuado para la medición del tamaño del software a desarrollar[26].

El rango estimado de costo (esfuerzo), para producir cierta cantidad de unidades de un proyecto puede ser determinado utilizando un modelo de estimación calibrado basado en datos históricos de tamaño y esfuerzo[13], siempre y cuando estén disponibles y puedan ser comparados.

1.3.2 Criterios de calidad para modelos de estimación

A continuación se presentan algunos de los criterios generalmente utilizados para analizar las relaciones de las variables de modelos de productividad creados a partir de un conjunto de datos[6], lo cual permite validar la confiabilidad y precisión de los modelos.

Coeficiente de determinación (R^2)

Describe el porcentaje de variación explicado por las variables independientes, cuyos valores se encuentran en el rango $[0 - 1]$.

Un valor cercano a 1 muestra una fuerte relación entre las variables independientes y las dependientes. Por otro lado, un valor cercano a 0 indica que la varianza no puede ser explicada por el modelo, y una baja relación las variables independientes con las dependientes.

Magnitud del error relativo (*MRE*)

La magnitud del error relativo (*Magnitude of the Relative Error - MRE*) indica la divergencia entre los valores estimados por el modelo y los valores reales, expresado en porcentaje.

$$MRE = \left| \frac{Actual - Estimado}{Actual} \right| \tag{1.4}$$

Media de la magnitud del error relativo (*MMRE*)

La media de la magnitud del error relativo (*Mean Magnitude of the Relative Error - MMRE*) es el promedio de la magnitud de los errores relativos.

$$MMRE = \frac{\sum_{i=1}^{N} MRE_i}{N} \tag{1.5}$$

Nivel de predicción

El nivel de predicción de un modelo, en la siguiente ecuación K la cantidad de elementos en el conjunto que tienen un valor de *MRE* menor o igual a un porcentaje dado, generalmente 25%, y N es el total de elementos en el conjunto.

$$Pred(l) = \frac{K}{N} \tag{1.6}$$

Raíz cuadrada de la media del error cuadrático (*RMS*)

La raíz de la media del error cuadrático (*Root Mean Square error - RMS*) se utiliza para comparar las diferencias entre dos valores. En la fórmula se puede observar que al elevar al cuadrado estas diferencias, los valores grandes tendrán un mayor impacto en el resultado.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(Actual_i - Estimado_i)^2} \tag{1.7}$$

1.4 Curvas de aprendizaje

El análisis de curvas de aprendizaje es una herramienta desarrollada para estimar costos recurrentes en procesos de producción[31], donde el factor dominante es el trabajo directo requerido para completar una tarea o producto.

Existen dos teorías predominantes[43]:

- Teoría Unitaria (*UT*), propuesta por *J. R. Crawford* en 1947 [31](Capítulo 11).
- Teoría del Promedio Acumulado (*CAT*), propuesta por *T. P Wright* en 1936 [56][31](Capítulo 11)

Ambas teorías están basadas en el hecho de que al repetir una tarea varias ocasiones se obtiene experiencia, permitiendo llevarla a cabo más rápido la siguiente ocasión que se realice[31], lo que significa un incremento en la productividad. Situación que se presenta de manera recurrente en el desarrollo de software, donde las personas, al pasar más tiempo desarrollando un mismo proyecto o en un mismo contexto, ganan experiencia dentro de ese dominio.

Es importante notar que no es posible llegar a un punto en donde se haya mejorado tanto al realizar una tarea que no tenga un costo al llevarla a cabo, ya que siempre requiere de un esfuerzo y tiempo mínimo el llevarla a cabo y la reducción de costos es cada vez más lenta, porque se ve reflejada cada que se duplica la cantidad producida.

En contextos donde existen variaciones considerables en costos o diseño se recomienda el uso de la Teoría del Promedio Acumulado ya que permite reducir el riesgo de estimación. Por otro lado, cuando los datos de los costos disponibles son precisos se recomienda utilizar la Teoría Unitaria.

1.4.1 Teoría Unitaria

En 1947 *J.R Crawford* llevó a cabo un estudio en el área de producción de aviones durante la segunda guerra mundial, con el fin de validar la Teoría del Promedio Acumulado de las curvas de aprendizaje anteriormente propuesta por *T. P. Wright* en 1936[56], la cual, como menciona *Gregory k. Mislick*[31] en el capítulo 11 referente a la Teoría Unitaria, la definió de la siguiente manera:

"Si existe aprendizaje en el proceso de producción, el costo de una unidad duplicada es igual al costo de la unidad no duplicada por la pendiente de la curva de aprendizaje".

La curva de aprendizaje, utilizando la Teoría Unitaria está definida por la siguiente ecuación:

$$Y(x) = A * x^b \tag{1.8}$$

Donde:

- $Y(x)$ es el costo de producción de la unidad número x.
- A es el costo de producción la primera unidad.
- x es la unidad número x.
- b es una constante que representa la pendiente de la curva de aprendizaje.

De acuerdo a la definición de curva de aprendizaje en esta teoría, cada que se duplica el número de unidades producidas, el costo por unidad se reduce en un porcentaje fijo ($slope$), que representa la pendiente de la curva, como se puede observar en la ecuación 1.8.

Sea $Y(x)$ el costo de la unidad x, la siguiente ecuación representa la definición del párrafo anterior:

$$Y(2n) = Y(n) * slope \tag{1.9}$$

Al despejar la pendiente $slope$ de la ecuación1.9, tenemos lo siguiente:

$$slope = \frac{Y(2n)}{Y(n)} = \frac{A * (2n)^b}{A * (n)^b} = 2^b \tag{1.10}$$

Finalmente para obtener el valor de b, tomamos el logaritmo natural de ambos lados de la ecuación 1.10:

$$\begin{aligned} ln(slope) &= ln(2^b) = b * ln(2) \\ b &= \frac{ln(slope)}{ln(2)} \end{aligned} \tag{1.11}$$

Ya que la ecuación 1.8 representa una curva, es necesario realizar una transformación a los datos para poder aplicar una regresión lineal. Lo cual se logra aplicando la función de logaritmo natural, $ln(n)$, en ambos lados de la ecuación, como se muestra a continuación:

$$\begin{aligned} ln(Y(x)) &= ln(A * x^b) \\ ln(Y(x)) &= ln(A) + ln(x^b) \\ ln(Y(x)) &= ln(A) + b * ln(x) \end{aligned} \tag{1.12}$$

Por lo que la ecuación 1.12 se puede reescribir como $Y(x)' = A' + b * x'$ que es una ecuación lineal y es posible realizar una regresión lineal.

Estimación de costos por lote

Sin embargo, debido a que el costo de producción de cada unidad es rara vez reportado no es posible el uso de la Teoría Unitaria, una opción viable es realizar la estimación de la producción por lotes.

Para estimar el costo de producción por lotes se utiliza la ecuación 1.8, sin embargo, es necesario ajustar la información para cada lote. Considerando que el análisis de Curvas de Aprendizaje requiere el conocimiento del costo asociado a la producción de cada unidad, estos valores estarán representados por el punto medio de cada lote (*Lot Mid Point, LMP*) y el costo promedio por unidad dentro de éste (*Average Unit Cost, CAT*).

El punto medio del lote puede ser estimado de manera directa utilizando una aproximación, utilizando las ecuaciones 1.13, 1.14, y 1.15.

Para obtener el valor del primer lote:

$$If\ lotSize < 10,\ then\ LMP = \frac{lotSize}{2} \tag{1.13}$$

$$If\ lotSize \geq 10,\ then\ LMP = \frac{lotSize}{3} \tag{1.14}$$

Para cualquier lote posterior al primero:

$$LMP = \frac{F + L + 2\sqrt{F \times L}}{4} \tag{1.15}$$

Donde:

- F es la primera unidad del lote.
- L es la última unidad del lote.

Finalmente, para calcular el costo promedio por cada unidad (*AUC*) de cada lote, se divide el número de unidades dentro del lote por el costo de haberlas desarrollado, como se muestra en la ecuación 1.16:

$$AUC = \frac{TotalLotCost}{LotSize} \tag{1.16}$$

De esta ecuación se puede observar que el costo total del lote son los recursos utilizados para producir el tamaño del lote, por lo *AUC* es igual al *PDR* (ecuación 1.3).

1.4.2 Teoría del Promedio Acumulado

Durante la primera guerra mundial, las compañías que producían aviones mostraron interés en la reducción de costos de producción, que mostraba un patrón regular, por lo que en 1936 *T. P. Wright*[56], encontró que estas compañías podían producir un mayor número de unidades utilizando el mismo esfuerzo, por lo que definió esta teoría como:

"Si existe aprendizaje en el proceso de producción, el costo acumulado promedio de una unidad duplicada es igual al costo acumulado promedio de la unidad no duplicada por la pendiente de la curva de aprendizaje".

Debido a que, al igual que en la Teoría Unitaria, esta teoría está definida por una curva de aprendizaje, sin embargo, a diferencia de la ecuación 1.8 de la *UT*, en la ecuación 1.17 la variable independiente x denota la cantidad acumulada de unidades producidas hasta x.

La curva de aprendizaje, utilizando Teoría del Promedio Acumulado está definida por la siguiente ecuación:

$$\bar{Y}(x) = A * x^b \tag{1.17}$$

Donde:

- $Y\bar{(}x)$ es el costo promedio acumulado de N unidades.
- A es el costo teórico para producir la primera unidad.
- x es el número acumulado de unidades producidas.
- b es una constante que representa la pendiente de la curva de aprendizaje.

La principal diferencia en la ecuación que define las curvas de aprendizaje utilizando la Teoría del Promedio Acumulado contra la ecuación que define la Teoría Unitaria (1.8), es que en la Teoría del Promedio Acumulado se calcula el Costo Promedio Acumulado (*Cumulative Average Cost, CAC*) de todas las unidades producidas, mientras que en la Teoría Unitaria solo se utiliza el costo de cada unidad o lote.

Debido a esta diferencia con respecto a la Teoría Unitaria, en el eje X se coloca la cantidad acumulada de producción, mientras que en el eje Y está el Costo Promedio Acumulado (*CAC*), el cual se calcula utilizando la ecuación 1.18.

$$CAC = \frac{CumulativeCost}{CumulativeQuantity} \tag{1.18}$$

De esta ecuación se puede observar que el costo acumulado son los recursos utilizados para producir la cantidad de unidades acumulada, por lo *CAC* es igual al *PDR* (ecuación 1.3).

Por la definición de curva de aprendizaje para esta teoría, tenemos que cada que se duplica el número acumulado de unidades producidas, el costo promedio por unidad se reduce en un porcentaje fijo ($slope$), que representa la pendiente de la curva, como se puede observar en la ecuación 1.17.

Sea $\bar{Y}(x)$ el costo promedio de las unidades 1 a x, la siguiente ecuación representa la definición del párrafo anterior:

$$\bar{Y}(2n) = \bar{Y}(n) * slope \tag{1.19}$$

De manera similar a las ecuaciones de $slope$ (ecuación 1.10 y b (ecuación 1.11) definidas en la Teoría Unitaria, tenemos las ecuaciones 1.20 y 1.21 para $slope$ y b en ésta teoría:

$$slope = 2^b \tag{1.20}$$

$$b = \frac{ln(slope)}{ln(2)} \tag{1.21}$$

Ya que la ecuación 1.17 representa una curva, de manera similar a la Teoría Unitaria, se deben transformar los datos utilizando la función de logaritmo natural, $ln(n)$ para poder aplicar una regresión lineal, quedando la siguiente ecuación:

$$ln(\bar{Y}(x)) = ln(A) + b * ln(x) \tag{1.22}$$

Por lo que la ecuación 1.22 se puede reescribir como $\bar{Y}'(x) = A' + b * x'$ que es una ecuación lineal y es posible realizar una regresión lineal.

Esta teoría muestra mejores resultados cuando el costo inicial para producir una unidad puede sufrir grandes cambios, debido a variaciones como: cambios de diseño en etapas tempranas del proyecto, plazos de entrega cortos o una base de proveedores inadecuada.

Estimación de costos por lote

Al igual que en la Teoría Unitaria, debido a que el costo de producción de cada unidad es rara vez reportado, pero sí se tiene el costo de producción y cantidad de unidades producidas para cada lote.

Por lo que se pueden obtener las ecuaciones para estimar el costo de producción promedio por lote utilizando la ecuación 1.17. Ya que $\bar{Y}(x) = A * x^b$ representa el costo promedio de las primeras x unidades, entonces el costo total de x unidades se puede obtener al multiplicar el costo promedio de x unidades por el total de x unidades, como muestra la siguiente ecuación:

$$CT(x) = A * x^b * x = A * x^{b+1} \tag{1.23}$$

Sin embargo, al igual que en la Teoría Unitaria se puede obtener el costo de cada unidad, para obtener el costo de una unidad x, se utiliza la derivada de la ecuación 1.23, que es la pendiente instantánea de la curva en ese punto, como se muestra en la siguiente ecuación:

$$CostUnit(x) = CT(x)' = (b+1) * A * x^b \tag{1.24}$$

Finalmente, para calcular el costo total estimado de un lote específico, se requiere tener la unidad inicial (f) y final (l), para utilizar la siguiente ecuación:

$$\begin{aligned} CT(f,l) &= CT(l) - CT(f-1) \\ &= A * l^{b+1} - A * (f-1)^{b+1} \\ &= A * [l^{b+1} - (f-1)^{b+1}] \end{aligned} \tag{1.25}$$

1.4.3 Comparación entre la Teoría Unitaria y la Teoría del Promedio Acumulado

Comparando las curvas de aprendizaje, representadas por las ecuaciones que definen a cada teoría, Teoría Unitaria y Teoría del Promedio Acumulado, es posible notar que a pesar de que disminuyen su costo en una cantidad fija, debido al aprendizaje dado por el parámetro b de sus respectivas ecuaciones, cada que el número de unidades es duplicado, la segunda Teoría siempre estará por arriba de la Teoría Unitaria.

Para esto, primero hay que notar que para graficar estas curvas, para cada teoría tenemos los siguientes valores para los ejes x y y:

- Teoría Unitaria (*UT*)
  - X-axis es el punto medio del lote
  - Y-axis es el costo promedio de cada unidad dentro del lote (*PDR*)
- Teoría del Promedio Acumulado (*CAT*)
  - X-axis es la cantidad acumulada
  - Y-axis es el costo promedio acumulado (*PDR*)

Por lo que una gráfica que muestre las curvas de cada teoría, utilizando los mismos datos se puede ver en la figura 1.3:

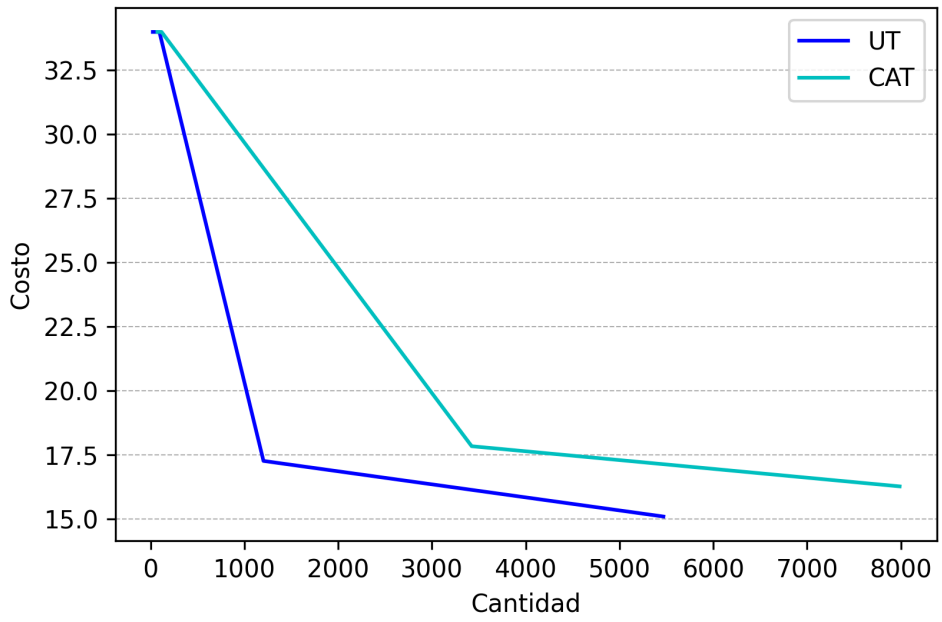

FIGURA 1.3: Comparación de la UT y CAT utilizando los mismos datos.

Esto sucede debido a que la Teoría del Promedio Acumulado utiliza el costo promedio de todas las unidades producidas hasta la número n, por lo que, para observar un cambio considerable debe existir una gran variación en los costos de producción a lo largo del tiempo, además de que, la reducción de costos, como menciona la definición de esta teoría, se da cada que el número de unidades acumuladas es duplicado, a diferencia de la Teoría Unitaria, que la reducción se ve reflejada cada que el número de unidades es duplicada, lo cual se puede observar en el eje x

de la figura 1.3, a pesar de utilizar los mismos datos, la *CAT* tiene un valor mayor para la última unidad producida que la *UT*.

En ambas teorías, para estimar el costo, es necesario contar con los valores de la unidad inicial y final de los lotes a producir, así como sus respectivos costos.

1.5 Caso de estudio

El caso de estudio es una metodología de investigación adecuada para el área de Ingeniería de Software ya que tiene como objetivo principal el estudio de fenómenos contemporáneos, que son difíciles de estudiar de manera aislada y proveen un conocimiento más profundo acerca del tema estudiado[38][23].

A diferencia de los estudios empíricos, analíticos y controlados, los casos de estudio han sido criticados por tener menor valor, estar sesgados por los investigadores, etc[38]. Por lo que es necesario aplicar las metodologías de investigación adecuadas, ya que la calidad de los resultados dependen en gran medida de llevar a cabo el caso de estudio de manera adecuada y bien fundamentada, así como reconocer que el conocimiento y los resultados obtenidos son más que solo estadísticas[17][27].

Dentro del área de la Ingeniería de Software generalmente se tiene como objeto de estudio a corporaciones privadas o agencias públicas que desarrollan software.

Dado que la Ingeniería de Software es un área multidisciplinal llevada a cabo por individuos, grupos y organizaciones, las preguntas de investigación de esta área son adecuadas para una investigación utilizando la metodología de caso de estudio.

1.5.1 Proceso de investigación de un caso de estudio

Cuando se lleva a cabo un caso de estudio, de acuerdo a la metodología que propone *Runeson*, se deben seguir cinco procesos principales[38]:

1. Diseño y planeación del caso de estudio
2. Recolección de datos
3. Recolección de las evidencias
4. Análisis de los datos obtenidos
5. Reporte de resultados

Es importante tener en cuenta que la recolección de datos y el análisis puede ser llevado a cabo de manera incremental.

Diseño y planeación del caso de estudio

De acuerdo a Robson[37], la planeación de un caso de estudio debe contener al menos los siguientes elementos: objetivo, definición del caso, teoría, preguntas de investigación, métodos y estrategia de selección.

El objetivo del caso de estudio puede ser exploratorio, descriptivo o explicativo[38][23] acerca de un fenómeno contemporáneo en el mundo real, y las preguntas de investigación definen qué es lo que se quiere lograr responder o tener conocimiento más profundo a partir del caso estudiado. Estas preguntas pueden modificarse a lo largo del desarrollo del caso de estudio con el fin de hacerlas más precisas[9].

El uso de teorías para llevar a cabo la investigación no está bien establecido en el área de Ingeniería de Software[20][41], sin embargo, definir un marco de referencia del estudio permite tener un contexto claro del caso de estudio, lo cual facilita la investigación y la revisión de los resultados[38].

Para definir la fuente y el tipo de datos, es necesario tener en cuenta qué tipo de resultados se espera obtener: típicos, críticos, relativos o únicos[17]. Mientras que en un caso de estudio comparativo los datos deben ser seleccionados con el fin de mostrar alguna variación en sus propiedades[38]. Sin embargo, en la práctica, los datos son seleccionados basados en la disponibilidad[12].

Dentro de esta fase también es necesario tener en cuenta ciertas consideraciones éticas[42], por lo que se debe definir de manera explícita qué información se puede hacer pública y cómo serán manejados los datos e información sensible, así como los resultados obtenidos.

Recolección de datos

Existen diferentes fuentes de donde obtener los datos para el caso de estudio, y de acuerdo a Lethbridge[28] las técnicas para obtenerlos se pueden clasificar en tres niveles.

- Las de primer nivel son métodos directos, en donde el investigador está en contacto directo con la fuente y recaba datos en tiempo real.
- Las de segundo nivel, son indirectas, donde la información es obtenida de manera automatizada.
- Mientras que en las de tercer nivel se utilizan datos ya existentes.

Análisis de los datos obtenidos

De acuerdo a los datos recabados para el caso de estudio, su análisis puede ser:

- Cuantitativo, que consiste generalmente de análisis de estadísticas descriptivas, de correlación y desarrollo de modelos de predicción.
- Cualitativo, el cual tiene como objetivo derivar conclusiones a partir de los datos, mantenido claras las evidencias y los pasos seguidos durante el análisis.

Reporte de resultados

El reporte del caso de estudio debe contener toda la información necesaria como el tema que se aborda, contexto, teoría, preguntas de investigación, cómo es que llevó a cabo y mostrar los datos relevantes y resultados a los que se llegó para fundamentar las conclusiones[37].

Capítulo 2

Implementación de algoritmos para el análisis de curvas de aprendizaje

Con el fin de realizar el análisis y comparación de los resultados obtenidos al aplicar las dos teorías de curvas de aprendizaje descritas en el capítulo 1 (Teoría Unitaria y Teoría del Promedio Acumulado), se desarrolló un programa para leer la información obtenida en este caso de estudio y preparar los datos para realizar los cálculos necesarios en ambas teorías, los resultados y gráficas necesarias.

En este capítulo se describe de manera general la función del programa, los datos que recibe y cómo es que realiza los cálculos para obtener la curva de aprendizaje de las teorías definidas en el capítulo 1.

2.1 Tecnologías utilizadas

Para desarrollar este programa se utilizó la versión $3.8.5$ de *Python3*[33], el cual es un lenguaje de programación interpretado de propósito general, el código fuente se encuentra de manera completa en el apéndice A.

2.1.1 Bibliotecas

Para desarrollar el programa se requirió el uso de las siguientes bibliotecas:

Matplotlib

La biblioteca *Matplotlib*[30], la cual permite crear gráficas estáticas, animadas y visualizaciones interactivas utilizando *Python* además de ser de código libre.

SciPy

Es un ecosistema de código libre para matemáticas, ciencia e ingeniería[39], el cual contiene funciones para realizar integrales, interpolación, álgebra lineal y estadísticas.

Pandas

Dentro del ecosistema de *SciPy* se encuentra el módulo de Pandas[32], el cual es de código abierto y contiene estructuras de datos y herramientas de análisis para el lenguaje de programación *Python*.

2.2 Uso del programa

Una vez instaladas las bibliotecas anteriormente mencionadas, se debe ejecutar el programa con el intérprete *python3* desde la línea de comandos:

$$python3\ curvasDeAprendizaje.py$$

Para utilizar otra base de datos es necesario editar el código fuente del programa, en la línea 123 se pasa como argumento a la función *read_data* la ruta del archivo de la nueva base de datos.

$$data = read\_data('data.csv')$$

A continuación se describe el funcionamiento del programa y el formato de la base de datos.

2.3 Preparación de los datos

Los datos deben estar en formato csv (valores separados por coma) para que el programa pueda leer el archivo. En este caso, cómo se encontraban originalmente en formato de *Excel*, fue necesario exportarlos a csv.

2.3.1 Formato inicial de los datos

Las columnas que debe contener el archivo csv con los datos son:

- **periodo**: es un valor que indica en que bloque de tiempo se desarrolló un proyecto, por ejemplo número de semestre.
- **proyecto**: identificador del proyecto que se está desarrollando, esto es debido a que varios equipos pueden estar trabajando en un mismo proyecto.
- **esfuerzo**: recursos utilizados para desarrollar el tamaño funcional de ese proyecto en el periodo indicado, por ejemplo cantidad de horas.

- ***pdr***: proporción de entrega del producto, cuantos recursos son necesarios para desarrollar una unidad de *CFP*.

Un ejemplo de el contenido de este archivo se muestra en la tabla 2.1.

TABLA 2.1: Ejemplo del formato que debe tener un archivo de entrada para el programa.

periodo	proyecto	esfuerzo	pdr
1	01	1712	16.7
2	02	400	6.7
2	03	174	2.6
3	01	257	8.6

2.4 Procesamiento de los datos

Lo primero es leer los datos para realizar el análisis, utilizando la función $read\_data$, la cual permite filtrar algunos periodos en caso de que sea necesario, utilizando el índice del mismo.

2.4.1 Cálculo de productividad y *PDR*

Dado que los datos solo tienen la información acerca del *PDR* y del esfuerzo, se puede calcular de manera sencilla el tamaño funcional despejando el valor de $production$ de la ecuación 1.3 dando como resultado la siguiente ecuación:

$$CFP = \frac{effort}{PDR} \tag{2.1}$$

Para obtener la productividad, que es el inverso del *PDR* que se encuentra en los datos, basta con utilizar la siguiente ecuación:

$$Productivity = \frac{1}{PDR} \tag{2.2}$$

O bien, utilizando la ecuación 1.1 ya que se haya calculado el tamaño funcional [*CFP*].

Esto se realiza también dentro de la función utilizada para leer los datos, $read\_data$, para así tener todos los datos necesarios para el análisis disponibles.

2.4.2 Cálculo de curvas de aprendizaje con Teoría Unitaria

Para el análisis de la Teoría Unitaria por lotes, es necesario calcular el tamaño de cada uno de los lotes, que en este caso están dados por el semestre en que fueron desarrollados. El tamaño del

lote esta dado por todos los proyectos desarrollados en cada periodo y el esfuerzo son todas las horas invertidas en desarrollar esa cantidad de *CFP*.

Los cálculos se encuentran dentro del bloque de cálculo de valores, que inicia en la línea 136 del código (Anexo A), en donde se puede observar el uso de las ecuaciones 1.13, 1.14, y 1.15 para calcular el punto medio de cada lote (*LMP*), lo cual se realiza con la función lmp, y la ecuación 1.16 para calcular el costo promedio por unidad dentro de cada uno (*AUC*), además de aplicar la función de logaritmo natural para *LMP* y *AUC*.

Posteriormente, en el bloque de Teoría Unitaria, que inicia en la línea 163 del código (Anexo A), se realiza la regresión lineal con los valores resultantes de aplicar el logaritmo natural del *LMP* y del *AUC*, como se describe en el capítulo 1.

Con el resultado de ésto, tenemos la pendiente, que es el porcentaje que define la curva de aprendizaje, y utilizando la ecuación 1.11 podemos obtener el valor de b que será utilizado en la ecuación que define la curva para esta teoría (ecuación 1.8)

Por otro lado, para obtener el valor de A, dado que el resultado de la regresión lineal fue utilizando los datos a los que se aplicó el logaritmo natural, es necesario utilizar una exponencial del intercepto:

$$A = e^{intercept} \tag{2.3}$$

Finalmente, a manera de ejemplo, se propone un lote a desarrollar para un siguiente periodo definiendo un tamaño funcional a desarrollar de maneara arbitraria, se calcula su primera y última unidad, así como su *LMP*. Y utilizando la ecuación 1.8, podemos obtener el *AUC* estimado para este nuevo lote.

2.4.3 Cálculo de curvas de aprendizaje con Teoría del Promedio Acumulado

Para el análisis de la Teoría del Promedio Acumulado, es necesario calcular el tamaño y costo acumulado por cada lote, que en este caso están dados por el semestre en que fueron desarrollados. El tamaño del lote está dado por todos los proyectos desarrollados en cada periodo y el esfuerzo son todas las horas invertidas en desarrollar esa cantidad de *CFP*.

Los cálculos se encuentran dentro del bloque de cálculo de valores, al igual que en la Teoría Unitaria inicia en la línea 136 del código (Anexo A), en donde se puede observar el uso de las ecuación 1.18 para calcular el costo promedio acumulado (*CAC*) y el cálculo iterativo de la cantidad acumulada *CumQuantity*, además de aplicar la función de logaritmo natural para *CAC* y *CumQuantity*.

Posteriormente, en el bloque de Teoría del Promedio Acumulado, que inicia en la línea 200 de l código (Anexo A), se realiza la regresión lineal con los valores resultantes de aplicar el logaritmo natural del *cac* y del *CumQuantity*, como se describe en el capítulo 1.

Con el resultado de ésto, tenemos la pendiente, que es el porcentaje que define la curva de aprendizaje, y utilizando la ecuación 1.21 podemos obtener el valor de b que será utilizado en la ecuación que define la curva para esta teoría (ecuación 1.17)

Por otro lado, para obtener el valor de A, similar al caso de la Teoría Unitaria, es necesario utilizar una exponencial del intercepto (ecuación 2.3)

Finalmente, a manera de ejemplo, se propone un lote a desarrollar para un siguiente periodo definiendo un tamaño funcional a desarrollar de maneara arbitraria, se calcula su primera y última unidad acumuladas, así como el costo acumulado promedio por unidad (*CAC*). Y utilizando la ecuación 1.25, podemos estimar el costo de este lote.

2.4.4 Criterios de calidad

Una vez obtenidos los valores de estimación por cada teoría, utilizando la función get_errors se calculan los siguientes criterios de calidad:

- Magnitud relativa del error (MRE)
- Media de la magnitud relativa del error (MMRE)
- Raíz cuadrada de la media del error cuadrático (RMS)
- Nivel de predicción (Pred 25%)

Utilizando el valor real del costo y el valor estimado utilizando las ecuaciones definidas por las curvas de aprendizaje y los valores obtenidos durante el análisis.

Este programa fue desarrollado con el fin de tener una herramienta que permitiera aplicar las teorías de curvas de aprendizaje descritas en el capítulo 1 en un caso de estudio y realizar comparaciones de los resultados obtenidos por los modelos de cada teoría utilizando los criterios de calidad.

Capítulo 3

Aplicación de las teorías de curvas de aprendizaje: casos de estudio

En este capítulo se describen los dos casos de estudio llevados a cabo durante la maestría.

En el primer caso de estudio se plantea la pregunta ¿cuál será la mejora en términos de *PDR* para el siguiente bloque de proyectos, dado el aprendizaje que ha tenido el proveedor?, buscando así, estimar de manera formal y confiable el *PDR* que se espera que una empresa de desarrollo de software tenga, utilizando el esfuerzo y tamaño funcional desarrollado en periodos anteriores y llevando a cabo un análisis de curvas de aprendizaje, desde la perspectiva de la Teoría Unitaria. Este artículo fue publicado y presentado en el congreso IWSM MENSURA + CNMES 2020[1][49].

Mientras que el segundo caso de estudio busca extender el primero, siendo la pregunta de investigación ¿cuál teoría de las definidas para las curvas de aprendizaje presenta mejores resultados, Teoría Unitaria o Teoría del Promedio Acumulado?. Este artículo fue enviado a la revista *Programming and Computer Software - Special Issue 2021* y actualmente ya fue aceptado para su publicación, de acuerdo a la revista, éste será publicado en su numero especial 2021[48].

3.1 Primera iteración: estimación de productividad del siguiente lote

Para conducir este caso de estudio en un contexto del mundo real, se siguió la metodología de cinco fases propuesta por Runeson[38].

En la tabla 3.1 se muestran los puntos más importantes, y de manera resumida, de cada fase de este caso de estudio.

[1] 5^o National Congress of Software Measurement and Estimation joint 30^{th} International Workshop on Software Measurement (IWSM) y el 15^{th} International Conference Software Process and Product Measurement (MENSURA).

TABLA 3.1: Metodología para llevar a cabo el caso de estudio.

1	Diseño de caso de estudio	**P1:** ¿Cuál será la mejora en términos de *PDR* para el siguiente bloque de proyectos, dado el aprendizaje que ha tenido el proveedor? **Objeto de estudio:** 21 proyectos desarrollados a lo largo de 4 semestres
2	Recolección de datos	**Datos de interés:** tamaño funcional del software, esfuerzo
3	Recolección de evidencias	**Métricas de la evidencia:** *CFP*, *PDR*, productividad
4	Análisis de datos recolectados	**Análisis cuantitativo:** cálculo del punto medio del lote (*LMP*), costo promedio por unidad (*AUC*), pendiente de la curva de aprendizaje
5	Reporte de resultados	**Reporte del caso de estudio:** metodología y resultados

3.1.1 Diseño de caso de estudio

El análisis de curvas de aprendizaje en este documento tiene como objetivo calcular el grado de aprendizaje que se puede solicitar de manera formal para periodos posteriores a una empresa desarrolladora de software. Se busca estimar el grado de aprendizaje con la productividad mostrada en periodos anteriores.

El análisis será llevado a cabo con la información de un contrato entre una entidad del gobierno Mexicano del sector energético y un proveedor de software. Por cuestiones de confidencialidad, el nombre de estas empresas no serán mencionados dentro de este documento.

Cada cierta cantidad de tiempo, aproximadamente dos años, esta entidad contrata proveedores de software para desarrollar los proyectos que requiere. El proveedor del servicio para el desarrollo de software actual fue contratado por dos años, 2018 y 2019, cuyo contrato fue extendido para el primer semestre del 2020.

3.1.2 Recolección de datos

Para determinar la productividad que se puede solicitar al proveedor de desarrollo de software para el primer semestre del 2020, la entidad brindó información de 21 proyectos desarrollados en los dos primeros años por el proveedor, con lo que se conformó el universo de datos para este caso de estudio. Esta información contiene, para cada proyecto desarrollado el año y semestre en que se desarrolló, el esfuerzo requerido y el tamaño funcional, el cual está medido en *CFP*[2](Tabla 3.2). Este tamaño funcional fue obtenido utilizando una estimación utilizando el método de Estimación de Proyectos en Contextos de Incertidumbre (*EPCU, Estimation of*

[2]CFP: COSMIC Function Points, la unidad de medida del método *COSMIC*.

Projects in Contexts of Uncertainty), como se define el la Guía de expertos para el dimensionamiento inicial del software con *COSMIC*[54]; por lo que los valores no son enteros.

3.1.3 Recolección de evidencias

Dado que la información de los proyectos se puede agrupar por el semestre en que fueron desarrollados, es adecuado utilizar el enfoque de la estimación por lotes de la Teoría Unitaria. Cada semestre contiene un lote de cierto número de unidades de tamaño funcional desarrolladas, y el esfuerzo para desarrollarlas, las cuales corresponden al conjunto de proyectos realizados en el semestre correspondiente.

Para modelar la productividad del proveedor, se utilizarán los conceptos definidos en la sección de Economía de la Ingeniería de Software (capítulo 1): el factor de productividad (*PDR*) y la productividad.

Por lo que, al utilizar la información de esta base de datos, la producción de software está dada en unidades de [*CFP*] y el esfuerzo requerido para producirlas está dado en horas de trabajo [WH]. Debido a esto, las unidades del *PDR* son [*WH/CFP*] y de la productividad son [*CFP/WH*].

Cabe mencionar que el enfoque para determinar el *PDR* en los dos primeros semestres no fue del todo correcto, ya que la entidad, debido a una mala asesoría, definió que todos los proyectos que fueron desarrollados en estos periodos tendrían un *PDR* fijo en 34[WH/CFP], el cual fue determinado mediante juicio de experto, es decir, no de manera formal y sin fundamentos derivados de un análisis de datos históricos como se recomienda en las buenas prácticas. Para los dos semestres del segundo año, 2019, se aplicó de manera correcta el método *COSMIC*, utilizando como referencia la base de datos de la Asociación Mexicana de Métricas de Software[11], por lo que los proyectos desarrollados en estos periodos tienen un *PDR* variable, derivado de la aceptación de las estimaciones y, en consecuencia, la validación del *PDR* fue desarrollada por la entidad en base a la definición de un proceso de validación de estimaciones, como el propuesto en [53].

TABLA 3.2: Información proporcionada sobre los proyectos desarrollados por la fábrica de software.

ID	Semestre	WH	PDR	CFP
1	1	667	34.00	19.61
2	1	660	34.00	19.41
3	1	583	34.00	17.14
4	1	574	34.00	16.88
5	2	509	34.00	14.97
6	2	381	34.00	11.20
7	2	363	34.00	10.67
8	2	257	34.00	7.55
9	3	19157	18.28	1047.97
10	3	14187	18.10	783.81
11	3	7324	16.28	449.87
12	3	6731	16.07	418.85
13	3	5772	15.99	360.97
14	3	2495	15.89	157.01
15	3	1389	15.78	88.02
16	4	12666	15.50	817.16
17	4	2275	15.37	148.01
18	4	17276	15.32	1127.67
19	4	27407	15.23	1799.54
20	4	2335	14.97	155.97
21	4	6880	13.36	514.97

3.1.4 Análisis de datos recolectados

Con los datos de la tabla 3.2, el análisis de la curva de aprendizaje se realizó utilizando la Teoría Unitaria para estimar el costo por lote, considerando a cada semestre de desarrollo como uno, por lo que es necesario calcular el Punto Medio de cada Lote (*LMP*) y el costo promedio de unidad por lote (*AUC*) como se menciona en el capítulo 1. La tabla 3.3 muestra los datos resultantes de estos cálculos.

TABLA 3.3: Resultado de los cálculos realizados para analizar el costo del lote con la Teoría Unitaria.

Lot	Unidades [CFP]	WH	PDR (AUC)	Primera Unidad	Unidades Acumuladas	LMP	ln(AUC)	ln(LMP)
1	73.05	2484	34.00	1.00	73.05	24.35	3.52	3.19
2	44.41	1510	34.00	74.05	117.47	94.51	3.52	4.54
3	3306.53	57055	17.25	118.47	3424.00	1204.07	2.84	7.09
4	4563.34	68839	15.08	3425.00	7987.34	5468.27	2.71	8.60

La figura 3.1 muestra la curva generada utilizando los datos de la tabla 3.3, en el eje Y va el *AUC*, costo promedio de cada unidad por lote, mientras que en el eje X está el *LMP*, punto medio de cada lote.

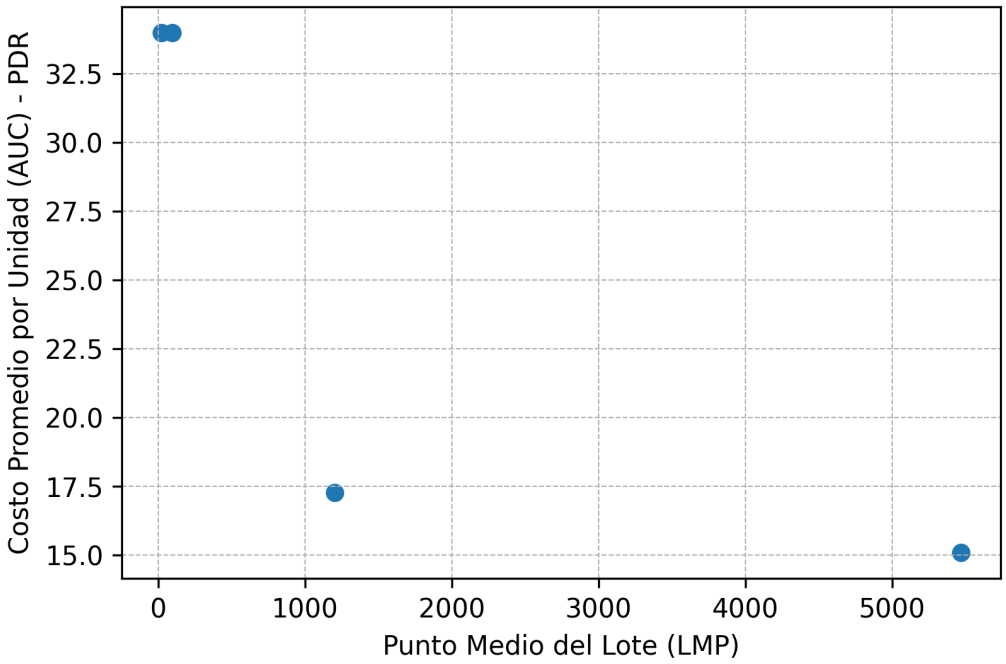

FIGURA 3.1: AUC(PDR) vs. LMP.

Análisis de curvas de aprendizaje utilizando la Teoría Unitaria

Se realiza una transformación sobre los valores de ambos ejes utilizando la función Logaritmo natural (ln) para hacer que los datos sean más lineales con el fin de realizar una regresión lineal [31]. Los valores obtenidos se muestran en la Tabla 3.3. Al graficar el $ln(AUC)$ para el eje Y y el $ln(LMP)$ en el eje X, se obtiene la gráfica mostrada en la figura 3.2. Es importante mencionar que la pendiente de la recta en la gráfica es negativa, lo que significa que hay aprendizaje.

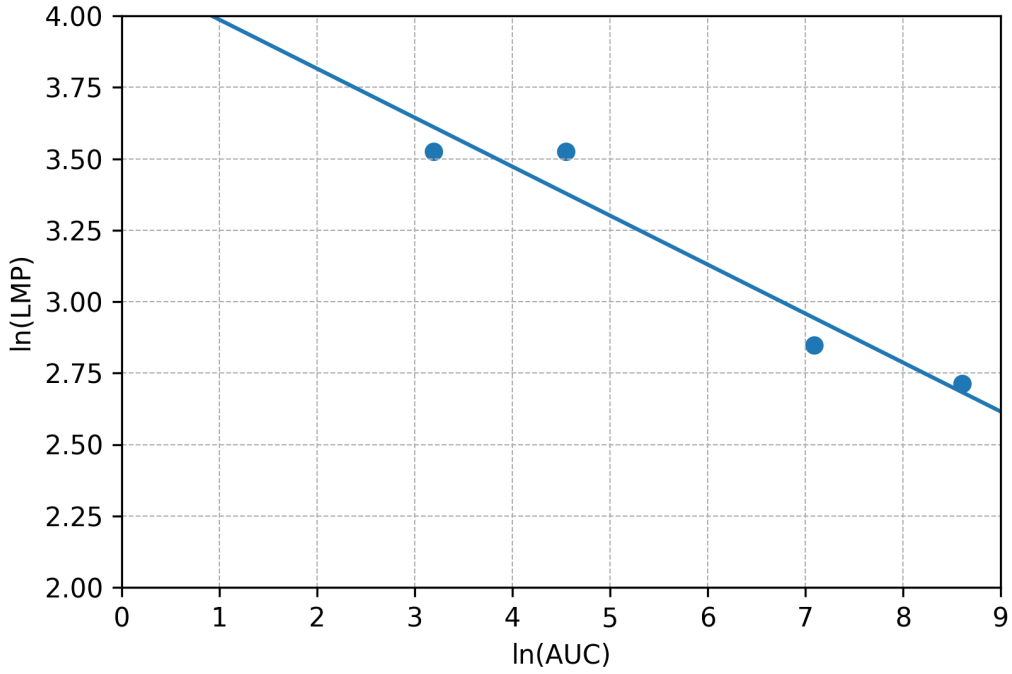

FIGURA 3.2: Regresión lineal, ln(PDR) vs. ln(LMP).

Los valores de las constantes que definen la ecuación de la recta mostrada en la figura 3.2 se pueden obtener realizando una regresión lineal utilizando los datos de la Tabla 3.3, lo que da como resultado la siguiente ecuación:

$$Y(x) = -0.171x + 4.1581 \tag{3.1}$$

Donde:

- Y es el *AUC*
- x es el *LMP*
- $A = 4.1581$ que es el costo teórico de la primer unidad
- $b = -0.171$ que la pendiente de la curva de aprendizaje

Sin embargo, estos resultados se encuentran en unidades de Logaritmo natural (ln), por lo que la ecuación que realmente se tiene es:

$$ln(PDR) = -0.171 * ln(LMP) + 4.1581 \tag{3.2}$$

Para transformar la ecuación 3.2 a su forma original, es necesario aplicar la función exponencial a cada lado de esta ecuación; que como resultado se obtiene la siguiente ecuación:

$$PDR = 63.951 * LMP^{-0.171} \tag{3.3}$$

En esta ecuación se puede observar que la pendiente de la curva de aprendizaje es:

$$2^b = 2^{-0.171} = 0.887 = 88.7\% \tag{3.4}$$

Esta es la ecuación de mejor modela el ambiente de producción para los datos de la tabla 3.2. Esto muestra que existe una curva de aprendizaje de 88.7%; lo que significa que hay un incremento de la productividad de 11.3% cada que se duplica la cantidad de *CFP* desarrollados.

Ahora es posible resolver la ecuación para conocer el costo promedio por unidad para cualquier lote posterior, calculando su *LMP*, el cual puede ser obtenido una vez que se conoce su primera y última unidad.

Estimación del costo de un lote para el siguiente periodo

Como ejercicio de validación, se realiza un estimado del tamaño funcional a desarrollar para el siguiente periodo, tomando el tamaño promedio de los proyectos desarrollados en los cuatro semestres previos, el cual es 1997[*CFP*].

A partir de esta información, los valores para el lote 5 (primer semestre del año 2020) son calculados, los cuales son la primera y última unidad, el punto medio del lote (*LMP*), y se utiliza la ecuación de predicción 3.3 para calcular el *PDR* del lote. Finalmente, ya que el el *PDR* = [WH/CFP], la cantidad de [WH] se puede obtener multiplicando el *PDR* por el tamaño funcional a desarrollar.

La tabla 3.4 muestra en la columna 4 el *PDR* para cada lote de los 4 periodos anteriores, mientras que en la columna 5 se muestra el valor estimado para cada lote utilizando la ecuación 3.3. En la última línea se agrega el lote 5, el cual corresponde al primer semestre del año 2020, donde en su primer columna está el tamaño funcional que se estimó anteriormente, 1997 [*CFP*], para el cual también se estima el *PDR* utilizando la ecuación 3.3. En la última columna (6) se calcula la Magnitud del Error Relativo (*MRE*) considerando el valor real y el estimado para los cuatro primeros lotes, columnas 4 y 5 respectivamente.

TABLA 3.4: Resultados de los cálculos realizados para analizar el costo por lotes utilizando la Teoría Unitaria.

Lote	Unidades	WH	PDR real	PDR Estimado	MRE
1	73.05	2484.00	34.00	36.99	0.088
2	44.41	1510.00	34.00	29.32	0.137
3	3306.53	57055.00	17.25	18.95	0.098
4	4563.34	68839.00	15.08	14.62	0.030
5	1997.00	-	-	13.44	-

Criterios de calidad

Utilizando la información de la columna 5 de la tabla 3.4, se evaluaron los criterios de calidad para analizar la robustez del modelo, los cuales son la media de la magnitud del error relativo *Mean Magnitude of Relative Error, MMRE*, desviación estándar del *MRE* (*MRE Standard Deviation, RMS*) y el nivel de predicción al 25%, cuyos resultados se muestran en la tabla 3.5.

TABLA 3.5: Criterios de calidad, MMRE, RMS, and Pred (25%).

Criterio de calidad	Valor
MMRE	0.088
RMS	2.913
Pred (25%)	100%

Basado en la información de la tabla 3.5, se puede mencionar que hay un promedio del error relativo (*MMRE*) de 8.8%, con una raíz cuadrada de la media del error cuadrático (*RMS*) de 2.913, y todos los puntos se encuentran dentro del 25% de nivel de predicción. Observando que el valor de *MRE* y considerando el nivel de predicción, todos los puntos se encuentran dentro de un nivel de predicción de 13.7%. Esto es, todas las estimaciones presentan un error relativo menor o igual a 13.7%; de esta forma, se espera que el valor de *PDR* estimado para el periodo 5 sea de $13.44[WH/CFP] \pm 13.7\%$

Después de graficar los datos de la tabla 3.4, se puede observar en la figura 3.3 que los valores estimados (verde) para los periodos conocidos muestran un comportamiento similar a los valores reales (azul), con un *MMRE* de 8.8% (tabla 3.5).

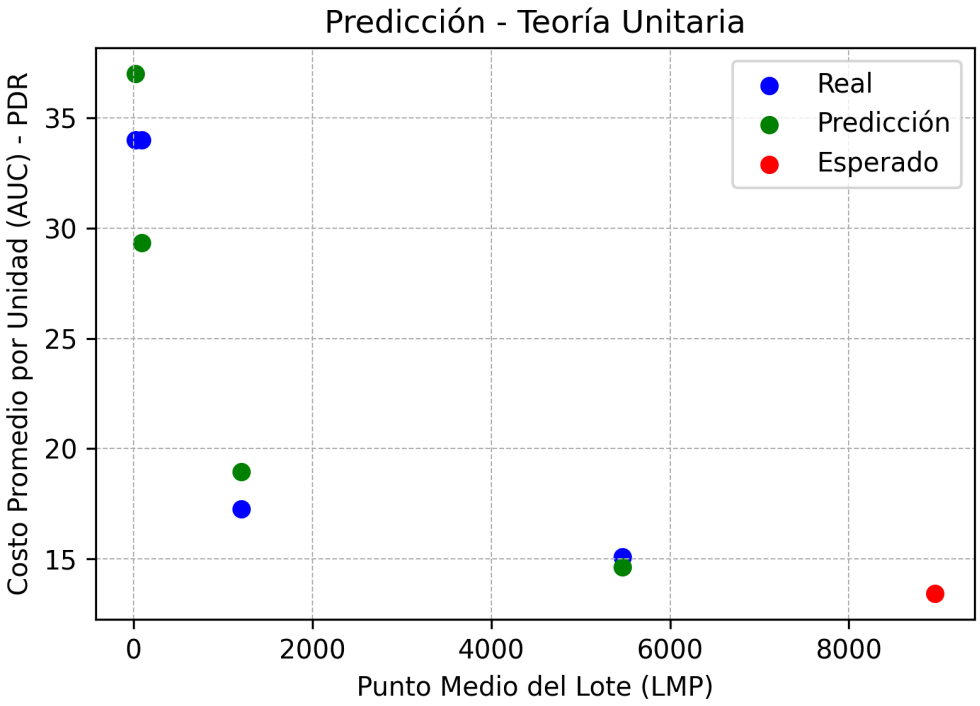

FIGURA 3.3: En azul, los valores originales y en verde los estimados por la ecuación 3.11, y en rojo el *PDR* estimado para el primer semestre del año 2020.

De esta forma, tenemos un modelo que de forma confiable representa la curva de aprendizaje definida por la ecuación 3.3, la cual tiene una proporción de aprendizaje de 88.7%. Lo que que es posible y alcanzable por parte del proveedor, ya que fue obtenida de manera formal realizando un análisis de curvas de aprendizaje utilizando la productividad observada en periodos previos de desarrollo.

3.1.5 Reporte de resultados

Un problema al que se enfrentan las empresas que cuentan con un mismo proveedor por varios periodos de tiempo, es conocer de manera formal qué proporción de mejora se podría esperar debido al *know-how* adquirido durante el servicio, esta situación se visualiza como un beneficio extra o motivador para el cliente en contratos a mediano o largo plazo.

Este caso de estudio presenta una propuesta para determinar el grado de aprendizaje de un proveedor para solicitar una cantidad de mejora del factor de productividad (*PDR*) utilizando la información de periodos anteriores a través de un estudio de caso real en la industria mexicana. El enfoque utilizado para determinar el grado de aprendizaje es el análisis de curvas de aprendizaje de la Teoría Unitaria.

El análisis de las curvas de aprendizaje para este caso de estudio ha mostrado que hay aprendizaje a lo largo de cada período en el que se desarrollaron diferentes proyectos. Esto muestra que la productividad mejora con el tiempo, exhibiendo una tasa de aprendizaje del 88.7%, lo que representa un aumento de productividad del 11.3% cada que la cantidad de unidades *CFP* desarrolladas se duplica.

Con estos resultados, podemos estimar el esfuerzo en [WH] para producir una unidad [*CFP*], es decir, el *PDR* para el siguiente lote, una vez que tengamos el tamaño estimado que se producirá de tamaño funcional de software para el quinto periodo.

Para estimar el tamaño funcional esperado a desarrollar en el lote del primer semestre del año 2020, se consideró el promedio del tamaño funcional de los lotes referidos a los años 2018 y 2019. El tamaño funcional esperado para el primer semestre es 1997 [*CFP*], usando la ecuación 3.3, muestra que se puede esperar un *PDR* de $13.44[WH/CFP] \pm 13.7\%$.

Este caso de estudio presenta una cantidad pequeña de datos para analizar, por lo que un trabajo futuro es buscar conjuntos de datos más grandes de desarrollo de proyectos de software para repetir el análisis y comparar resultados.

3.2 Segunda iteración: comparación de los resultados de dos teorías de curvas de aprendizaje

Derivado de los resultados de éste primer caso de estudio, se planteó la siguiente pregunta de investigación: ¿cuál teoría de las definidas para las curvas de aprendizaje presenta mejores resultados?, pensando que se podía obtener un modelo más robusto para la predicción de el *PDR* esperado por parte de la empresa proveedora de desarrollo de software al realizar una comparación de los modelos obtenidos por la Teoría Unitaria (desarrollada en el primer caso de estudio) y la Teoría del Promedio Acumulado definidas por las curvas de aprendizaje.

Debido a que este segundo caso de estudio surge para ampliar el primero, solo se describirán las secciones agregadas y los cambios que hubo en cuanto a objetivos, análisis cuantitativo (la Teoría del Promedio Acumulado y la comparación de ambas teorías) y conclusiones, ya que la base de datos utilizada es la misma, las fases: 2. Recolección de datos y 3. Recolección de evidencias son las mismas que las del primero.

En la tabla 3.6 se muestran los puntos más importantes, y de manera resumida, de cada fase de este caso de estudio, donde los cambios se encuentran principalmente en las preguntas que se busca resolver, el análisis cuantitativo, para incluir la Teoría del Promedio Acumulado, y el reporte de resultados, donde se agrega la comparación de ambas teorías.

TABLA 3.6: Metodología para llevar a cabo el caso de estudio.

1	Diseño de caso de estudio	**Preguntas de casos del estudio:** 1. ¿Cuál será la mejora en términos de *PDR* para el siguiente bloque de proyectos, dado el aprendizaje que ha tenido el proveedor? 2. ¿Cuál teoría de las definidas para las curvas de aprendizaje presenta mejores resultados? **Objeto de estudio:** • 21 proyectos desarrollados a lo largo de 4 semestres
2	Recolección de datos	**Datos de interés:** • tamaño funcional del software desarrollado por semestre • esfuerzo requerido para desarrollar la cantidad de *CFP* por semestre
3	Recolección de evidencias	**Métricas de la evidencia:** • *CFP*, *PDR*, productividad • criterios de calidad: MRE, MMRE, RMS, Pred(25%)
4	Análisis de los datos recolectados	**Análisis cuantitativo:** • cálculo del punto medio del lote (*LMP*), costo promedio por unidad (*AUC*), pendiente de la curva de aprendizaje para la Teoría Unitaria • cálculo de la cantidad acumulada, costo acumulado, costo promedio acumulado, pendiente de la curva de aprendizaje para la Teoría del Promedio Acumulado
5	Reporte de resultados	**Reporte del caso de estudio:** • metodología, criterios de calidad, resultados de ambas teorías, comparación

3.2.1 Diseño de caso de estudio

Este caso de estudio presenta una comparación entre dos de los modelos definidos por la teoría de la curva de aprendizaje. El objetivo es determinar con cuál teoría se obtendría un mejor resultado.

3.2.2 Análisis de datos recolectados

Con los datos de la tabla 3.2, el análisis de la curva de aprendizaje se realizará utilizando la Teoría del Promedio Acumulado para estimar el costo por lote, considerando a cada semestre de desarrollo como uno, por lo que es necesario calcular la *Cantidad Acumulada*, *Costo Acumulado*

y *Costo Promedio Acumulado* como se describe en el capítulo 1. La tabla 3.7 muestra los datos resultantes de estos cálculos.

TABLA 3.7: Resultado de los cálculos realizados para analizar el costo del lote con la Teoría del Promedio Acumulado.

Lot	Unidades [CFP]	WH	PDR (CAC)	Primera Unidad	Unidades Acumuladas	WH Acumuladas	ln (CAC)	ln (CmQty)
1	73.05	2484	34.00	1.00	73.05	2484.00	3.52	4.29
2	44.41	1510	34.00	74.05	117.47	3994.00	3.52	4.76
3	3306.53	57055	17.82	118.47	3424.00	61049.00	2.88	8.13
4	4563.34	68839	16.26	3425.00	7987.34	129888.00	2.78	8.98

La figura 3.4 muestra la curva generada utilizando los datos de la tabla 3.7, en el eje Y el Costo Promedio Acumulado (*CAC*) de cada unidad por lote, mientras que en el eje X, la cantidad acumulada de unidades producidas.

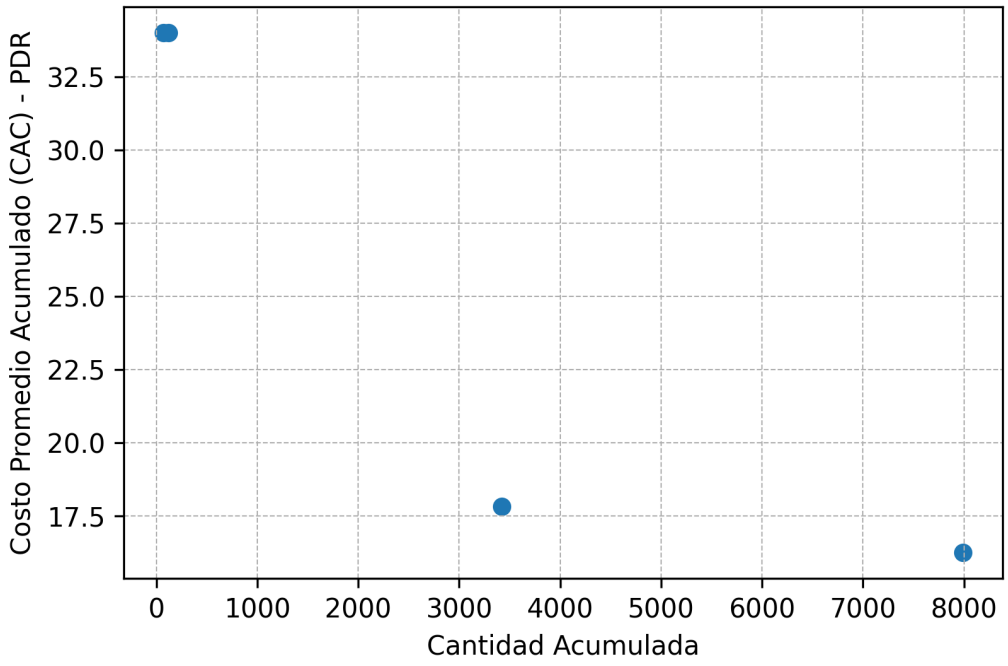

FIGURA 3.4: Cantidad acumulada vs. Costo Promedio Acumulado.

Análisis de curvas de aprendizaje utilizando la Teoría del Promedio Acumulado

Se realiza una transformación sobre los valores de ambos ejes utilizando la función Logaritmo natural (ln) para hacer que los datos sean más lineales con el fin de realizar una regresión lineal

[31]. Los valores obtenidos se muestran en la tabla 3.7. Al graficar $ln(CAC)$ para el eje Y y el $ln(CumQuantity)$ para el eje X, se obtiene la gráfica mostrada en la figura 3.5. Es importante mencionar que la pendiente de la recta en la gráfica es negativa, lo que significa que hay aprendizaje en la producción.

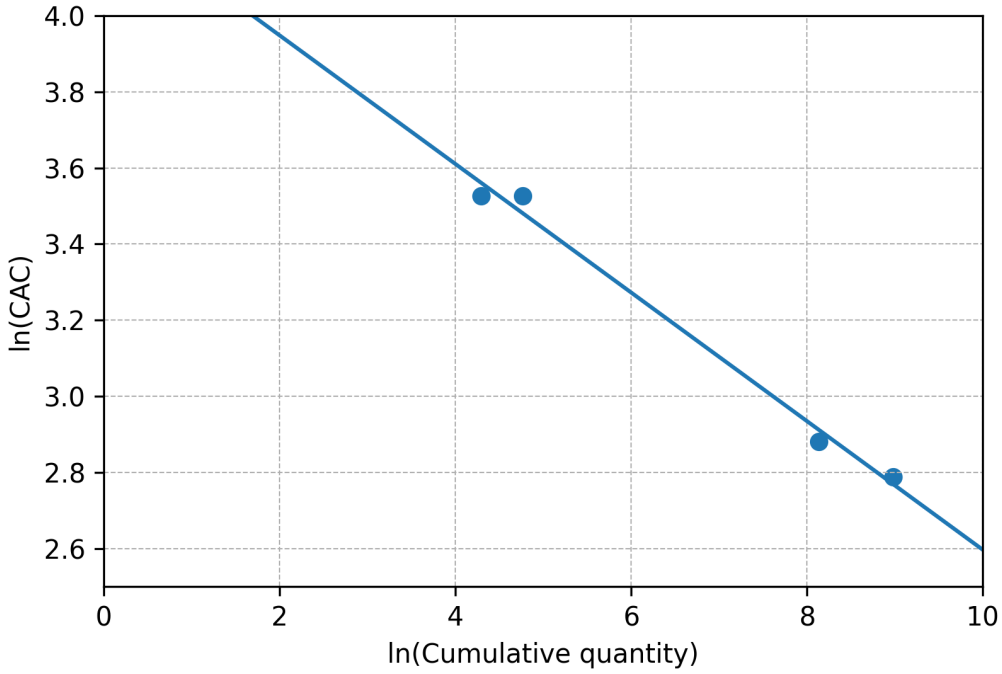

FIGURA 3.5: $ln(cantidadAcumulada)$ vs. $ln(CAC)$, utilizando los datos trasformados de la tabla 3.7.

Los valores de las constantes que definen la ecuación de la recta mostrada en la figura 3.5 se pueden obtener realizando una regresión lineal usando los datos de la Tabla 3.7, lo que da como resultado la siguiente ecuación:

$$Y(x) = -0.169x + 4.2864 \tag{3.5}$$

Donde:

- Y es el Costo Acumulado Promedio (*CAC*)
- x es la cantidad producida acumulada
- $A = 4.2864$ es el costo teórico del primer unidad
- $b = -0.169$ es la pendiente de la curva de aprendizaje

Sin embargo, estos resultados se encuentran en unidades de logaritmo natural (ln), por lo que la ecuación que realmente se tiene es:

$$ln(CAC) = -0.169 * ln(CumQuantity) + 4.2864 \tag{3.6}$$

Para transformar la ecuación 3.6 a su forma original, es necesario aplicar la función exponencial a cada lado de la ecuación; como resultado se obtiene la siguiente ecuación:

$$CAC = 72.71 * N^{-0.169} \tag{3.7}$$

En esta ecuación se puede observar que la pendiente de la curva de aprendizaje es:

$$2^b = 2^{-0.169} = 0.8895 = 88.95\% \tag{3.8}$$

Para conocer el costo total de x unidades, se utiliza la ecuación 1.23, definida en el capítulo 1, por lo que al sustituir las variables tenemos el siguiente resultado:

$$CT(x) = A * x^{b+1} = 72.708 * x^{-0.169+1} = 72.708 * N^{0.831} \tag{3.9}$$

Sin embargo, de la misma forma que en la Teoría Unitaria, para obtener el costo de una sola unidad x, se utiliza la ecuación 1.24, definida en el capítulo 1, por lo que al sustituir las variables tenemos el siguiente resultado:

$$CostUnit(x) = CT(x)' = (b+1) * A * N^b = (0.831) * 72.708 * N^{-0.169} \tag{3.10}$$

Finalmente, para calcular el costo total de un lote específico, se requiere de conocer la primera (f) y última (l) unidad de éste, para utilizar ecuación 1.25:

$$\begin{aligned} CT(f,l) &= CT(l) - CT(f-1) \\ &= A * [l^{b+1} - (f-1)^{b+1}] \\ &= 72.708 * [l^{0.831} - (f-1)^{0.831}] \end{aligned} \tag{3.11}$$

Esta es la ecuación que mejor modela el ambiente de producción para los datos de tabla 3.2. Esto muestra que existe una curva de aprendizaje de 88.95%; lo que significa que hay un incremento de la productividad de 11.05% cada que se duplica la cantidad acumulada de *CFP* desarrollada.

Estimación del costo de un lote para el siguiente periodo

Como ejercicio de validación, se utilizará el mismo tamaño funcional estimado a desarrollar para el siguiente periodo que el utilizado en el primer caso de estudio, que fue 1997[*CFP*].

A partir de esta información, se calculan los valores para el lote 5 (primer semestre del año 2020), los cuales son las unidades acumuladas, el costo acumulado y la ecuación 3.11 se utiliza para calcular el esfuerzo ([*WH*]) requerido para desarrollar el tamaño funcional del quinto lote. Por último, dado que el *PDR acumulado = [WH acumulado/ CFP acumulado]*, es necesario calcular el costo acumulado sumando el costo del quinto lote al costo acumulado hasta el cuarto lote, es decir, agregar 1997 [*CFP*] al costo acumulado anterior.

La tabla 3.8 muestra en la columna 4 el *PDR* acumulado promedio para cada lote de los períodos anteriores, mientras que la columna 5 muestra el *PDR* estimado para ese lote usando la ecuación 3.11. En la última línea se incluye el lote 5, el cual corresponde al primer semestre del año 2020,donde en su primer columna se encuentra el promedio de [*CFP*] desarrollado de los períodos anteriores se consideró como el último tamaño de lote (1997 [*CFP*]). En la última columna (6) se calcula la Magnitud del Error Relativo (*MRE*) el *PDR* real (columna 4) y el *PDR* estimado (columna 5).

TABLA 3.8: Resultados de los cálculos realizados para analizar el costo por lotes utilizando la Teoría del Promedio Acumulado.

Lote	Unidades	WH Acumuladas	PDR real (CAC)	PDR Estimado	MRE
1	73.05	2484.00	34.00	35.21	0.035
2	117.47	3994.00	34.00	32.49	0.044
3	3424.01	61049.00	17.82	18.38	0.031
4	7987.34	129888.00	16.26	15.93	0.020
5	9985.34	-	-	15.34	-

Criterios de calidad

De igual manera que en el primer caso de estudio, se utilizó la información de la columna 5 de la tabla 3.8 para evaluar los criterios de calidad y así poder analizar la robustez del modelo, los cuales se muestran en la tabla 3.9.

TABLA 3.9: Criterios de calidad, MMRE, RMS, and Pred (25%).

Criterio de calidad	Valor
MMRE	0.032
RMS	1.017
Pred (25%)	100%

Basado en la información de la tabla 3.9, se puede mencionar que hay un promedio del error relativo (*MMRE*) de 3.2%, con una raíz cuadrada de la media del error cuadrático (*RMS*) de 1.017, y todos los puntos se encuentran dentro del 25% de nivel de predicción. Observando el valor de *MRE* y considerando el nivel de predicción, todos los puntos se encuentran dentro de un nivel de predicción de 4.4%. Esto es, todas las estimaciones presentan un error relativo menor o igual a 4.4%; de esta forma, se espera que el valor de *PDR* estimado para el periodo 5 sea de $15.34[WH/CFP] \pm 4.4\%$

Después de graficar los datos de la tabla 3.8, se puede observar en la figura 3.6 que los valores estimados (amarillo) para los periodos conocidos muestran un comportamiento similar a los valores reales (cyan), con un *MMRE* de 3.2% (table 3.9).

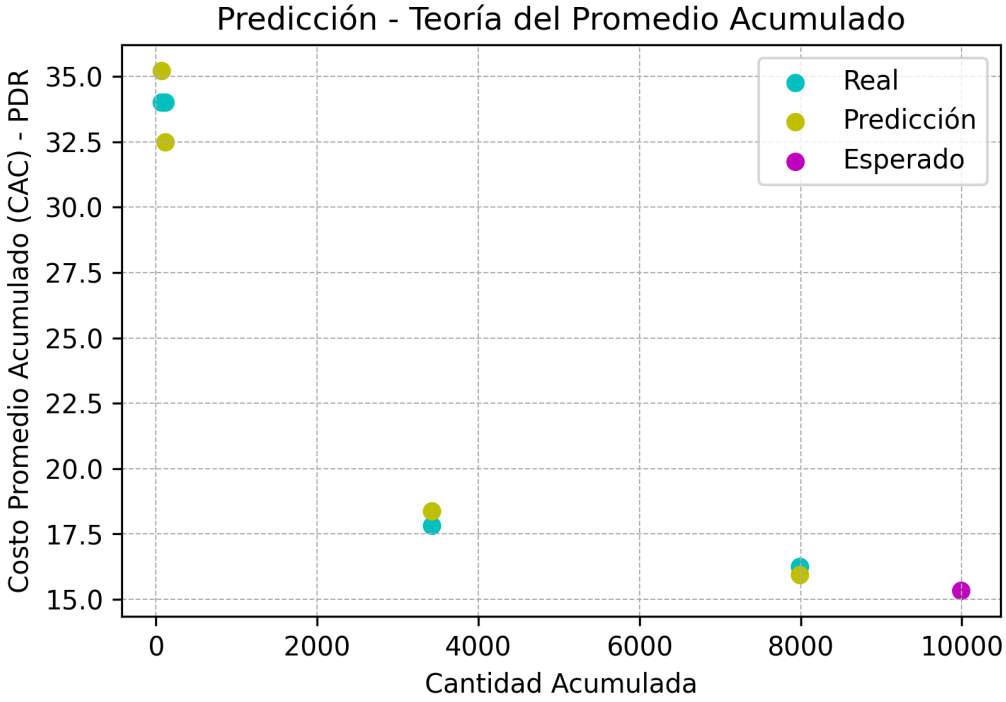

FIGURA 3.6: En cyan, los valores originales y en amarillo los estimados por la ecuación 3.11, y en magenta el *PDR* estimado para el primer semestre del año 2020.

De esta forma, tenemos un modelo con mayor precisión, cuya curva de aprendizaje esta definida por la ecuación 3.7, la cual tiene una proporción de aprendizaje de 88.95%. Cantidad que es posible y alcanzable por parte del proveedor, ya que fue obtenida de manera formal realizando un análisis de curvas de aprendizaje utilizando la productividad observada en periodos previos de desarrollo.

3.2.3 Comparación entre ambas teorías, UT y CAT

En la tabla 3.10 se puede observar que aunque ambas teorías exhiben una precisión del 100% a un nivel de predicción del 25%, la Teoría del Promedio Acumulado tiene un error relativo promedio más bajo que la Teoría Unitaria, además de ser más conservadora. Por otra parte, la pendiente de la curva de aprendizaje para ambas teorías es similar, 88.70% para la Teoría Unitaria y 88.95% para la Teoría del Promedio Acumulado.

TABLA 3.10: Comparación de resultados y criterios de calidad de ambas teorías.
* Cada que la cantidad de unidades o unidades acumuladas producidas se duplica.

	UT	CAT
MMRE	0.088	0.032
RMS	2.913	1.017
Pred (25%)	100.00 %	100.00 %
Costo teórico del primer unidad	63.951	72.710
Pendiente de la curva de aprendizaje	88.70%	88.95%
Incremento de la productividad *	11.30%	11.05%
PDR estimado	13.44	15.34

En la figura 3.7 se muestra una comparación de las dos teorías, donde se puede observar que a partir de los datos analizados, el *PDR* disminuye en ambas, aproximadamente de manera similar, pero la Teoría Unitaria siempre por debajo.

Dado que la curva de promedio acumulado se basa en el costo promedio de la cantidad de producción en lugar del costo de una unidad o lote en particular, muestra menos cambios a las tendencias de costos que la curva de la Teoría Unitaria. Sería necesario un cambio considerable en el costo de alguna unidad o lote de unidades antes de que se refleje un cambio en la curva de la Teoría del Promedio Acumulado.

Para mostrar las curvas de aprendizaje de ambas teorías en una misma gráfica es necesario considerar las diferencias que los ejes cada una representan:

- Teoría Unitaria (*UT*)
  - X es el punto medio del lote (*LMP*)
  - Y es el costo promedio de cada unidad dentro del lote (*PDR*)
- Teoría del Promedio Acumulado (*CAT*)
  - X es la cantidad acumulada producida
  - Y Es el Costo Acumulado Promedio (*PDR*)

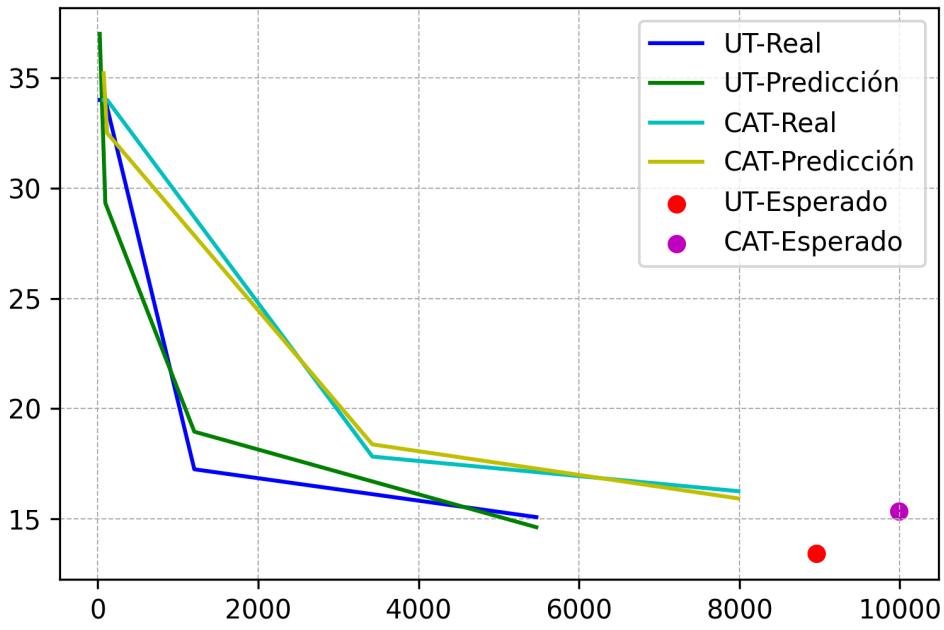

FIGURA 3.7: Comparación de las curvas de aprendizaje de ambas teorías, Teoría Unitaria (figura 3.3) y Teoría del Promedio Acumulado (figura 3.6).

3.2.4 Reporte de resultados

Es importante tener la información necesaria para elegir de manera correcta cuál de las teorías de las curvas de aprendizaje ofrece un modelo con mejores resultados de acuerdo a su ambiente de producción.

Este caso de estudio presenta una comparación entre dos enfoques de las curvas de aprendizaje para determinar el grado de aprendizaje que obtiene un proveedor para así solicitarle un aumento de eficiencia de los equipos en la construcción del software (*PDR*) con base en la productividad mostrada en periodos anteriores.

El análisis de las Curvas de Aprendizaje para este caso de estudio ha mostrado que existe aprendizaje a lo largo de cada período en el que se desarrollaron diferentes proyectos. Lo cual muestra que la productividad aumenta con el tiempo, mostrando un aprendizaje del 88.7% y aumento de la productividad del 11.3% utilizando la Teoría de la Unitaria, mientras que al usar la Teoría del Promedio Acumulado se obtiene un grado de aprendizaje del 88.95%, lo que representa un aumento de la productividad de 11.05%.

Con estos resultados, podemos estimar el esfuerzo requerido en [WH] para producir una unidad de tamaño funcional [*CFP*], es decir, el *PDR* para el siguiente lote, una vez que se tenga el tamaño

estimado a producir.

A manera de ejercicio, se estimó el tamaño funcional esperado a desarrollar en el lote del primer semestre del año 2020, considerando el promedio del tamaño funcional de los lotes anteriores, 2018 y 2019, dando como resultado 1997 [*CFP*], con este valor podemos utiliza la ecuación (3.3) correspondiente a la Teoría Unitaria, para obtener el *PDR* esperado, que es de $13.44[WH/CFP] \pm 13.7\%$, y con la Teoría del Promedio Acumulado, utilizando la ecuación 3.11 podemos esperar un *PDR* de $15.34[WH/CFP] \pm 4.4\%$.

Dado que la curva de costo de la Teoría Unitaria siempre esta por debajo de la curva de costo de la Teoría de Promedio Acumulado, los negociadores generalmente prefieren utilizar las curvas de costo de la Teoría Unitaria, ya que da una estimación más restrictiva y responde mejor a las variaciones de costos unitarios[31].

Qué tipo de curva de aprendizaje utilizar es una decisión importante durante estimación de costos ya que hay otros factores a considerar al momento de seleccionar la curva de aprendizaje para realizar el análisis. La Teoría de Promedio Acumulado se utiliza cuando existe incertidumbre debido a que los costos unitarios no se conocen con precisión y se sabe que habrá cambios en el producto. Por otra parte, la Teoría Unitaria muestra mejores resultados cuando se tiene un mayor control del proceso de desarrollo, no hay ambigüedades y se tiene la información del costo para cada unidad. Por lo que en ambientes de desarrollo de software es recomendable utilizar la Teoría del Promedio Acumulado mientras no se tenga un sistema reproducible de métricas que permita tener certeza de los costos unitarios.

Dado que los proyectos de software muestran diferencias con respecto a proyectos de otras áreas[14], donde una de ellas es la gran cantidad de cambios que tienen durante su desarrollo, así como un menor índice de éxito[19], además de que la información acerca de los requerimientos funcionales es ambigua al inicio y se conoce a detalle en etapas avanzadas del desarrollo.

Por lo tanto en este caso de estudio se considera la Teoría del Promedio Acumulado, ya que presenta un modelo que arroja mejores resultados de criterios de calidad que el modelo de la Teoría Unitaria para este caso de estudio. Un ejemplo de esta situación es que en este caso de estudio se tuvo que utilizar una técnica de aproximación para medir el tamaño funcional de los proyectos desarrollados en los dos primeros semestres. El uso de técnicas de aproximación es cuando no hay suficiente información o tiempo para llevar a cabo una medición de manera formal, lo que implica que hay incertidumbre.

Este caso de estudio presenta una cantidad pequeña de datos para analizar, por lo que un trabajo futuro es buscar conjuntos de datos más grandes de desarrollo de proyectos de software para repetir el análisis y comparar resultados.

Conclusiones

El objetivo de esta tesis fue determinar el grado de aprendizaje de una empresa desarrolladora de software considerando los periodos previos en los que ha trabajado bajo un mismo o similar contexto de una manera formal y fundamentada para no comprometer el éxito de sus proyectos utilizando la herramienta de curvas de aprendizaje, y así poder estimar el factor de productividad (*PDR*) esperado para un siguiente periodo.

Para lograr ésto, se llevaron a cabo dos casos de estudio, utilizando la información que proporcionó una empresa del sector energético mexicano, la cual consiste de 21 proyectos desarrollados en los años 2018 y 2019, y contiene la información del tamaño funcional, medido en *CFP*, y el esfuerzo requerido para desarrollarlos ([*WH*]).

En el primer caso de estudio se utilizó la Teoría Unitaria para estimar el grado de aprendizaje con esta información, donde se mostró que efectivamente existe aprendizaje a lo largo de cada período en el que se desarrollaron los diferentes proyectos. Por lo que se puede observar que la productividad mejora con el tiempo, exhibiendo una tasa de aprendizaje del 88.7%, lo que representa un aumento de productividad del 11.3% cada que la cantidad de unidades *CFP* desarrolladas se duplica.

Además, realizando el ejercicio de utilizar un valor estimado de producción para el primer semestre del año 2020 de 1997[*CFP*], utilizando las ecuaciones para realizar la estimación del *PDR* para este lote, se obtuvo que se espera que la empresa desarrolladora de software presente un valor de $13.44[WH/CFP] \pm 13.7\%$.

Para el segundo caso de estudio, cuyo objetivo era comparar las dos teorías definidas en las curvas de aprendizaje, se llevó a cabo el análisis de esta misma información utilizando la Teoría del Promedio Acumulado con el fin de comparar los resultados y criterios de calidad.

Por lo que con la Teoría del Promedio Acumulado se muestra una tasa de aprendizaje del 88.95%, lo que representa un aumento de la productividad de 11.05% cada que el número de unidades *CFP* desarrolladas acumuladas se duplica.

Al igual que en el primer caso de estudio, se utiliza un valor de 1997[*CFP*] estimado a desarrollar en el siguiente periodo, y al utilizar las ecuaciones de predicción de esta teoría para el nuevo lote, se espera que la empresa desarrolladora de software presente un *PDR* de $15.34[WH/CFP] \pm 4.4\%$.

De acuerdo a los criterios de calidad de ambas teorías (tabla 3.5), se puede observar que la Teoría del Promedio Acumulado presenta mejores resultados, ya que ésta teoría se adapta mejor a los ambientes de desarrollo de software, debido a que en éstos, hay una gran cantidad de cambios que tienen durante su desarrollo[19], además de que la información acerca de los requerimientos funcionales es ambigua al inicio y se conoce a detalle en etapas avanzadas del desarrollo.

Para facilitar el realizar los cálculos del análisis se desarrolló una herramienta en el lenguaje de programación *Python* (Apéndice A), que puede ser utilizada de manera general para cualquier conjunto de datos. Cabe mencionar que, como las curvas de aprendizaje se aplican en cualquier sector donde haya procesos de producción, tiene un amplio dominio de aplicación.

Para las organizaciones, poder estimar el valor del factor de productividad de un equipo de desarrollo de software que trabaja en un mismo contexto por varios periodos de tiempo, de manera formal y bien fundamentada, ayudará a tomar decisiones de negocio más precisas (utilizando información de costos de producción por unidad y tiempos de entrega para periodos de desarrollo posteriores) sin comprometer el éxito del proyecto.

De esta manera, se logró cumplir el objetivo de esta tesis, exhibiendo la existencia de aprendizaje en el desarrollo de software a través de dos casos de estudio, además de presentar dos modelos de estimación del *PDR* para periodos posteriores utilizando la información de proyectos realizados anteriormente. Adicionalmente a ésto, se compararon los resultados de ambas teorías para mostrar cuál es la que mejor modela la producción de software.

Además se logró desarrollar una herramienta para realizar el análisis de curvas de aprendizaje con las dos teorías en el enfoque por lotes, lo que facilita su aplicación, con esto se incentiva el uso de métricas estandarizadas, como lo es *COSMIC*, para poder realizar el análisis formal en los proyectos de software.

Limitaciones

Para analizar la productividad de una empresa de desarrollo de software, se deben incluir otras variables, como la rotación de personal y las interrupciones en la producción [31], donde el factor de mejora se ve afectado negativamente.

Estos casos de estudio fueron llevados a cabo utilizando una cantidad pequeña de datos para analizar, por lo que los resultados con diferentes bases de datos pueden llegar a tener variaciones.

También, es necesario notar que la medición del tamaño de software debe ser realizada de manera correcta y utilizando una métrica estandarizada para poder obtener resultados más precisos.

Trabajo futuro

Derivado de las limitaciones, un trabajo futuro es buscar conjuntos de datos más grandes de desarrollo de proyectos de software para repetir el análisis (lo cual se puede realizar de manera sencilla utilizando la herramienta implementada en esta tesis) y comparar resultados.

Además del tamaño de las bases de datos para realizar el análisis, buscar obtener mayor cantidad de datos, como rotación de personal e interrupción de la producción, lo cual afecta de manera negativa el aprendizaje (pérdida de aprendizaje), y agregarlo al análisis de curvas de aprendizaje para obtener modelos de predicción más precisos.

Por otra parte, se puede incluir la variable de la metodología utilizada para desarrollar el software y realizar comparaciones de mejora del factor de productividad conforme se trabaja en un mismo dominio o similar contexto.

E.N.D.

Apéndice A

Código fuente del programa para el análisis de curvas de aprendizaje

```
1  ##############################################################################################
2  # Analysis of Learning Curves on software development with CFP
3  # Daniel Torres Robledo
4  ##############################################################################################
5  import matplotlib.pyplot as plt
6  import pandas as pd
7  import math
8
9  from scipy import stats
10 from statistics import stdev
11
12 pd.options.display.max_columns = 100#None
13
14
15 ##############################################################################################
16 #Read and prepare the initial data
17 def read_data(data_name):
18     data = pd.read_csv(data_name)
19     data["nombre"] = "p"+data["periodo"].map(str) +"-p"+  data["proyecto"].map(str)
20     data["cfp"] = data["esfuerzo"]/data["pdr"]
21     #Remove first period
22     data = data[data.periodo>1]
23
24     plt.scatter(x=data["cfp"], y=data["esfuerzo"])
25     plt.xlabel("CFP")
26     plt.ylabel("Effort")
27     plt.title("Effort vs CFP")
28     plt.savefig('effort.png', dpi=300)
29     plt.show()
30
31     #how long it takes to develop a cfp, the PDR is that measure how many HH per CFP
32     plt.scatter(x=data["nombre"], y=data["pdr"])
```

```
33      plt.xlabel("Project")
34      plt.ylabel("PDR")
35      plt.title("Product Delivery Rate")
36      plt.savefig('pdr.png', dpi=300)
37      plt.show()
38
39      return data
40
41
42  #Calculate the Lot Mid Point
43  def lmp(first, last):
44      if first==1:
45          size = last-first+1
46          if size<10: return(size/2)
47          else: return size/3
48      else:
49          return (first+last+(2*math.sqrt(first*last)))/4
50
51  #Cost prediction with the equation from UT - lot cost
52  def pred(a, b, n):
53      return a*(n**b)
54
55  #Cost prediction with the equation from CAT - lot cost
56  def pred_cat(a, b, f, l):
57      return a*(l**(b+1)-(f-1)**(b+1))
58
59  #Calculate the errors from a dataframe with originals and predicted values
60  def get_errors(data, model, var_original, var_pred):
61      data["diff"] = data[var_original] - data[var_pred]
62      data["mre"] = abs((data[var_original] - data[var_pred])/data[var_original])
63      data["diff2"] = (data[var_original] - data[var_pred])**2
64      data["diffAbs"] = abs(data["diff"])
65      print(data)
66
67      errs = {"MMRE"       : sum(data["mre"])/len(data),
68              "RMSE"       : math.sqrt(sum(data["diff2"])/len(data)),
69              "SDMRE"      : stdev(data["mre"]),
70              "RRMS"       : (math.sqrt(sum(data["diff2"])/len(data))*len(data))/sum(data[
                    ↪ var_original]),
71              "pred(0.25)": len(data[data["mre"]<=0.25]),
72              "mar"        : sum(data["diff2"])/len(data),
73              "r2"         : model.rvalue**2
74      }
75      return errs
76  #mre    = magnitude of relative error
77  #mmre   = mean magnitude of relative error
78  #rmse   = root of mean square error
79  #sdmre  = mre standard deviation
```

```
80  #rrms    = relative rms
81  #pred    = Prediction level at 25%
82  #mar     = mean absolute residual
83
84
85  #Create and save a scatter graph
86  def curve_graph(dx, dy, xlab, ylab, title, name):
87      plt.scatter(x=dx, y=dy)
88      plt.xlabel(xlab)
89      plt.ylabel(ylab)
90      plt.title(title)
91      plt.grid(linestyle="--", linewidth=0.5, axis="both")
92      plt.savefig(name, dpi=300)
93      plt.show()
94
95  #Create and save a scatter graph with the linear regression line
96  def ln_graph(dx, dy, xlab, ylab, model, x_axis, y_axis, title, name):
97      plt.scatter(x=dx, y=dy)
98      plt.plot(x_axis, list(map(lambda x: model.slope*x+model.intercept, x_axis)))
99      plt.xlabel(xlab)
100     plt.ylabel(ylab)
101     plt.xlim(x_axis[0], x_axis[1])
102     plt.ylim(y_axis[0], y_axis[1])
103     plt.title(title)
104     plt.grid( linestyle="--", linewidth=0.5, axis="both")
105     plt.savefig(name, dpi=300)
106     plt.show()
107
108 #Create and save a scatter graph with the originals and predicted values
109 def prediction_graph(dx, dy, px, py, ex, ey, xlab, ylab, title, name, col3='bgr'):
110     plt.scatter(x=dx, y=dy, c=col3[0], label="Real")
111     plt.scatter(x=px, y=py, c=col3[1], label="Predicción")
112     plt.scatter(x=ex, y=ey, c=col3[2], label="Esperado")
113     plt.xlabel(xlab)
114     plt.ylabel(ylab)
115     plt.legend()
116     plt.title(title)
117     plt.grid(linestyle="--", linewidth=0.5, axis="both")
118     plt.savefig(name, dpi=300)
119     plt.show()
120
121 ##############################################################################################
122 #Read the data and define de lots indexes
123 data = read_data('data.csv')
124 data = data.sort_values(by=["pdr", "esfuerzo"], ascending=False)
125 lots = [0, 4, 8, 15, 21]
126
127 semester = []
```

```
128 for i in range(1, len(lots)):
129     semester.extend([i]*(lots[i]-lots[i-1]))
130 #    print(data[lots[i-1]:lots[i]])
131 data["semestre"] = semester
132 
133 
134 ##############################################################################################
135 #Make calculus for both theories
136 l_curves = pd.DataFrame(columns=["LotQuantity", "LotCost", "first", "CumQuantity", "
        ↪ CumCost"])
137 
138 
139 l_curves["LotQuantity"] = data.groupby(['semestre'], as_index=False).sum()['cfp']
140 l_curves["LotCost"] = data.groupby(['semestre'], as_index=False).sum()['esfuerzo']
141 
142 
143 l_curves['CumQuantity'] = l_curves['LotQuantity'].cumsum(axis=0)
144 l_curves['CumCost'] = l_curves['LotCost'].cumsum(axis=0)
145 l_curves['first'] = l_curves['CumQuantity']-l_curves['LotQuantity']+1
146 
147 l_curves = l_curves[["LotQuantity", "LotCost", "first", "CumQuantity", "CumCost"]]
148 
149 
150 #For UNIT THEORY - ESTIMATING LOTS COSTS
151 l_curves["auc"] = l_curves["LotCost"]/l_curves["LotQuantity"]#AUC is PDR
152 l_curves["lmp"] = l_curves.apply(lambda x: lmp(x["first"], x["CumQuantity"]), axis=1)
153 l_curves["lnAuc"] = list(map(math.log, l_curves["auc"]))
154 l_curves["lnLmp"] = list(map(math.log, l_curves["lmp"]))
155 #For CAT
156 l_curves['cac'] = l_curves['CumCost']/l_curves['CumQuantity']#CAC is Cumulative PDR
157 l_curves['lnCac'] = list(map(math.log, l_curves['cac']))
158 l_curves['lnCumQuantity'] = list(map(math.log, l_curves['CumQuantity']))
159 
160 
161 ##############################################################################################
162 # UNIT THEORY - ESTIMATING LOTS COSTS
163 curve_graph(l_curves["lmp"], l_curves["auc"], "Punto Medio del Lote (LMP)", "Costo 
        ↪ Promedio por Unidad (AUC) - PDR", "", 'lots.png')
164 
165 #Linear regression
166 model_unit_lot = stats.linregress(l_curves["lnLmp"], l_curves["lnAuc"])
167 a_lot = math.exp(model_unit_lot.intercept)
168 b_lot = model_unit_lot.slope
169 print("AUC = %f * (N^%f)"%(a_lot,b_lot))#This is the prediction equation
170 print('slope = %.2f%c'%(100*2**b_lot, 37))
171 
172 l_curves["predPDR"] = l_curves.apply(lambda x: pred(a_lot, b_lot, x["lmp"]), axis=1)
173 
```

```
174 #Logarithmic graph
175 ln_graph(l_curves["lnLmp"], l_curves["lnAuc"],
176          "ln(AUC)", "ln(LMP)", model_unit_lot,
177          [0, 9], [2, 4], "", 'linRegSemesters.png')
178
179 #Estimate the next lot size
180 next_size = round(sum(l_curves["LotQuantity"])/len(l_curves))
181 new_lot_start = l_curves["CumQuantity"][len(l_curves)-1]+1
182 new_lot_end = new_lot_start + next_size
183 next_lmp = lmp(new_lot_start, new_lot_end)
184 next_pdr = pred(a_lot, b_lot, next_lmp)
185 next_estimated_effort = next_pdr*next_size#effort/size * size = effort
186
187 #Prediction graph
188 prediction_graph(l_curves["lmp"], l_curves["auc"],
189                  l_curves["lmp"], l_curves["predPDR"],
190                  next_lmp, next_pdr,
191                  "Punto Medio del Lote (LMP)", "Costo Promedio por Unidad (AUC) - PDR",
192                  "Predicción - Teoría Unitaria", 'lotsPred.png', col3='bgr')
193
194 #Get the errors
195 data_lot = l_curves.copy()
196 get_errors(data_lot, model_unit_lot, 'auc', 'predPDR')
197
198
199 ##########################################################################################
200 # CUMULATIVE AVERAGE THEORY -
201 curve_graph(l_curves["CumQuantity"], l_curves["cac"], "Cantidad Acumulada", "Costo
        ↪ Promedio Acumulado (CAC) - PDR", "", 'catlots.png')
202
203 #Linear regression
204 #This is to predict the cost of a particular lot, not PDR
205 model_cat_lot = stats.linregress(l_curves["lnCumQuantity"], l_curves["lnCac"])
206 a_cat = math.exp(model_cat_lot.intercept)
207 b_cat = model_cat_lot.slope
208 #This is the equation that models the curve
209 print("CAC = %.3f * (N^%.3f)"%(a_cat, b_cat))
210 #Learning curve
211 print('slope = %.3f%c'%(100*2**b_cat, 37))
212 #Calculates the total cost of N units , A*N**b * N = A*N**(b+1)
213 print('CT_n = %.3f * (N^%.3f)'%(a_cat, b_cat+1))
214 #Unit cost N, instantaneous slope, derived from CT_n
215 print('CostUnit_n = (%.3f) * %.3f (N^%.3f)'%(1+b_cat, a_cat, b_cat))
216 #Total cost of a specific lot, firts, last
217 #CT_fl = CT_l - CT_(f-1) = A * (l**(b+1) - (f-1)**(b+1))
218 print('CT_fl = %.3f * (l^(%.3f) - (f-1)^(%.3f))'%(a_cat, b_cat+1, b_cat+1))
219
```

```
220 l_curves["predLotCost"] = l_curves.apply(lambda x: pred_cat(a_cat, b_cat, x['first'], x
        ↪ ['CumQuantity']), axis=1)
221 l_curves["cumPredLotCost"] = None
222 l_curves.iloc[0, 14] = l_curves.iloc[0]["predLotCost"]
223 for i in range(1, len(l_curves)):
224     l_curves.iloc[i, 14] = l_curves.iloc[i-1]["cumPredLotCost"]+l_curves.iloc[i]["
            ↪ predLotCost"]
225 #pdr = efuerzo/size
226 l_curves["cpredPDR"] = l_curves["predLotCost"]/l_curves["LotQuantity"]
227 #cumulative_pdr = cumulative_effort/cumulative_size
228 l_curves["cumPredPDR"] = l_curves["cumPredLotCost"]/l_curves["CumQuantity"]
229
230 #Logarithmic graph
231 ln_graph(l_curves["lnCumQuantity"], l_curves["lnCac"],
232          "ln(Cumulative quantity)", "ln(CAC)", model_cat_lot,
233          [0, 10], [2.5, 4], "", 'catlinRegSemesters.png')
234
235
236 #Estimate the next lot size
237 next_size_cat = round(sum(l_curves["LotQuantity"])/len(l_curves))
238 new_lot_start_cat = l_curves["CumQuantity"][len(l_curves)-1]+1
239 new_lot_end_cat = new_lot_start_cat + next_size_cat
240 next_lot_cost_cat = pred_cat(a_cat, b_cat, new_lot_start_cat, new_lot_end_cat)
241 next_cpdr_cat = (next_lot_cost_cat+l_curves["cumPredLotCost"][len(l_curves)-1])/
        ↪ new_lot_end_cat
242 #next_pdr_cat = next_lot_cost_cat/next_size_cat
243
244 #Prediction graph
245 prediction_graph(l_curves["CumQuantity"], l_curves["cac"],
246                  l_curves["CumQuantity"], l_curves["cumPredPDR"],
247                  new_lot_end_cat, next_cpdr_cat,
248                  "Cantidad Acumulada", "Costo Promedio Acumulado (CAC) - PDR",
249                  "Predicción - Teoría del Promedio Acumulado", 'catlotsPred.png', col3=
                         ↪ 'cym')
250
251 #Get errors
252 data_cat_lot = l_curves.copy()
253 get_errors(data_cat_lot, model_cat_lot, 'cac', 'cumPredPDR')
```

Bibliografía

[1] ISO/IEC 19761:2011. *Software engineering – COSMIC-FFP – A functional size measurement method*. Geneva, 2011.

[2] ISO/IEC 20926:2003. *Software engineering - IFPUG 4.1 Unadjusted functional size measurement method - Counting practices manual*. Geneva, 2003.

[3] ISO/IEC 20968:2002. *Software engineering - Mk II Function Point Analysis - Counting practices manual*. Geneva, 2002.

[4] ISO/IEC 24570:2005. *Software engineering – NESMA functional size measurement method version 2.1 - Definitions and counting guidelines for the application of Function Point Analysis*. Geneva, 2005.

[5] ISO/IEC 29881:2008. *Information Technology – Software and systems engineering – FiSMA 1.1 functional size measurement method*. Geneva, 2008.

[6] Alain Abran. 1st ed. Hoboken, NJ, USA: John Wiley & Sons, 2015. ISBN: 9781118954089. DOI: 10.1002/9781118959312. URL: https://onlinelibrary.wiley.com/doi/book/10.1002/9781118959312.

[7] Alain Abran. *Software Metrics and Software Metrology*. John Wiley & Sons, Ltd, 2010, pp. 129–130. ISBN: 9780470606834. DOI: https://doi.org/10.1002/9780470606834.part2. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/9780470606834.

[8] Armen Albert Alchian. *Reliability of Progress Curves in Airframe Production*. Santa Monica, CA: RAND Corporation, 1963, 679–693.

[9] Carina Andersson y Per Runeson. "A Spiral Process Model for Case Studies on Software Quality Monitoring - Method and Metrics". eng. In: Software Process Improvement and Practice 12.2 (2007), pp. 125–140. ISSN: 1077-4866.

[10] Harold Asher. *Cost-Quantity Relationships in the Airframe Industry*. Santa Monica, CA: RAND Corporation, 1956.

[11] *Asociación Mexicana de Métricas de Software*. https://www.amms.org.mx/. [Online; accessed 5-June-2020]. 2020.

[12] Izak Benbasat, David K. Goldstein y Melissa Mead. "The Case Research Strategy in Studies of Information Systems". In: *MIS Q.* 11.3 (Sept. 1987), pp. 369–386. ISSN: 0276-7783. DOI: 10.2307/248684. URL: https://doi.org/10.2307/248684.

[13] Pierre Bourque. *Software Engineering Body of Knowledge (SWEBOK Guide)*. Sept. 2010. DOI: 10.13140/RG.2.2.12178.27846.

[14] Pinar Efe y Onur Demirors. “Applying EVM in a Software Company: Benefits and Difficulties”. In: Sept. 2013, pp. 333–340. DOI: 10.1109/SEAA.2013.55.

[15] Epple, D., Argote, L., and Murphy, K. *An empirical investigation of the micro structure of knowledge acquisition and transfer through learning by doing*. Operations Research, 1996, 77–86.

[16] Michelle Texis Flores y Alejandro Mungaray Lagarda. “Aprendizaje en microempresas de Baja California”. In: *Estudios Fronteriza, nueva época, vol 12, num.23*. Universidad Autonoma de Baja California, 2011, pp. 95–116.

[17] Bent Flyvbjerg. “Five Misunderstandings About Case-Study Research”. In: vol. 12. Nov. 2006, pp. 390–404. ISBN: 1412934206. DOI: 10.4135/9781848608191.d33.

[18] H Gruber. *The yield factor and the learning curve in semiconductor production*. Applied Economics, 1994, 837–843.

[19] Robert Hanna. “Earned Value Management Software Projects”. In: *Space Mission Challenges for Information Technology, IEEE International Conference on* 0 (July 2009), pp. 297–304. DOI: 10.1109/SMC-IT.2009.42.

[20] Jo Hannay, Dag Sjøberg y Tore Dybå. “A Systematic Review of Theory Use in Software Engineering Experiments”. In: *Software Engineering, IEEE Transactions on* 33 (Mar. 2007), pp. 87–107. DOI: 10.1109/TSE.2007.12.

[21] W. B. Hirschmann. *Profit from the learning curve*. Harvard Business Review, 1964, 125–139.

[22] *History*. https://cosmic-sizing.org/cosmic-sizing/functional-size-measurement/history/. [Online; accessed 5-June-2020]. 2020.

[23] Trista Hollweck. “Robert K. Yin. (2014). Case Study Research Design and Methods (5th ed.). Thousand Oaks, CA: Sage. 282 pages.” In: *The Canadian Journal of Program Evaluation* (Mar. 2016). DOI: 10.3138/cjpe.30.1.108.

[24] Saavedra Martínez Jesús Iván. 2020. URL: https://tesiunam.dgb.unam.mx/F/H5V78822IRMA2DY116KCEKVJFF5E1KB1A7IU7U83CUHNV2ASL7-07126?func=full-set-set&set%5C_number=691674&set%5C_entry=000003&format=999.

[25] Capers Jones. “Impact of Software Size on Productivity”. In: ISBSG, 2013.

[26] Linda M. Laird y M. C. Brennan. “Software Measurement and Estimation: A Practical Approach”. In: NJ, USA: John Wiley & Sons, 2006.

[27] Allen S. Lee. “A Scientific Methodology for MIS Case Studies”. In: *MIS Quarterly* 13.1 (1989), pp. 33–50. ISSN: 02767783. URL: http://www.jstor.org/stable/248698.

[28] Timothy Lethbridge, Susan Sim y Janice Singer. “Studying Software Engineers: Data Collection Techniques for Software Field Studies”. In: *Empirical Software Engineering* 10 (July 2005), pp. 311–341. DOI: 10.1007/s10664-005-1290-x.

[29] M. B. Lieberman. *he learning curve and pricing in the chemical processing industries*. The Rand Journal of Economics, 1984, 213–228.

[30] *Matplotlib*. https://matplotlib.org/. [Online; accessed 21-April-2021]. 2021.

[31] Gregory K. Mislick y Daniel A. Nussbaum. *Cost Estimation Methods and Tools*. 1st ed. Wiley, 2015.

[32] *pandas*. https://pandas.pydata.org/. [Online; accessed 21-April-2021]. 2021.

[33] *Python*. https://www.python.org/. [Online; accessed 21-April-2021]. 2021.

[34] Panagiota Ralli et al. "Comparative Evaluation of Learning Curve Models for Construction Productivity Analysis". In: *The 10th International Conference on Engineering, Project, and Production Management*. Ed. by Kriengsak Panuwatwanich y Chien-Ho Ko. Singapore: Springer Singapore, 2020, pp. 347–358. ISBN: 978-981-15-1910-9.

[35] L Rapping. *Learning and World War II production functions*. Review of Economics and Statistics, 1965, 81–86.

[36] *Real Academia Española*. https://dle.rae.es/productividad. [Online; accessed 21-April-2021]. 2021.

[37] Colin Robson. *Real world research : a resource for social scientists and practitioner-researchers*. eng. 2nd ed.. Oxford: Blackwell Publishers, 2002. ISBN: 9780631213055.

[38] Per Runeson y Martin Host. *Guidelines for conducting and reporting case study research in software engineering*. Springerlink, 2008. DOI: 10.1007/s10664-008-9102-8.

[39] *SciPy*. https://www.scipy.org/. [Online; accessed 21-April-2021]. 2021.

[40] *Second Generation*. https://cosmic-sizing.org/cosmic-sizing/functional-size-measurement/second-generation/. [Online; accessed 5-June-2020]. 2020.

[41] Forrest Shull y Raimund L. Feldmann. "Building Theories from Multiple Evidence Sources". In: *Guide to Advanced Empirical Software Engineering*. Ed. by Forrest Shull, Janice Singer y Dag I. K. Sjøberg. London: Springer London, 2008, pp. 337–364. ISBN: 978-1-84800-044-5. DOI: 10.1007/978-1-84800-044-5_13. URL: https://doi.org/10.1007/978-1-84800-044-5_13.

[42] J. Singer y N. G. Vinson. "Ethical issues in empirical studies of software engineering". In: *IEEE Transactions on Software Engineering* 28.12 (2002), pp. 1171–1180. DOI: 10.1109/TSE.2002.1158289.

[43] Rodney D Stewart. *Cost Estimating*. 2nd ed. Haarleem, Netherlands: John Wiley & Sons, 1991.

[44] Charles Symons. 2020. URL: https://cosmic-sizing.org/publications/measurement-guide/.

[45] CMMI Product Team. *CMMI for Development, Version 1.3*. Tech. rep. CMU/SEI-2010-TR-033. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2010. URL: http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=9661.

[46] *The COSMIC Functional Size Measurement Method: Measurement Manual*. v. 4.0.2. 2017. URL: http://www.cosmic-sizing.org.

[47] D. R. Towill y J. E. Cherrington. “Learning curve models for predicting the performance of AMT”. In: *The International Journal of Advanced Manufacturing Technology*. Springer-Verlag London, 1994, pp. 195–203. DOI: 10.1007/BF01754598.

[48] Francisco Valdés, Daniel Torres-Robledo y Hanna Jadwiga-Oktaba. “Comparing two learning curves approaches to predict the Product Delivery Ratein a software factory contract”. In: *Programming and Computer Software - Special Issue 2021* (2021 - En revisión).

[49] Francisco Valdés, Daniel Torres-Robledo y Hanna Jadwiga-Oktaba. “Product Delivery Improvement in a Software Factory Contract Applying Learning Curves”. In: *IWSM-Mensura*. Vol. 2725. IWSM MENSURA, 2020. URL: https://www.iwsm-mensura.org/proceedings-2020/.

[50] Francisco Valdés Souto. “Competitividad en contratos de software”. In: (Apr. 2012).

[51] Francisco Valdés Souto. “Earned Scope Management: Scope Performance Evaluation for Software Projects Considering People and Effort as Resources”. In: Oct. 2019, pp. 213–222. DOI: 10.1109/CONISOFT.2019.00038.

[52] Francisco Valdés-Souto. *Impacto del Tamaño de Software en la Productividad para la Industria Mexicana de desarrollo de Software*. México, CDMX: Asociación Mexicana de Métricas de Software (AMMS), 2018. ISBN: 03-2018-080210560800-01.

[53] Francisco Valdés-Souto. *Validation of supplier estimates using COSMIC method*. Haarleem, Netherlands: CEUR Workshop Proceedings, 2019. ISBN: 1613-0073.

[54] Frank Vogelezang, ed. *Early Software Sizing with COSMIC: Experts Guide*. 2nd ed. The Netherlands, February 27, 2020. DOI: 10.13140/RG.2.1.4195.0567.

[55] Frank Vogelezang y Harold van Heeringen. “Benchmarking: Comparing Apples to Apples”. In: *Rethinking Productivity in Software Engineering*. Ed. by Caitlin Sadowski y Thomas Zimmermann. Berkeley, CA: Apress, 2019, pp. 205–217. ISBN: 978-1-4842-4221-6. DOI: 10.1007/978-1-4842-4221-6_18.

[56] T. Wright. “Factors affecting the cost of airplanes”. In: *Journal of the Aeronautical Sciences* 3 (1936), pp. 122–128.