EDITORIAL
NAVEGANTE



LA PRODUCTIVIDAD
Y SU INFLUENCIA EN LAS
FÁBRICAS DE SOFTWARE

PEDRO SEGUNDO CASTAÑEDA VARGAS
ITALO MALDONADO RAMIREZ
ROBERTO CARLOS SANTA CRUZ ACOSTA

LA PRODUCTIVIDAD Y SU INFLUENCIA EN LAS FÁBRICAS DE SOFTWARE

PEDRO SEGUNDO CASTAÑEDA VARGAS
ÍTALO MALDONADO RAMIREZ
ROBERTO CARLOS SANTA CRUZ ACOSTA

Pedro Segundo Castañeda Vargas

Correo de contacto: pcsipcas@upc.edu.pe

ORCID: https://orcid.org/0000-0003-1865-1293

Afiliación: Universidad Peruana de Ciencias Aplicadas (UPC)

Perú

Ítalo Maldonado Ramírez

Correo de contacto: Italo.Maldonado@untrm.edu.pe

ORCID: https://orcid.org/0000-0002-3147-3519

Afiliación: Universidad Nacional Toribio Rodríguez de Mendoza de Amazonas

Perú

Roberto Carlos Santa Cruz Acosta

Correo de contacto: roberto.santacruz@untrm.edu.pe

ORCID: https://orcid.org/0000-0001-8802-9083

Afiliación: Universidad Nacional Toribio Rodríguez de Mendoza de Amazonas

Perú

Todas nuestras publicaciones son sometidas a revisión doble-ciego de pares académicos *(Peer Review Double Blinded)*.

Esta publicación cuenta con licencia Creative Commons Reconocimiento - NoComercial - SinObra-Derivada 3.0 Unported License



ISBN 978-628-95261-9-6

© Pedro Segundo Castañeda Vargas

© Ítalo Maldonado Ramírez

© Roberto Carlos Santa Cruz Acosta

© Editorial Navegante

Cra 22 N.º 35-40 Oficina 217, Edificio Apolo, Bucaramanga, Colombia

www.editorialnavegante.com

Este libro es resultado de la investigación *Educación para el Trabajo en el Pensamiento Crítico – Creativo en Estudiantes de la Institución Educativa Politécnico Regional del Centro Huancayo,* realizada en la Universidad Nacional de Huancavelica, Perú.

Queda prohibida la reproducción bajo cualquier modalidad de toda o una parte de esta obra sin autorización expresa del titular de los derechos

Diseño de carátula y composición: Rodrigo Medrano Serrano

Edición electrónica: Editorial Navegante

Editado en Colombia

Published in Colombia

ÍNDICE

Índice de tablas 4

Índice de figuras 5

PRESENTACIÓN 6

CAPÍTULO I NOCIONES GENERALES DE LA FÁBRICA DE SOFTWARE 7

1.1. Fábrica de software 7

1.2. Estructura de una fábrica de software 8

1.3 El proceso de desarrollo de software 9

1.4 Ventajas de implementar softwares en las empresas 12

CAPÍTULO II PRODUCTIVIDAD Y CATEGORIZACIÓN DE FACTORES DE LAS FÁBRICAS DE SOFTWARE 13

2.1 Productividad en las fábricas de software 13

2.2 Categorización de factores influyentes en la productividad 16

2.2.1. Producto 16

2.2.2. Proceso 16

2.2.3. Entorno de desarrollo 17

2.2.4. Cultura corporativa 17

2.2.5. Cultura del equipo 18

2.2.6. Capacidades y experiencia 18

2.2.7. Proyecto 18

CAPÍTULO III MODELOS DE MEDICIÓN EN LA PRODUCTIVIDAD DE LAS FÁBRICAS DE SOFTWARE 19

3.1. Análisis envoltorio de datos (DEA) 19

3.2. Conceptos clave del sistema DEA 20

3.3. Modelo CMMI y Six Sigma 21

3.4. Modelo de lógica difusa 22

3.5. Modelo de reutilización del software 22

3.6. Modelo SEM 23

3.7. Modelo factor total de productividad 23

3.8. Modelo ISO 9126 y IEEE Std. 1045 24

CAPÍTULO IV MODELO PARA LA MEDICIÓN DE LA PRODUCTIVIDAD DE LA FÁBRICA DE SOFTWARE 25

4.1. Metodología 26

4.2. Análisis de los resultados 29

4.3. Fábrica de software 30

4.4. Planificación de la revisión 33
4.4.1. Criterios de selección 34
4.5. Estadísticas 36
4.5.1. Tendencia temporal de publicaciones 36
4.5.2. Número de publicaciones seleccionadas 36
4.6. Análisis 37
4.6.1. Q1: ¿Qué es la productividad de fábricas de software? 37
4.6.2. Q2: ¿Qué factores influyen en la productividad y cómo se clasifican? 37
4.6.3. Q3: ¿Qué modelos se han desarrollado? 40
4.6.4. Q4: ¿Qué factores incluyen los modelos? 45
4.7. Modelo conceptual 48
4.7.1. Unidades de trabajo 48
4.7.2. Factores propuestos 51
4.7.3. Hipótesis 54
4.8. Discusión 58
CAPÍTULO V MODELO DEA PROPUESTO PARA LA MEDICIÓN DE LA PRODUCTIVIDAD DE LA FÁBRICA DE SOFTWARE 62
5.1 Modelos de productividad DEA 62
5.2 Metodología 64
5.2.1. Elección del modelo de la DEA 64
5.2.2. Parámetros para el modelo DEA 65
5.3. Fuente de datos 69
5.3.1. Recopilación de datos 69
5.3.2. Análisis descriptivo de la muestra 69
5.4. Resultados empíricos e interpretaciones 72
5.5. Procedimiento de medición 78
CAPÍTULO VI IMPORTANCIA DE LA MEDICIÓN DE LA PRODUCTIVIDAD EN LAS FÁBRICAS DE SOFTWARE 82
CAPÍTULO VII CONSIDERACIONES FINALES 84
REFERENCIAS BIBLIOGRÁFICAS 87

ÍNDICE DE TABLAS

Tabla 1. Destino de la producción de software en Argentina, 2010 15
Tabla 2. Estructura organizacional de fábricas de software 32
Tabla 3. Cadenas de búsqueda utilizada en las bases de datos 34
Tabla 4. Criterios de inclusión y exclusión 35
Tabla 5. Estudios potencialmente elegibles y seleccionados 35
Tabla 6. Distribución por tipo de publicación y base de datos 36
Tabla 7. Clasificación de factores por categoría 39
Tabla 8. Cantidad de factores por componente y categoría que impactan en la productividad 39
Tabla 9. Modelos de medición de la productividad de software 41
Tabla 10. Otros modelos de medición de la productividad de software no seleccionados 44
Tabla 11. Evaluación de modelos de productividad de software 44
Tabla 12. Modelos de medición de la productividad de software por componente 45
Tabla 13. Factores incluidos en los modelos de productividad de software por componente 45
Tabla 14. Componentes de la fábrica de software 49
Tabla 15. Unidades de trabajo de los componentes de la fábrica de software 50
Tabla 16. Nuevos factores que impactan en la productividad 53
Tabla 17. Hipótesis planteadas para la definición de factores que afectan la productividad en el desarrollo de software 55
Tabla 18. Regresión de la variable dependiente (ADI) con todas las variables independientes 55
Tabla 19. Regresión de la variable dependiente (ADI) con las variables independientes significativas 56
Tabla 20. Regresión de la variable dependiente (PRO) con todas las variables independientes 56
Tabla 21. Regresión de la variable dependiente (PRO) con las variables independientes significativas 57
Tabla 22. Regresión de la variable dependiente (TES) con todas las variables independientes 57
Tabla 23. Regresión de la variable dependiente (TES) con las variables independientes significativas 58
Tabla 24. Estudios de proyectos de software aplicando DEA 63
Tabla 25. Información acerca de fábricas de software 68
Tabla 26. Fábricas de software considerados en la muestra 70
Tabla 27. Estadísticas descriptivas para variables de entrada y salida-Fábricas de software 70
Tabla 28. Proyectos de fábricas de software considerados en la muestra 71

Tabla 29. Estadísticas descriptivas para variables de entrada y salida-Proyectos de fábricas de software 72
Tabla 30. Tabla de abreviaturas 72
Tabla 31. Eficiencia DEA VRS-I, fábrica de software 73
Tabla 32. Eficiencia DEA VRS-I, fábrica de software 74
Tabla 33. Eficiencia DEA VRS-I, Proyectos de fábrica de software 75
Tabla 34. Eficiencia promedio DEA VRS 76
Tabla 35. Caracterización de los proyectos de fábrica de software 77
Tabla 36. Conjunto de referencias VRS para los proyectos de fábrica de software 78
Tabla 37. Responsabilidades de los participantes en el procedimiento 79
Tabla 38. Entradas y salidas 79
Tabla 39. Actividades del proceso de medición de la productividad 80
Tabla 40. Formato de revisión 81

ÍNDICE DE FIGURAS

Figura 1. Esquema de ejecución con el equipo de fábrica de software 10
Figura 2. Esquema general de productividad 14
Figura 3. Factores determinantes de la productividad laboral 16
Figura 4. Empresas acreditadas por países 26
Figura 5. Modelo conceptual de la productividad 30
Figura 6. Estructura de una fábrica de software 33
Figura 7. Proceso de revisión de la literatura 35
Figura 8. Número de publicaciones sobre productividad de software por año y base de datos 36
Figura 9. Datos de la caracterización de la muestra 47
Figura 10. Modelo conceptual inicial de los factores que afectan la productividad 54
Figura 11. Modelo final de factores que afectan la productividad 61
Figura 12. Parámetros para el modelo DEA 65
Figura 13. Fábricas de software consideradas en la muestra 70
Figura 14. Línea de regresión estadística para fábricas de software 71
Figura 15. Resumen de eficiencias 73
Figura 16. Resultados para SF004 74
Figura 17. Conjunto de referencias VRS para la fábrica de software 75
Figura 18. Cantidad de veces que un proyecto es referencia de otro proyecto 76
Figura 19. Cactores y su influencia en los proyectos de fábrica de software eficientes 76
Figura 20. Proceso para la medición de la productividad 80

INTRODUCCIÓN

Una empresa moderna enfocada en su propio desarrollo tiene como exigencia ir a la par del crecimiento tecnológico suscitado a lo largo de los años a nivel mundial. En otros términos, si se trata de una compañía preocupada por un crecimiento económico, debe invertir en la adquisición de recursos especializados que brinden un mejor desempeño a su equipo de trabajo, así como una grata experiencia a sus consumidores.

En esa línea y basado en dichos requerimientos, surge la empresa de la industria del software o, en otros términos, la fábrica de *software*, organización que se dedica al diseño y desarrollo de aplicaciones que puede permitir al cliente un acceso fácil de la información que se quiere tener sobre un producto o la adquisición de un bien o servicio.

Actualmente, entre las compañías que ofrecen un servicio de *software* a nivel mundial se encuentran Microsoft, IBM, Oracle, entre otros. Estas, como ya se señaló, se encargan de elaborar sistemas informáticos que realizan actividades específicas dentro de compontes físicos (*hardware*). Este servicio se ha convertido en uno de los más solicitados por las empresas, lo cual ha llevado a que se dé una mayor producción de sistemas *software* en el sector informático. La gran demanda que suscita un programa informático está relacionada también con las ansias de competir por parte de las empresas, así como la producción de un proceso minucioso que garantice la calidad del producto que se ofrece, concepto que engloba a la productividad.

Hacer mención a la productividad involucra una serie de procesos y elementos que trabajan en conjunto para llevar a cabo la culminación de un producto. Bastaría decir, entonces, que a mayor productividad mejores ingresos para una empresa. Sin embargo, el concepto relacionado con la producción es, ciertamente, más complejo de lo que parece, pues presenta dificultades para una adecuada medición, más aún cuando se trata de medir el nivel de productividad de los desarrolladores informáticos.

Hacer mención a la productividad involucra una serie de procesos y elementos que trabajan en conjunto para llevar a cabo la culminación de un producto. Bastaría decir, entonces, que a mayor productividad mejores ingresos para una empresa. Sin embargo, el concepto relacionado con la producción es, ciertamente, más complejo de lo que parece, pues presenta dificultades para una adecuada medición, más aún cuando se trata de medir el nivel de productividad de los desarrolladores informáticos.

Al respecto, Nemur (2016) afirma que la productividad es "el arte de ser capaz de crear, generar o mejorar bienes y servicios o en otros términos en una manera de medir en promedio la eficiencia de la producción" (p. 2). Sin embargo, realizar la medida exacta en cuestiones de productividad no es una actividad fácil, por lo que diversos investigadores buscarán proponer un modelo que mejor pueda realizarse. Uno de ellos es el modelo basado en el análisis envoltorio de datos (DEA) que mide la productividad en dos fases: (1) en la fábrica de *software* y (2) en sus respectivos proyectos.

Por ello, el objetivo principal de este libro de investigación, además de exponer información actualizada sobre las investigaciones realizadas en relación con la fábrica de *software* y la productividad, entre otros; es diseñar un modelo que permita medir dicho proceso de la fábrica de *software*, a fin de determinar la efectividad del uso de los recursos productivos para el cumplimiento de las metas organizacionales.

CAPÍTULO I
NOCIONES GENERALES DE LA FÁBRICA DE SOFTWARE

El sistema económico con el que se rige el mundo se puede graficar como un mercado de intercambio ya sea de bienes o servicios. En este es fácil reconocer cuándo un producto es de calidad o posee deficiencias. Por ello, cada sector económico existente se esmera por ofrecer al consumidor un resultado óptimo, para lo cual el bien o servicio atraviesa una serie de procesos que garanticen la confiabilidad. Bajo los mismos parámetros surgen las fábricas de *software*, empresas encargadas de elaborar programas para un adecuado funcionamiento de las computadoras. De esta manera, entre sus funciones más relevantes se encuentran las de brindar un sistema software de seguridad a bancos, entidades educativas, etc.

De ello se desprende que la producción de *software* es más que la implementación de un programa, esta involucra aspectos como seguridad, eficiencia e incluso comodidad para el cliente en tiempos en los que la modernidad y la tecnología se relacionan bidireccionalmente.

1.1. Fábrica de software

Para conocer a detalle la definición de fábrica de *software* es necesario esclarecer algunos términos preliminares. El sistema informático está conformado principalmente por dos compontes base (Gómez y Moraleda, 2020): el *hardware*, elemento de fácil identificación por ser visible al ojo humano, hace referencia directa a los aparatos físicos; y el *software*, que comprende los programas que permiten el funcionamiento de los equipos, además de documentos, bases de datos, procedimientos de operación, entre otros.

Aunque se puede pensar que las gestiones de estos elementos funcionan de forma aislada, cabe señalar que ambos componentes informáticos son considerados productos intercambiables en el mercado económico. En el caso del *hardware* mediante la compra y venta de monitores, teclados, *mouse*, etc., y en el caso del *software*, como "un procesador de texto o un programa de tratamiento de imágenes [...] o el programa que gobierna la inyección de gasoil de un motor diésel o un programa que permite la transferencia bancaria" (Gómez y Moraleda, 2020, p. 20).

Si bien actualmente existe una gran producción de sistemas informáticos, a inicios de su industrialización, los productos eran elaborados de manera artesanal; es decir, estaban a cargo de personas con grandes habilidades informáticas y no se fabricaban en masa. Sin embargo, con el paso del tiempo, la industria comenzó a comercializar un mayor número de ordenadores, hecho que generó una gran demanda de programas como aplicaciones desarrolladas, soporte, entre otros. Con ello se consolidó la formación de empresas dedicadas a elaborar productos informáticos, lo cual dio paso a la fábrica de *software*.

Una fábrica de *software* es una industria que tiene como objetivo desarrollar softwares específicos de acuerdo con los requerimientos de los clientes. Esta generará ingresos mediante la venta de proyectos de desarrollo, es decir, no requiere de la elaboración de un diseño, sino que puede replicarlos. En términos de Gómez y Moraleda (2020), "el *software* tiene una particularidad especial frente a cualquier producto físico que podamos imaginar: una vez diseñado, este se puede replicar con tremenda facilidad, sin necesidad de un proceso de fabricación propiamente dicho" (p. 23). Esta particularidad permitirá a las empresas obtener una implementación de *software* más inmediata, de manera que asegurarán la modernidad de su producto o servicio. Esta característica beneficia al

empresario que conoce los niveles altamente competitivos que existen en el mercado económico.

Por otra parte, para Monsalve (2013, citado en Garces, 2016), una fábrica de *software* es como un ambiente laboral de alta productividad que se enfoca en el proceso industrial de productos *software*. De ello se puede inferir que los productos generados poseerán índices que validen su calidad, por lo que el empresario deberá seguir procesos entre los que se involucra la revisión paso a paso sobre el resultado final. Por ello, para los productores de un *software* el concepto *mantenimiento* no es novedoso, pues dentro del servicio que ofrecen se encuentra el de validar el producto ofrecido.

Por otro lado, Cabach (2006) propone que "una fábrica de *software* es una línea de productos de *software* configurada por herramientas genéricas y procesos, [...] para automatizar el desarrollo y el mantenimiento de un producto, adaptando, ensamblando y configurando componentes a base de marcos de trabajo" (p. C-210).

1.2. Estructura de una fábrica de software

Las empresas o fábricas que conforman el sistema económico poseen estructuras independientes que les permite organizarse de tal manera que la elaboración del producto que ofrecen al mercado cumpla con las exigencias del cliente. En el sector informático, existe también una organización que, de acuerdo con Nomura *et al.* (2011, citado en Ambrocio, 2017), tiene los siguientes componentes:

- Servicio al cliente, que está conformado por gerentes y analistas del área de negocios, ventas e interacción con el cliente. Es el área responsable de gestionar acuerdos de negociación para la posterior elaboración de un contrato.

- Oficina de proyecto, que se encarga de supervisar las gestiones que debe realizar el proyecto. En esta área se desarrollan los planes que se ejecutarán a futuro.

- Planificación y control de la producción (PPC), sección encargada de gestionar solicitudes de servicio. En esta área se determinará qué requerimientos tener en cuenta para cada solicitud de trabajo. Es decir, se asignará las acciones adecuadas para un buen servicio de implementación de *software*.

- Área de proceso, responsable de diseñar prototipos preliminares y el plan de prueba con el que se ejecutará el proyecto. Asimismo, está a cargo de la elaboración del documento de la visión, el alcance y el glosario.

- Área de análisis y diseño, se encarga de elaborar la infraestructura que tendrá el sistema desde lo más simple a lo más complejo, así como los detalles en relación con los informes y el modelo lógico de datos.

- Área de implementación, responsable de codificar los elementos que formarán el sistema *software*. Se encarga de verificar que los componentes estén completos.

- Área de prueba y homologación que realizará las pruebas de integración. Está encargada de garantizar la calidad y la verificación final del producto.

- Área de soporte que brinda servicio técnico en las distintas áreas que conforman la fábrica de *software*. Tiene la función de desarrollar tecnología para la empresa, así como su mantenimiento.

• Área de infraestructura que se encargará de sostener los recursos de la fábrica sean *hardware* o *software* con los que se podrán llevar a cabo los procesos de producción.

• Área de calidad, encargada de garantizar la calidad del proceso, productos, metodologías, etc., que constituyan una empresa.

1.3. El proceso de desarrollo de software

La elaboración de desarrollo de *software* está sustentada en lineamientos relacionados con la ciencia de la computación y las matemáticas. Se caracteriza por ser aplicativo, sistemático, disciplinado y cuantificable en su conformación (Noriega, 2015); sin embargo, no siempre contó con dichas características.

Las primeras implementaciones de *software* presentaban fallas en la entrega del producto final. Esto debido al cambio continuo del personal de una empresa o el tiempo de duración que presentaba una implementación, la cual podía ser de tres a cinco años, ya que, según Noriega (2015), esta cuenta con un ciclo de vida o proceso de *software*, por lo que se debe considerar, además, un acompañamiento por parte de la empresa desarrolladora.

Por lo tanto, es importante conocer qué procesos existen en el desarrollo de un sistema informático. Sobre este punto hay diferentes enfoques, a continuación, se evidenciarán los más importantes, según Noriega (2015):

a. Investigar los requisitos de los usuarios

Usualmente, la persona que solicita el servicio de un desarrollador tiene ideas generales sobre el tipo de servicio que desea adquirir. Ello dificulta, en cierta medida, la elaboración del servicio porque el cliente no tiene muy claro lo que requiere; sin embargo, no es una limitante. En esta primera etapa, se recomienda escuchar atentamente la solicitud del cliente para que luego se puedan realizar preguntas que permitan conocer a profundidad cuál es su necesidad. Con ello se podrá comenzar a escribir un documento con los requerimientos señalados.

b. Definir claramente las características necesarias para el sistema (especificación)

En esta etapa, el principal objetivo es establecer cuáles son las especificaciones con las que debe contar el *software* a elaborar. Esta es considerada la última fase del análisis previo y se debe realizar en constante comunicación con el cliente. De esa forma no habrá problemas al momento de implementar el proyecto.

c. Crear o adaptar una solución adecuada al problema

Se debe iniciar la elaboración del proyecto usando técnicas estandarizadas basadas en la experiencia del profesional desarrollador. En esta fase, el cliente podrá ver mediante un documento cuál es la propuesta clara y concisa que le ofrecerá la fábrica para implementar el *software*.

d. Desarrollar la solución propuesta

Se debe elaborar y escribir el código utilizado en la implementación del *software* en un documento aparte. De esa manera, se documentará toda la información sobre dicho proceso, y si existe algún

error, podrá detectarse con facilidad. Después de finalizar esta etapa, el proyecto quedará listo para ser aprobado.

e. Garantizar que la solución responde al problema originalmente propuesto

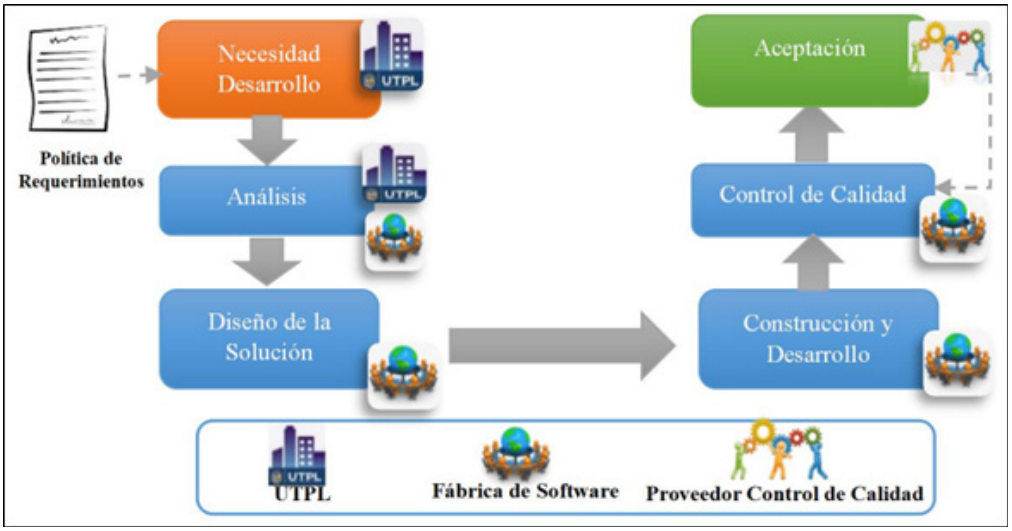
El profesional encargado deberá demostrar en esta fase que la implementación corresponde a lo que el cliente ha solicitado. Se debe hacer pruebas que validen lo que la implementación indica que ofrece.

f. Modificar las soluciones de trabajo ante los nuevos requerimientos

Finalmente, si el desarrollo de un *software* queda desfasado, la fábrica debe considerar la actualización por un periodo de tiempo que se debe detallar en el documento del proyecto.

Por otro lado, Neyra *et al.* (2017) proponen otro esquema de trabajo de desarrollo de *software* que se caracteriza por las siguientes etapas (Figura 1):

Figura 1.
Esquema de ejecución con el equipo de fábrica de *software*



Nota. Adaptado de Neyra *et al.* (2017)

a. Necesidad de desarrollo

En esta etapa se establece el proyecto de acuerdo a la política institucional de requerimientos, los cuales quedan definidos por las especificaciones del producto. Esta también puede estar conformada por solicitudes emergentes o de mantenimiento. Sea cual sea el caso, este debe pasar al comité técnico.

b. Análisis

Esta etapa trabaja con dos elementos importantes de la fábrica de *software*: un analista técnico y un miembro del equipo con el que se priorizará los requerimientos del proyecto, quienes deben determinar si el producto puede desarrollarse en la fábrica de *software* o no. A partir de ello, deben decidir si se puede construir todo o solo una parte en la fábrica. Sin embargo, también se puede construir un producto que no esté apto, pero se tendría que cambiar el esquema y plantilla de la fábrica para acomodar el proceso al nuevo producto y sus requerimientos.

c. Diseño de la solución

El equipo de arquitectura se reúne con otros miembros del equipo para registrar las diferencias de requerimiento del cliente y aprobar la solución y el diseño designados en ellos, con el fin de producir una arquitectura de producto y un proceso de desarrollo de productos personalizados. Además, estos deben cumplir con estándares de calidad que garanticen soluciones idóneas para su funcionalidad.

d. Construcción y desarrollo

En esta etapa se define la cantidad de desarrolladores a cargo del proyecto, así como las fechas en las que se entregarán los avances que permitan conocer al cliente el proceso de su solicitud o del servicio requerido. Esta etapa implica las actividades de desarrollo y construcción de piezas, la realización de pruebas y el acople de los componentes. Para ello se pueden utilizar una serie de diferentes mecanismos, siempre teniendo en cuenta las diferencias del diseño.

e. Control de calidad

Es el momento en el que el líder de la fábrica realizará la inspección de calidad de servicio, así como de los procesos a través de algunas pruebas; es decir, verificará si estos funcionan adecuadamente, tal como el cliente lo ha solicitado.

f. Aceptación

En esta última etapa, el proveedor, después de evaluar una serie de requisitos, validará si el producto que le ofrecen es el mismo que ha solicitado. De ello, se puede deducir que tanto la propuesta de Neyra *et al.* (2017) como la de Noriega (2015) siguen un proceso que, si bien difiere en la clasificación, se orienta con el objetivo de brindar un buen servicio e implementación de *software*.

Por otro lado, existe un modelo de proceso de solo dos etapas propuesto por Alzate *et al.* (2008, pp. 14-15), quienes consideran que son las etapas fundamentales en la elaboración de un *software*:

a. Composición

En esta etapa, la fábrica de *software* define una serie de activos, los cuales componen una base de la aplicación a generar. Esta base en su mayoría cumple con las características no funcionales que debe tener la aplicación como son *logger*, seguridad, manejo de excepciones, entre otros.

b. Generación

Etapa en la que se crean las aplicaciones y se genera automáticamente su código fuente, se utiliza como base una arquitectura de referencia y los insumos proporcionados por los diseñadores de la aplicación a generar, esto quiere decir que la fábrica tiene autonomía e inteligencia para decidir qué archivos y funcionalidades debe crear de acuerdo a lo solicitado por el desarrollador.

Sin embargo, en este proceso no se considera la interacción que debe existir entre la fábrica y el cliente para la elaboración del *software* de tal manera que cumpla con todos los requerimientos. Asimismo, deja fuera del proceso al desarrollador como si este solo se encargara del diseño del *software*, mas no de la implementación en la empresa solicitante.

Cabe señalar que todas las fábricas de *software* buscan la reutilización del código fuente y reducir tiempos en cada etapa del proceso de desarrollo del *software*. Al tener una aplicación ya desarrollada, solo sería necesario modificarla agregando algunas funciones nuevas y específicas, según el requerimiento del cliente, y no rehacer todo desde cero.

1.4. Ventajas de implementar softwares en las empresas

La implementación de un sistema *software* en una empresa le permitirá no solo brindar un servicio de calidad, sino brindará a los trabajadores la posibilidad de realizar funciones diarias en un corto tiempo. Ello se traducirá en una mayor productividad, además de la sistematización de los procesos en la empresa.

De acuerdo a Osorio (2016), el apoyo tecnológico en la institución permitirá que esta le brinde al público un servicio más ágil, con lo cual el cliente podrá obtener una mejor administración de su tiempo. Así, por ejemplo, si la implementación de *software* se da en un centro educativo, esta herramienta tecnológica brindará un aprendizaje más dinámico, por tanto, un mayor rendimiento académico.

Al respecto, Coello (2018) indica que se debe considerar alguna manera de sintetizar información que ofrece el surgimiento y desarrollo de Internet, pues es posible que la persona que requiera realizar gestiones o adquirir información complique el proceso de su actividad. Esta también podría considerarse una situación en la que una adecuada implementación de *software* no cae, pues la diversidad de sistemas que existen puede enfocarse en resolver dicha problemática desde distintas áreas.

En conclusión, y con base en lo planteado por los autores citados anteriormente, el implementar un *software* en una empresa tiene como ventajas el aumento de la productividad, la agilización de los procesos administrativos, la facilidad en el uso y acceso a la información, lo que permite realizar un análisis detallado. Si bien es importante implementar el *software* en las empresas, también se debe considerar capacitar al personal para que no solo pueda manejar el *software*, sino para que en caso se presente un problema puedan solucionarlo sin necesidad de recurrir al fabricante del *software*.

CAPÍTULO II

PRODUCTIVIDAD Y CATEGORIZACIÓN DE FACTORES DE LAS FÁBRICAS DE SOFTWARE

En el marco de la productividad de un país, las fábricas tienen un rol importante. No obstante, la manera de medir dicha productividad difiere en cada empresa de acuerdo al rubro en el que se desempeñan. Entre las fábricas de mayor producción económica se encuentra la fábrica de *software*, pues esta no solo toma un rol importante en el rubro de la computación o maquinarias, sino que, como señalan Motta *et al.* (2017), su rol en la actualidad es aplicable a distintas áreas productivas y, por ende, sociales.

En ese sentido, la importancia que ha ido adquiriendo el *software* se debe a la aparición de productos más sofisticados, pues cuanto más inteligentes son los productos, más complejidad en su sistema operativo requieren. Cabe resaltar que las áreas donde mayor aplicación ha tenido son los campos de telefonía e internet; sin embargo, su uso se extiende a distintos sectores e interviene ampliamente en la competitividad de los países.

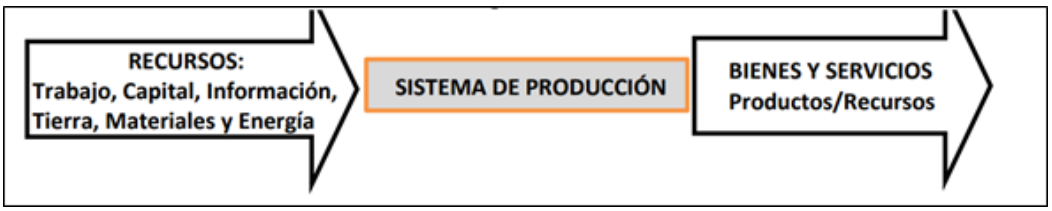
Ahora bien, para determinar los componentes que influyen en la productividad de una fábrica de *software*, es preciso comparar esta última con una empresa de producción regular y adaptarla a las exigencias que se presentan, pues, como señala Carranza (2008), la fábrica de *software* no existe como tal porque no se dedica a obtener un producto propiamente, sino que busca mostrar sus procesos y potenciarlos.

2.1. Productividad en las fábricas de software

La productividad, en términos económicos, se define como "el valor del producto por unidad de insumo" (Loayza, 2016, p. 11), donde intervienen cuatro componentes fundamentales: innovación, educación, eficiencia e infraestructura. La dinámica que se genere entre los cuatro componentes mencionados permitirá mayor o menor productividad. No obstante, es pertinente indicar que la productividad es una variable que puede medirse de distintas maneras y puede estar dirigida a diversos agentes de la producción: capital humano, maquinarias, esfuerzo empleado, etc.

Por otra parte, para Medina (2010, citado en Fontalvo *et al.*, 2017), la productividad es la forma en cómo se utilizan los factores de la producción durante la fabricación de productos o la prestación de servicios, con la finalidad de cubrir las necesidades de la sociedad. Por otro lado, Fontalvo *et al.* (2017) consideran que cuanto menor sea la cantidad de recursos utilizados en una producción y esta última no se ve afectada, mayor productividad existe, pero, en términos generales, hablar de productividad implica relacionar ciertas actividades con la intervención de ciertos elementos, como se visualiza en el esquema de la Figura 2:

Figura 2.
Esquema general de productividad



Nota. Adaptado de Fontalvo *et al.* (2017)

Ahora bien, *fábrica de software* es un término que ha surgido por la necesidad de adoptar las buenas prácticas de las empresas que fabrican bienes a la labor de desarrollar *softwares* más modernos; sin embargo, es necesario aclarar que un *software* no se fabrica.

Dicha expresión se utilizó por primera vez en 1968 en el Congreso International Federation of Information Processing, y a lo largo de los años ha evolucionado y evidenciado cambios en su acepción, así como en sus objetivos (Carranza, 2008). Por ejemplo, en la década de los noventa, las fábricas de *software* se basaban en entornos de desarrollo integrados, cuyo objetivo era mejorar la gestión de calidad del *software*. Posteriormente, en el 2000, las fábricas de *software* eran empresas dirigidas a la producción de componentes, modelos y líneas de productos, cuyo objetivo era desarrollar el sistema *software* con base en sus componentes.

Es visible, entonces, que la acepción de fábrica de *software* se ha modificado con el paso de los años, por esa razón, no se ha registrado aún una definición precisa; sin embargo, según Carranza (2008), hay un punto de coincidencia en todas las posturas: las fábricas de *software* están dirigidas a efectuar procesos con el objetivo de mejorar el desarrollo y la calidad del *software*.

Por otra parte, Cerón y Arboleda (2018) acotan que cada fábrica de *software* debe buscar redefinir todas las dimensiones y los aspectos que componen la realidad del *software* actual. Entre ellas se encuentran las estrategias, las condiciones ambientales, la cultura organizacional y la gestión del conocimiento. Entonces, la productividad en las fábricas de *software* hace referencia a la eficiencia de las empresas en el alcance de sus objetivos. Estos últimos cambiantes en la medida que evoluciona el mundo del *software*. Al respecto, Rubio (2013, citado en Francia, 2019) define a la productividad de software de la siguiente manera: "es la relación del valor funcional del *software* desarrollado y la mano de obra o recursos utilizados para producirlo" (p. 30).

Actualmente, la producción de *software* puede ser evaluada en la vida práctica, en la medida que las empresas presten servicios informáticos (SSI), pues este rubro representa uno de los más dinámicos a nivel internacional por su implicancia económica en sí mismo y porque es un mecanismo de impulso para otras empresas (Motta *et al.* 2017).

En un estudio realizado en Argentina, se recalcan cuatro factores que dieron lugar a un crecimiento considerado en servicios informáticos (SSI), a saber, mano de obra calificada, el aumento de la demanda, el accionar de las empresas locales y la presencia de centros de desarrollo (de origen extranjero); además, el establecimiento de políticas públicas dedicadas al fortalecimiento del sector.

Según Motta *et al.* (2017), el sector software en Argentina tiene, actualmente, un papel relevante en su economía, pues se ha expandido de manera amplia respecto a los demás sectores de producción y ha evidenciado mayores tasas de innovación. Por ejemplo, en el 2010, la productividad en las

fábricas de *software* fue dilatada, pues muchos otros sectores compraron los servicios del *software* como insumo productivo. Algunos de ellos fueron la industria alimentaria, la industria automotriz, la industria de proceso, el sector químico, el sector de bienes de capital y el equipo electrónico, entre otros. El porcentaje de participación de la fábrica de software en cada sector se visualiza en la Tabla 1.

Tabla 1. Destino de la producción de software en Argentina, 2010

Sector destinatario de las ventas	Producción destinada en porcentajes
Sector primario (agricultura, pesca y minería)	9.23
Industria de alimentos, bebidas y tabaco	16.41
Maquinaria y equipo (incluida la electrónica)	6.67
Química	8.21
Automotores y autopartes	10.77
Industrias de proceso (siderurgia, petroquímica)	9.23
Otros sectores (industria en general)	26.67
Servicios de logística y transporte	14.36
Software y servicios informáticos	24.62
Telecomunicaciones	21.54
Comercio	20.00
Servicios financieros	29.23
Servicios de salud	18.46
Otros (servicios en general)	44.62
Consumidores finales	11.79
Administración pública	30.26

Nota. Adaptado de Motta *et al.* (2017)

El crecimiento de la productividad del *software* fue tan destacable que, en el 2012, el valor bruto de su producción era el triple que el de la producción minera y representaba casi la mitad del valor bruto del sector siderúrgico.

En adición, Aravena *et al.* (2012) señalan que Brasil es otro de los países que han mostrado un aumento en la productividad de *software* y, en general, en las tecnologías de la información y la comunicación (TIC), lo cual ha tenido una repercusión en el aumento de su PBI entre 1990 y 2008. Incluso, a finales del 2008 registró una inversión en TIC mayor que la de países desarrollados como Estados Unidos, Corea y España.

En el caso del Perú, Tello (2017) sostiene que en los últimos años no se ha observado el interés suficiente por parte de las autoridades para incentivar y hacer crecer el sector ciencia y tecnología; además, señala que no se ha establecido una articulación sólida del Sistema Nacional de Ciencia y Tecnología e Innovación Tecnológica (SINACYT). Asimismo, afirma que fortalecer las políticas de Computer Telephony Integration (CIT) podría mejorar la productividad en general y argumenta que un factor clave para elevar la atención y la productividad en las CIT es el tamaño de las empresas nacionales, pues cuanto más "grandes" o consolidadas estén, mayor necesidad de los servicios tecnológicos van a requerir, además, habrá más posibilidad de pagar el costo de los servicios de las empresas tecnológicas.

2.2. Categorización de factores influyentes en la productividad

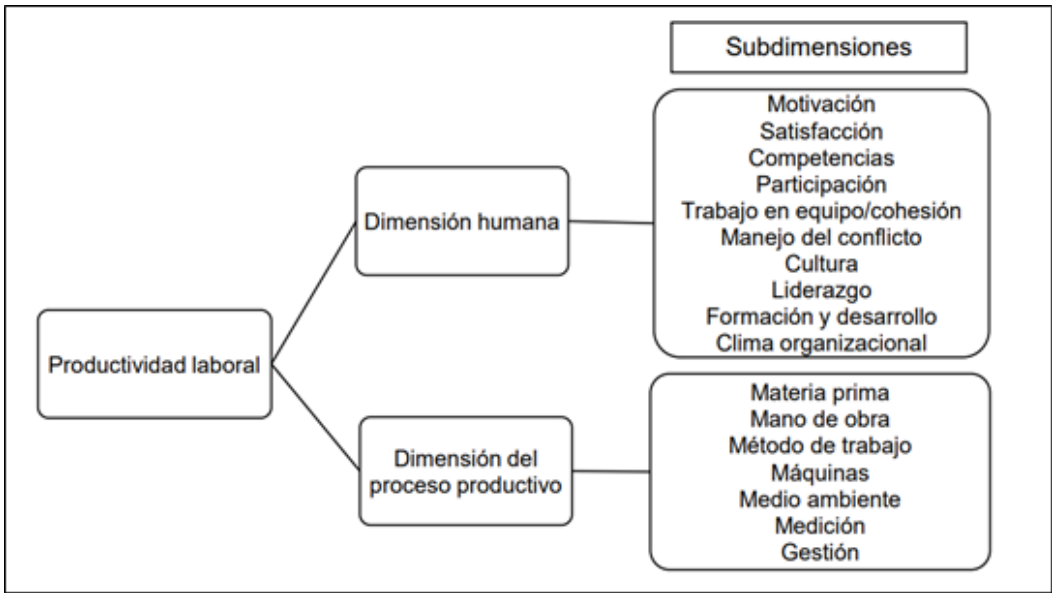
2.2.1. Producto

El factor producto influye en la productividad, debido a que es un elemento destinado a satisfacer determinadas necesidades de un comprador, y este último es el que se encarga de evaluar si el producto cumple con la satisfacción requerida; con base en ello, designa un valor, que se refleja en un precio. Si el producto no cumple con la calidad esperada, la venta del mismo será menor y, en consecuencia, la productividad menguará. Por esa razón, Ramírez (2011) asegura que las empresas buscan fabricar un producto (y las partes que lo conforman) de gran eficiencia para poder brindar a los demandantes la satisfacción esperada y, por ende, conseguir un mayor porcentaje en ventas (citado en Fontalvo *et al.*, 2017).

2.2.2. Proceso

Jaimes *et al.* (2018), en un estudio de la productividad en Colombia, concluyeron que el proceso dentro de la productividad laboral representa una dimensión muy relevante, en la cual están incluidos los siguientes ítems: materia prima, mano de obra, método de trabajo, máquinas, medioambiente, medición y gestión, como se observa en Figura 3.

Figura 3.
Factores determinantes de la productividad laboral



Nota. Adaptado de Jaimes *et al.* (2018)

La materia prima es un elemento que forma parte del proceso productivo y debe evaluarse la funcionalidad, la disponibilidad y la calidad de la misma para garantizar un proceso adecuado. Asimismo, la mano de obra es otro elemento que debe ser considerado y comprende la observación de las habilidades, incentivos y capacitaciones de los trabajadores.

Por otra parte, el elemento método de trabajo es aquel donde se analiza la distribución de las operaciones y la estandarización del proceso, con la finalidad de mejorar la productividad de la empresa. En adición, las máquinas, debido a que en la actualidad son elementos incorporados en casi todo el rubro empresarial, deben ser evaluadas constantemente; los puntos más relevantes a considerar son el tiempo de duración, su mantenimiento y la adquisición de las mismas.

El elemento medioambiente está relacionado con el espacio donde se ejecuta la producción, se valora la iluminación, el orden, la limpieza y la seguridad. Además, la medición hace referencia al hecho de medir la producción, registrar los retrasos y controlar la calidad. Por último, el elemento gestión se relaciona con la labor o dirección de las autoridades.

2.2.3. Entorno de desarrollo

El entorno de desarrollo es un factor altamente influyente en la productividad, pues es el campo donde se debe promover una constante búsqueda de nuevas técnicas, metodologías e innovadoras formas de aplicar pruebas a las nuevas propuestas. En ese sentido, la prioridad que se brinde al desarrollo de este factor puede ser un diferenciador respecto a los resultados de otras empresas. También se le considera al entorno de desarrollo a aquellos recursos que permiten a los programadores crear aplicaciones o *softwares*. Por ejemplo, en el ámbito de la ingeniería de *software*, hay un enfoque llamado desarrollo de software continuo (DSC) en el que se producen distintas versiones de *software* en tiempos reducidos; esto permite probar nuevas técnicas y metodologías: despliegues continuos, integración continua (Mascheroni et al., 2016).

2.2.4. Cultura corporativa

Según García y Flores (2017), la cultura corporativa hace alusión al conjunto de valores, actitudes, creencias, etc. que concibe un conjunto de trabajadores respecto a la empresa donde pertenece. En otras palabras se le puede considerar la fuerza vital de la empresa, debido a que puede influir positivamente en la eficacia con la que se desenvuelven los trabajadores, incluso puede motivar una actitud proactiva en ellos.

Por lo tanto, la cultura corporativa es un factor que se forma a partir de la participación integral de una empresa: los líderes, los fundadores y los empleados en general. En ella se establece una serie de clasificaciones que se comparte con todos los integrantes: valores, ideas de moralidad, ritos, símbolos de carga afectiva, etc. Ahora bien, para que estas clasificaciones sean asumidas por los trabajadores, es importante establecer actos de poder como recompensas o "castigos".

En ese sentido, el aspecto cultural en una empresa es uno de los más influyentes en el rendimiento laboral, pues genera un ambiente de bienestar y comodidad. Al respecto, Madero y Barboza (2015) afirman que "una empresa que emplea prácticas organizacionales flexibles, y enfocadas a mejorar el ambiente laboral mantendrá sus empleados satisfechos, y consecuentemente logrará posicionarse dentro de su industria con un mayor nivel de competitividad sostenible" (p. 753). Además, García y Flores (2017) mencionan que la cultura corporativa cumple ciertas funciones básicas dentro de una empresa:

- Apoya en la construcción de una identidad corporativa.
- Fomenta la cohesión general entre los integrantes de la empresa.
- Crea en el trabajador una relación entre su éxito personal y el empresarial.

2.2.5. Cultura del equipo

Este factor hace referencia al conjunto de concepciones que orienta a los trabajadores a desempeñarse en equipo. Para Richardson y West (2010, citados en Acosta *et al.*, 2019), t*rabajo en equipo* se entiende como "personas con objetivos comunes y con interdependencia de tareas" (p. 88) y comprende un factor medular porque es una práctica muy impulsada por las empresas actualmente debido a que acelera la gestión laboral (Fernández, 2015).

En ese sentido, se busca que los trabajadores muestren un desempeño óptimo y estén comprometidos con la empresa a la que brindan sus servicios, de manera que las labores de los demás impulsen un avance en las tareas y no un retroceso. Para lograr ello, Acosta *et al.* (2019) sostienen que debe estar presente un elemento fundamental: la confianza. Esta va a ser la intermediaria entre el trabajo en equipo y el logro eficaz de las metas planteadas.

2.2.6. Capacidades y experiencia

El factor capacidades y experiencia está relacionado con las características necesarias en los recursos humanos (personal de trabajo). Según Reina (2016), estas características se ven proyectadas en la empresa, pues permiten desarrollar productos más innovadores y, a la vez, mejora la productividad laboral.

Al respecto, Galindo y Haskel (2005) sostienen que existe evidencia empírica para afirmar que las empresas más productivas son aquellas que cuentan con trabajadores más cualificados. En ese sentido, es importante que las empresas promuevan constantemente políticas de formación laboral, con el fin de lograr una mayor productividad empresarial, pues no basta con tomar mano calificada del mercado, sino también formarla (Guisado *et al.*, 2016).

2.2.7. Proyecto

Este factor hace referencia a las consideraciones tomadas en cuenta en la elaboración de la estructura organizacional de la empresa, donde se debe especificar los distintos procesos de gestión y las limitaciones de desarrollo. En ese sentido, Santamaría (2016) argumenta que el diseño de un proyecto debe ajustarse al ambiente donde se ejecuta y estar elaborado por profesionales de las diferentes áreas. Además, señala que el proyecto debe tomar en cuenta cada etapa de la producción, desde la elección de materias primas hasta su finalización (como producto o servicio).

CAPÍTULO III

MODELOS DE MEDICIÓN EN LA PRODUCTIVIDAD DE LAS FÁBRICAS DE SOFTWARE

La productividad se puede entender, a grandes rasgos, como la forma de utilización de los factores propiamente de producción en cuanto a la generación de bienes y servicios, para el logro de objetivos, casi siempre con la finalidad de satisfacer tanto demandas como necesidades. Por lo tanto, la productividad como concepto tiene un objetivo claro, así como preciso, el cual consiste precisamente en elevar los niveles de competitividad fundamentales y necesarios en una sociedad cada día más globalizada.

Muchos han sido los sistemas, métodos y maneras de medir la productividad dentro de una organización; sin embargo, queda claro que, dentro del universo tecnológico, los sistemas operativos como los *softwares* son los más efectivos, así como los que logran mejores resultados, además de medirlos con eficiencia, con la finalidad de satisfacer necesidades de los consumidores o clientes.

3.1. Análisis envoltorio de datos (DEA)

El análisis o "metodología DEA es una herramienta que emplea la programación lineal para establecer la eficiencia entre unidades homogéneas de decisión" (Iribarren, 2020, p. 18). Este análisis envoltorio de datos, tal y como se le conoce, es ampliamente conocido; se trata de una técnica de programación matemática ampliamente empleada por los teóricos Charnes, Cooper y Rhodes a finales de la década de los setenta, que luego fue extendida por Banker, Charnes y Cooper a principios de los ochenta, quienes la incluyeron para medir los retornos de escala variable. Esta técnica generaliza la medida de eficiencia técnica de Farrell a finales de los cincuenta, con el fin de evaluar no solo la eficiencia relativa de unidades pares, sino también las medidas múltiples de rendimiento.

De acuerdo a Charnes *et al.* (1978, citado en Nolasco *et al.*, 2020), la DEA "es un modelo de retorno de escala constante que solo medía la eficiencia técnica. Básicamente, las primeras aplicaciones de la metodología midieron la eficiencia relativa entre unidades del mismo sector, específicamente instituciones públicas y educativas" (p. 81).

Asimismo, De los Reyes (2020) menciona que DEA analiza las unidades de producción, qué se debe hacer para mejorar su eficiencia y cuáles son aquellas mejoras que se pueden aplicar a estas para dicho objetivo.

> En la actualidad esta metodología se usa para valorar las fusiones de entidades desde la perspectiva de la eficiencia. La valoración que la metodología DEA puede realizar de las fusiones de empresas se divide entre la evaluación de las ganancias de eficiencia que una fusión determinada y conocida de ante mano puede aportar y determinar qué fusiones de entre un conjunto de DMUs son la que más beneficios aportan desde el punto de vista de la eficiencia (p. 1).

En ese sentido, el DEA emerge como una técnica para evaluar la eficiencia de una serie de elementos denominados, por lo general, unidades de toma de decisiones. En adelante, a este se le denominó DMU (*Decision Making Unit*) y se empleó tanto para entradas como salidas. La DEA contempla una serie de metodologías que a continuación se detallan.

Como primera metodología se tiene la identificación de un conjunto de posibilidades de producción de problemas, esto se traduce en los siguientes aspectos. La definición de los posibles puntos de operación, los cuales están representados en dos alternativas: la primera, retorno de escala constante (*constant return to scale*); la segunda, retorno de escala variable (*variable return to scale*).

Como segundo paso está la selección del modelo DEA, el cual debe ser el más adecuado al problema por resolver, no obstante, existe una gran cantidad de modelos DEA, todos ellos tanto con fines como objetivos similares, lo cual consiste precisamente en hallar un punto admisible de mayor productividad. Esta técnica si bien ha sido empleada para medir los efectos de productividad es importante saber que contempla una serie de ventajas que a continuación se presentan.

En primer lugar, se dice que esta técnica no requiere la imposición de una función específica de producción que relacione tanto las variables independientes con las dependientes, sino que asigna los valores de forma ponderados tanto con las entradas como las salidas. Otra de las ventajas es que esta técnica realiza una serie de comparaciones en simultáneo de procesos con múltiples variables tanto de entrada como de salida, de manera que genera una suerte de eficiencia para cada organización, institución o empresa.

Por otro lado, esta técnica tiene como ventaja calcular la cantidad de recursos que se pueden ahorrar, de igual manera, la cantidad sobrante para aquellas empresas o instituciones que no son eficientes. Además, se puede emplear para determinar la eficiencia técnica o económica si la información es proporcionada. No obstante, el DEA presenta, *grosso modo*, ciertas y hasta eventuales limitantes, como las que se mencionan a continuación. Esta técnica no presenta indicadores que midan errores o fallas en el sistema, por lo que no es una técnica apropiada para comprobar hipótesis.

Un aspecto que debe ponerse en relieve es que este sistema forma parte de un proceso netamente de producción, por lo que se entiende que este mide los niveles que son inherentes a los de una organización, empresa o institución.

3.2. Conceptos clave del sistema DEA

A continuación, se muestran los conceptos clave del sistema DEA, los cuales son los siguientes:

- Unidad productiva

La unidad productiva se define, a grandes rasgos, como aquella organización que produce ciertos recursos, con la capacidad de poder modificar, por un lado, aquellos recursos que se consumen, mejor conocidos como de entrada; por otro, los de salida.

- Productividad

Esta se define como la relación entre los resultados y los recursos que se emplean en la producción. Además, es una de las formas clásicas conocidas y busca de qué manera se están aprovechando los recursos. Un caso, por ejemplo, se relaciona con la entrada y salida.

Dentro del universo de la productividad es importante poner en relieve que una de las grandes dificultades encontradas es determinar si existen factores relevantes al momento de definir tanto las entradas como las salidas de la unidad productiva. En algunos casos, los productos o recursos no son fácilmente identificables, incluso a veces no es nada fácil obtener una información en el sentido

estricto. En el caso de existir varias entradas y salidas, la expresión matemática es la siguiente:

$$Productividad = \frac{Suma\ ponderadas\ de\ salidas}{Suma\ ponderadas\ de\ entradas}$$

Por lo tanto, si se denota como x_{ij} a la cantidad de entrada o recurso i utilizado por la unidad j, y como y_{kj} a la cantidad de salida o resultado k que produce la misma unidad, se obtiene la expresión:

$$Entrada\ virtual_j = \sum_{i=1}^{m} u_{ij} x_{ij}$$

$$Salida\ virtual_j = \sum_{k=1}^{s} v_{kj} y_{kj}$$

En donde los términos u_{ij} y v_{kj} son, respectivamente, los pesos correspondientes a cada entrada y salida, que adimensionalizan las expresiones de entrada y salida virtuales; m, el número total de entradas consideradas, y s, el número de salidas de la unidad. Con estos nuevos conceptos se puede definir la productividad de la siguiente manera:

$$Productividad_j = \frac{\sum_{k=1}^{s} v_{kj} y_{kj}}{\sum_{i=1}^{m} u_{ij} x_{ij}}$$

- Eficiencia relativa

La eficiencia relativa se define de la siguiente manera:

$$Eficiencia_j = \frac{Productividad_j}{Productividad_{max}} = \frac{Salida\ virtual_j / Entrada\ virtual_j}{Salida\ virtual_{max} / Entrada\ virtual_{max}}$$

Donde el subíndice j indica la unidad que se va a estudiar, y el subíndice max, la unidad de máxima productividad. Se pueden distinguir varios tipos de eficiencias relativas en función de la unidad de referencia que se utilice.

Se habla de la eficiencia global cuando se escoge como unidad de referencia a la de mayor productividad entre las que están en estudio. Por otro lado, la eficiencia técnica se utiliza cuando se elige como unidad de referencia la de mayor productividad de entre las unidades de su tamaño. Por lo tanto, se define la eficiencia de escala como el cociente entre la eficiencia global y la eficiencia técnica.

3.3. Modelo CMMI y Six Sigma

El modelo CMMI es ciertamente un modelo que tiene como propósito reconocer qué prácticas se necesitan para aplicar el mejor modelo en cuanto al rendimiento y la productividad en una organización, empresa o institución. Por otro lado, al Six Sigma se le considera un método que establece de qué manera aplicar prácticas de un modo particular o específico.

Es bien sabido que el CMMI tiene un objetivo claro, además de definido, puesto que busca mejorar de forma continua tanto la calidad de los procesos y los productos de una determinada organización como ser una guía estableciendo ciertos niveles de madurez, lo cual permite predecir la capacidad para la obtención de resultados, metas y objetivos de una empresa.

Dentro de la gama de organizaciones, por ejemplo, las que tienen un nivel alto suelen experimentar destacadas variaciones en cuanto a costos, pero también los mismos cronogramas de entrega, sin mencionar que estos pueden incidir en la calidad de los productos o servicios que suelen ofrecer. Por lo tanto, a medida que la madurez crece, los resultados son aún mayores.

Por otro lado, en cuanto al sistema Six Sigma, este se basa en una política de mejora permanente a través de la medición de los servicios que presta a la organización. Por lo general, tanto el primero como el segundo suelen presentar una fuerte sinergia. Sigma es una letra del alfabeto griego y se emplea, principalmente, para representar la desviación; es decir, la unidad estadística de medición, tal y como se le conoce, también representa la misma variabilidad o dispersión de una serie de datos.

En ese sentido, el Six Sigma es una metodología que busca mejorar los procesos. Esta fue creada con la finalidad de reducir la variabilidad de los fallos o defectos que solían presentarse. Su registro data desde la década de los setenta cuando una empresa nipona se hizo cargo de una de las fábricas de Motorola que manufacturaba televisores en los Estados Unidos y de forma rápida comenzaron a hacer cambios drásticos en la manera en que se operaba (Pyzdek, 2003, citado en Pérez, 2016), de igual manera, es una forma de mejorar una empresa o una organización.

Esencialmente, investiga y se encarga de encontrar, así como eliminar las causas de errores o defectos en los procesos, enfocándose en las salidas que sean críticas para los clientes (Vazquez, 2019). El objetivo de esta metodología era precisamente alcanzar una meta que consistía en detectar las fallas. Para comprender qué es un Six Sigma conviene, en primer lugar, saber qué es una variación, así también, cómo se mide esta variación con el fin de ser empleada en organismos, empresas o instituciones.

3.4. Modelo de lógica difusa

Este modelo fue dado a conocer a mediados de los años sesenta por el ingeniero Lotfy A. Zadeh. No obstante, esta teoría ya había sido practicada hace unos cuantos siglos por las sociedades griegas. Para algunos filósofos, entre los más destacados como Aristóteles, quien consideraba que existían ciertos grados no solo de veracidad en los objetos, sino también que estos comportaban ciertos y eventuales grados de falsedad. En ese sentido, según Díaz-Contreras *et al.* (2014), "la lógica difusa es un formalismo matemático que pretende emular la habilidad que tienen algunas personas para tomar decisiones correctas a partir de datos vagos o imprecisos y que están expresados lingüísticamente" (p. 549).

El aspecto de esta teoría estriba, por lo tanto, en los sistemas basados esencialmente en la teoría de la lógica difusa que, a diferencia de la lógica clásica o tradicional, tiene la capacidad de reproducir los modos usuales del razonamiento, así como del conocimiento mismo. En términos más claros, y para dar un ejemplo más concreto, si la lógica es la ciencia de los principios, como formales; la lógica difusa, por su parte, se refiere a principios formales de un razonamiento aproximado y da a entender que la lógica difusa se caracteriza por su flexibilidad, también por su alto nivel de tolerancia en cuanto a los propios principios de imprecisión.

3.5. Modelo de reutilización del software

En los últimos años, las exigencias en cuanto a la calidad de los productos se han ido acentuando día a día, por lo que las medidas que suelen emplearse han sido cruciales y determinantes. De allí la importancia de la reutilización de los sistemas que consiste, precisamente, en crear sistemas a partir

de los ya existentes, por ello, se ha hecho imperativa la necesidad de contar con procesos de reúso, por un lado, para ofrecer al cliente un servicio de calidad con la finalidad de lograr estándares de calidad, y por el otro, alcanzar futuras evaluaciones de corte global (Tamayo *et al.* 2019).

Tradicionalmente, el desarrollo de un *software* se ha dividido en fases para comprender su funcionamiento, así como su desarrollo y, por supuesto, las técnicas que bien pudieran servir en su aplicación. Asimismo, para manejar, si lo hubiera, alguna dificultad y, de esta manera, dominar mejor la situación. Como se sabe, el desarrollo de un *software* es un proceso consistente que mantiene una estrecha relación con sus pasos, casi siempre estos mismos parten de ciertas especificaciones más o menos de naturaleza formal.

En ese sentido, cada paso es un proceso de transformación, el cual se divide en dos grupos: métodos generativos y métodos composicionales. El primero es un tipo de modelo que se concentra, precisamente, en reusar o reutilizar dichas transformaciones de descripciones de problemas. En el caso de los métodos composicionales, su principal objetivo es reutilizar las descripciones de problemas como, por ejemplo: bloques de construcción que se usarán en el desarrollo del *software*.

3.6. Modelo SEM

Este modelo data de principios del siglo XX cuando el científico Sewall Wright desarrolló por primera vez un modelo "de trayectorias para describir la contribución genética entre compañeros de camada" (Manzano, 2017, p. 68). Asimismo, es considerado, debido a sus características, como una "herramienta estadística multivariada" (Manzano, 2017, p. 68), igualmente, es conocido como un análisis de covarianzas. Estos modelos, tal y como lo ha señalado Manzano (2017), permiten probar qué relación existe entre las variables observadas y aquellas cuya característica es latente.

Pero ¿qué es una variable observada? Para definirla se debe tener en cuenta que esta hace posible una medición directa, por ejemplo, la edad o la estatura. Cuando se refiere a una variable latente, por lo general, esta mide aspectos como la inteligencia, la motivación, la depresión o el estrés. Para Martínez y Fierro (2018) "la teoría puede definirse como un conjunto sistemático de relaciones que da una explicación exhaustiva de un fenómeno y permite al investigador distinguir qué variables predicen cada variable dependiente" (p. 6).

3.7. Modelo factor total de productividad

Antes de definir un modelo factor total de productividad, es menester saber qué se entiende por productividad. Esta hace referencia a la "relación existente entre el volumen total de producción y los recursos utilizados para alcanzar dicho nivel de producción, es decir la razón entre las salidas y las entradas" (Medina, 2010, citado en Fontalvo *et al.*, 2017, p. 50); por lo que se entiende como la manera en que se logran emplear los factores de producción durante los procesos de elaboración no solo de un determinado producto, sino de un servicio específico.

También se entiende productividad como un elemento clave, además de estratégico dentro de las organizaciones, ya que los servicios y los productos que se elaboran o se ofrecen no pueden ser competitivos sin antes haber cumplido con cierto rigor o haber reunido los estándares elevados de competitividad. En general, cuando se habla de productividad se encuentran aspectos relacionados casi siempre con elementos y actividades para la obtención de resultados, con el fin de mejorar tanto los productos como los servicios que ofrece.

Por otro lado, la productividad es entendida como el resultado de una suerte de articulación que se da cuando entran en armonía aspectos relacionados con la tecnología, la organización y el talento humano, con el objetivo de lograr los objetivos planificados (Cequea y Rodríguez-Monroy, 2012).

En ese sentido, ¿cuándo un modelo es productivo?, ¿qué características debe tener un modelo para ser catalogado no solo como modelo, sino como productivo? Un modelo para ser factor clave de productividad debe tener y reunir ciertas características, que se traduzcan en eficiencia, además de tener un destacado sistema operativo que brinde respuestas, haga posible los fines y los objetivos que se tengan. En el campo de la tecnología, un factor se convierte en modelo de productividad cuando este logra los objetivos, así como lo exigido por los clientes.

3.8. Modelo ISO 9126 y IEEE Std. 1045

Un modelo se define, a grandes rasgos, como "una representación de calidad a partir de criterios o características" (Fernández, 2018, p. 13), la cual permite satisfacer, por un lado, las necesidades de los desarrolladores, así también de aquellos que se encargan de su mantenimiento, y por otro, a quienes lo adquieren y sus usuarios finales. Además, la calidad mide qué tanto un producto o un servicio llega a reunir los estándares de calidad, asimismo, evalúa tanto los productos como los servicios. Para que un modelo de calidad tenga éxito es menester que este sea simple y concreto (Fernández, 2018).

Los modelos de calidad de los productos, señala Fernández (2018), son los siguientes: McCall, Boehm, FURPS, Dromey, y los ISO 9126 e ISO 25010. Cada uno de estos define los estándares de calidad con la finalidad de cubrir no solo las demandas exigidas, sino los objetivos y las metas de las organizaciones. Estos modelos presentan características genéricas, por lo que su aplicación puede hacerse en cualquier tipo de *software*. En cuanto a la norma ISO 9126, esta se define como:

> Un marco de representación de los modelos de calidad, siguiendo una estructura jerárquica basada en los modelos anteriores. La calidad se determina mediante la especificación de características generales del software que progresivamente se descomponen en subcaracterísticas, compuestas a su vez de atributos, de manera que en la base de la jerarquía estarán atributos del software evaluables mediante algún tipo de métrica. (Fernández, 2018, p. 17)

Por otra parte, el IEEE Std. 1045 prescribe medidas para el logro de objetivos clave, así como de productividad en el software. Un aspecto resaltante en este es que no mide la calidad del *software*, por lo tanto, su objetivo se centra en solo medir y no calificar en términos de productividad. Otro de los objetivos resaltantes es una mejor comprensión de los procesos de *software*, lo que permite que el sistema mejore en la medida en que este va abarcando los procesos tanto de información como de medición.

CAPÍTULO IV

MODELO PARA LA MEDICIÓN DE LA PRODUCTIVIDAD DE LA FÁBRICA DE SOFTWARE

En algunos estudios realizados sobre la productividad, y de acuerdo a lo indicado por Bonilla (2012), es muy común decir "hay que ser competitivos", pero no se destaca la importancia de la productividad como componente fundamental de la competitividad. Según Echeverry y Hernández (2005), la mayoría de los estudios la han analizado de manera agregada y de estos no se pueden tomar recomendaciones para cada sector, ya que presentan una dinámica diferente.

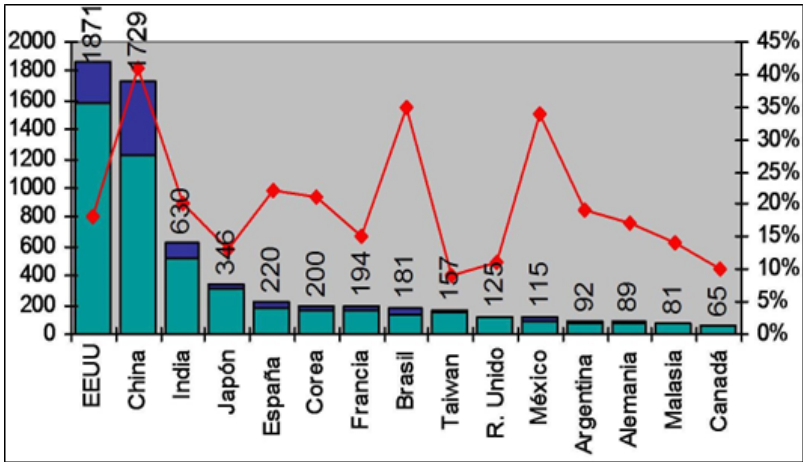
En 1997, E. Prescott, Premio Nobel de Economía en el 2004, enfatiza la importancia de la productividad como fuente de crecimiento, de esta manera, destaca que su estudio es insuficiente. Por otra parte, según Reyes (2010), el comportamiento de esta variable explica el 80 % del crecimiento económico en los países desarrollados y cerca del 40 % en los del Tercer Mundo:

> Paul Krugman (2007) [...] señala que para lograr competitividad en los mercados internacionales se requiere fortalecer las economías domésticas y, por tanto, tener en cuenta la productividad sujeta tanto al capital humano como a los recursos tecnológicos de que se disponga [...] Michael Porter (2005) afirma que "la productividad es, a largo plazo, el determinante primordial del nivel de vida de un país y del ingreso nacional por habitante" Asimismo, Romo y Andel (2005) asocian la productividad y la competitividad a "la capacidad de crear un entorno que favorezca el crecimiento sostenido de la productividad y que se refleje en niveles de vida más elevados de la población. Esto incorpora niveles macro, meso y microeconómicos en un marco de integración de la economía global". (Bonilla, 2012, pp. 2-3)

Por otro lado, Agosin *et al.* (2010) plantean que el bajo crecimiento de la productividad es la raíz del deficiente crecimiento económico de América Latina. En adición, el crecimiento de la industria del *software* ha conllevado a que las organizaciones se tornen cada vez más competitivas y agresivas en la búsqueda de la medición de la productividad y la eficiencia en ambientes de desarrollo de *software*. De ese modo, se pone énfasis en aumentar su eficiencia, reducir costos y, sobre todo, estimar el tiempo y recursos que deberán asignar a sus proyectos.

En ese sentido, la medición de la productividad ha sido siempre una preocupación en las organizaciones, por lo que, la industria del *software* no es ajena a esta realidad. En la actualidad, pocos dudan de la importancia de la medición si se quiere mejorar la calidad y la productividad en el proceso de desarrollo del *software*. Así pues, la necesidad y la motivación por medir se han incrementado notablemente con la preocupación de las organizaciones por alcanzar mayores niveles de madurez y las consiguientes certificaciones basadas en modelos y normas como ISO 9000, ISO 15504 o CMMI (Figura 4).

Figura 4.
Empresas acreditadas por países



Nota. Adaptado de Capability Maturity Model Integration [CMMI] (2018).

4.1. Metodología

Contar con un modelo de medición de la productividad permitirá una gestión eficaz y eficiente de los recursos, de manera que se logrará: a) identificar la fábrica de *software*, los proyectos más eficientes, así como las buenas prácticas y lecciones aprendidas en los proyectos que conllevan a un mejor desempeño en la ejecución de sus actividades; b) diseñar estrategias para la mejora de la productividad en los proyectos de desarrollo de *software*, y c) realizar un *benchmarking* entre las fábricas de *software* y los proyectos implementados.

Por tal motivo, en este libro, se plantea como propuesta un modelo que permita la medición de la productividad de la fábrica de *software* para la efectividad en el uso de los recursos productivos para el cumplimiento de los objetivos organizacionales. A través de este modelo, se busca diagnosticar el estado del arte sobre los factores que influyen en la productividad de la fábrica de *software*.

Asimismo, se desea describir cómo los factores de productividad afectan a las unidades de trabajo de análisis y diseño de las fábricas de *software*. Al utilizar este modelo, se busca establecer cómo ciertos aspectos afectan a algunas áreas en la producción de *software* en las fábricas de estos; cómo el uso de reglas, eventos y la comunicación "cara a cara" afectan al área de análisis y diseño; cómo la heterogeneidad y el tamaño del equipo afectan a las áreas de análisis y diseño, y programación, respectivamente, y cómo la comunicación osmótica afecta al área de programación.

Además, este modelo va a permitir establecer cómo la confianza en el equipo afecta a las áreas de análisis y diseño, programación, pruebas, respectivamente, en la producción de *software* en las fábricas de estos. Ello permitirá diseñar y validar el modelo para la medición de la productividad de las fábricas de *software* y sus respectivos proyectos. Por otro lado, confirmará si el factor confianza en el equipo influye en las unidades de trabajo de análisis y diseño, programación, pruebas, respectivamente.

Cabe señalar que este estudio cae dentro de la tipificación de investigación tecnológica, dado que responde a problemas técnicos y se aprovecha del conocimiento teórico-científico producto de la investigación básica. Asimismo, organiza reglas técnicas, cuya aplicación posibilita cambios en la realidad (Sánchez y Reyes, 2015). En ese sentido, tendría como finalidad solucionar problemas o situaciones que el conocimiento científico consolidado como tecnología demanda; por lo tanto, no sería su finalidad descubrir nuevas leyes y casualidades, sino reconstruir procesos en función de

descubrimientos ya realizados (Bello, 2013).

Por ello, esta investigación se define como tecnológica debido a que se hará la propuesta de un modelo a partir del cual se pueda evaluar la productividad de las empresas que se dedican a la producción de *software*, además de cuantificar el logro de los objetivos trazados en virtud de que los modelos clásicos que permiten la medición de la productividad dentro de una empresa no aplican para este tipo de organizaciones.

Para ese propósito, la investigación se dividió en dos momentos. El primero fue la revisión sistemática, la cual, según Kitchenham y Charters (2007), ha sido adaptada e involucra las siguientes fases: 1) planificación de la revisión, etapa en la que se plantea las preguntas de investigación y se define el protocolo de revisión; 2) conducción de la revisión, donde se ejecuta el plan y se seleccionan los estudios primarios de acuerdo a los criterios de inclusión y exclusión establecidos para el efecto; 3) reporte de la revisión, etapa en la cual se presentan los resultados estadísticos y el análisis de los documentos encontrados y seleccionados, los cuales son presentados posteriormente.

Con esta revisión se busca identificar los factores que influyen en la productividad de la fábrica de *software*. Para este fin, se revisó el concepto de *fábrica de software* y *productividad* conforme a las fases anteriormente comentadas, lo que permitió, a la vez, identificar aspectos a considerar en la medición de la productividad: factores que influyen y modelos para medir la productividad.

El segundo momento lo conformó la aplicación del cuestionario anteriormente descrito y a partir del cual se determinó la percepción de los diferentes profesionales y expertos que interactúan en la fábrica de *software* en relación con los factores que influyen en la productividad. Con el mismo cuestionario, se recolectaron los datos a partir de los cuales se hizo las pruebas de hipótesis.

Para esta investigación se utilizó la técnica de la encuesta, y como instrumento, un cuestionario, el cual fue estructurado en tres secciones. La primera, información general, consiste de cinco preguntas y está relacionada con la caracterización de la empresa y las personas que llenan el cuestionario.

La segunda sección, factores de productividad de la fábrica de *software*, corresponde a preguntas que determinan la influencia de los factores que afectan la productividad en las diferentes unidades de trabajo de la fábrica de *software*. Esta ha sido establecida con base en una escala de Likert de cinco niveles de respuesta: 1 = No tiene influencia, 2 = Baja influencia, 3 = Mediana influencia, 4 = Alta influencia y 5 = Influye totalmente.

Por último, la tercera sección, otros aspectos, tiene como finalidad obtener un mayor entendimiento del impacto de los factores en la productividad de las unidades de trabajo de la fábrica de *software*.

El cuestionario fue aplicado en el período junio-octubre de 2017 a diferentes profesionales y expertos que participan en la supervisión y monitoreo en la fábrica de *software*. Este estudio ha sido desarrollado en organizaciones que han implementado modelos de fábricas de software en el Perú, que tienen un rol importante en la industria del *software* y operan en América del Sur.

El objetivo con la aplicación del instrumento fue determinar la percepción de los diferentes profesionales y expertos que interactúan en la fábrica de *software* en relación con los factores que influyen en la productividad. Para ello se enviaron cartas y correos electrónicos invitando a la participación en las encuestas a la Asociación Peruana de Productores de *Software* (APESOFT),

la Cámara de Comercio de Lima (CCL), la Advent International (GMD S. A.), las instituciones públicas y privadas que tienen un área de TI, los estudiantes de posgrado de la Universidad Nacional Mayor de San Marcos que tienen roles de supervisor en áreas de TI, entre otros. Se obtuvieron 160 encuestas de las cuales se descartaron 10 por tener incongruencias en las respuestas.

En lo que respecta a la validez, se realizó una prueba piloto para validar las preguntas, donde participaron gerentes y jefes de proyectos sénior, de esta manera, se revisó si las preguntas guardaban relación con las hipótesis planteadas, a partir de ello, se corrigió la redacción de las preguntas y el lenguaje utilizado. (La versión definitiva del cuestionario se encuentra disponible en http://tinyurl.com/y5kbpk7p).

Para la confiabilidad, Celina y Campo (2005) indican que el valor aceptable para el coeficiente alfa de Cronbach es 0.7. Un valor superior a este revela una fuerte relación entre las preguntas, con lo cual la validez del instrumento es aceptable. Según los resultados obtenidos con la herramienta SPSS, se obtuvo un valor del alfa de Cronbach igual a 0.783, que comprueba que la validez del instrumento es aceptable.

Se obtuvo la participación de 150 personas, por lo que es necesario considerar cuál es el porcentaje de error muestral, de tal manera que se pueda establecer la validez del presente estudio. Al no contar con información histórica de estudios similares ni saber con exactitud cuántas empresas de fábricas de *software* hay en el Perú, se halló el error muestral en función de la varianza máxima posible, la cual asegura la validez (representatividad) de la muestra tomada. Para el cálculo del nivel de error, se aplicó la siguiente fórmula de tamaño de muestra:

$$n = \frac{z^2 pq}{\varepsilon^2}$$

Al despejar ε, se obtuvo lo siguiente:

$$\varepsilon = \sqrt{\frac{z^2 pq}{n}}$$

Donde:
n = 150 (tamaño de muestra),
z = 1.96 (95 % de nivel de confianza),
p = 0.5 (varianza máxima)
q = (1 − 0.5) = 0.5

$$\varepsilon = \sqrt{\frac{1.96^2 0.5 * 0.5}{150}} = 0.08$$

En otras palabras, el resultado fue un 8 % de error muestral.

Por lo tanto, se puede asegurar que para un nivel de confianza del 95 %, un p de varianza máxima igual que 0.5, un q = 1 − p = 0.5 y un error muestral del 8 %, se tomó una muestra de 150 personas (n = 150) de una población de la cual se desconoce su tamaño exacto, pero se sabe que es mayor que 10 000 (N > 10 000).

4.2. Análisis de los resultados

El instrumento para el análisis de los datos fue el cuestionario, luego, se realizaron los siguientes análisis estadísticos a los datos obtenidos: 1) análisis descriptivo de la muestra (análisis de aspectos demográficos de la encuesta: tamaño de empresa, sector del negocio y años de experiencia del encuestado, media, varianza, moda y la distribución de las respuestas de los encuestados), 2) fiabilidad de datos (prueba de fiabilidad usando el alfa de Cronbach), cuyo valor manifiesta la consistencia interna, es decir, muestra la correlación entre cada una de las preguntas, 3) regresión lineal múltiple para la comprobación de las hipótesis planteadas y explicar el nivel de significancia entre las unidades de trabajo de la fábrica (variable dependiente) y los seis factores propuestos (variables independientes).

Diagnosticar el estado del arte sobre los factores que influyen en la productividad de la fábrica de software

En este aparte, se realizó una revisión de la literatura sobre los factores que influyen en la productividad de la fábrica de *software*. Para este fin, se revisó el concepto de *fábrica de software y productividad*, y se siguió una metodología de tres fases, lo cual permitió que se identificaran algunos aspectos a considerar en la medición de la productividad: factores que influyen y modelos para medir la productividad.

Hernández et al. (2012) indican que en la ingeniería de *software* la medición de la productividad se ha centrado en el tiempo de entrega del producto, quizá influenciada en parte por las fórmulas utilizadas para estimar los proyectos de *software*. La estimación del esfuerzo de desarrollo de software es el proceso de predecir la cantidad de esfuerzo más realista requerida para desarrollar o mantener el *software*; mientras que, según IEEE (1992), la productividad del *software* se define como la relación entre el producto y el esfuerzo de trabajo.

Existe un gran interés de las organizaciones por medir la productividad de la fábrica de *software*, ya que les permitirá tener un panorama claro de la capacidad para utilizar sus recursos disponibles con el fin de producir bienes o servicios rentables que permitan dar satisfacción a las necesidades de sus clientes.

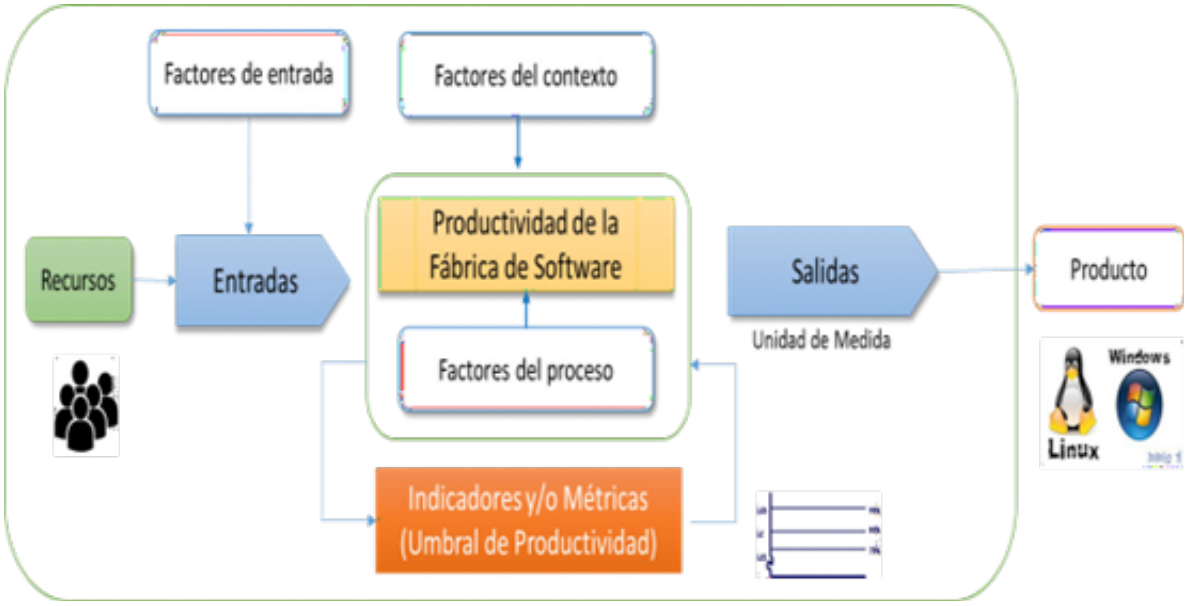
Medir la productividad es medir el rendimiento de una organización, tanto interna como externa, con el fin de evaluar su progreso. Al medir la productividad, una organización cuenta con indicadores que le permiten compararse en el mercado, a fin de plantearse acciones que incrementen su eficiencia global, a la vez, ello le permitirá utilizar todos los recursos de una manera eficaz y eficiente para obtener los mejores resultados posibles.

En otras palabras, los directivos necesitan conocer lo que están haciendo al compararse con el desempeño de períodos anteriores, por lo que surgen preguntas como: ¿se está incrementando, disminuyendo, avanzando o se está retrocediendo? ¿Cuál es la magnitud de ese avance o de ese retroceso? ¿Son eficaces los programas?

De acuerdo a Petersen (2011), el enfoque del proceso de mejora del software frecuentemente es mejorar la productividad, lo cual implica realizar una adecuada medición, por ello, es necesario no solo recopilar información de diversas fuentes y luego establecer resultados, sino que demanda un mayor esfuerzo en la comprensión de los diferentes elementos que la conforman, tal como se puede visualizar en el modelo conceptual para la medición de la productividad de Arcudia-Abad *et al.* (2007), que se muestra en la Figura 5.

El proceso de productividad está compuesto de diversos tipos de factores (entrada, salida, contexto, proceso) que la influencian, por ello es determinante homogeneizar la unidad de medida a utilizarse, de tal manera que los datos obtenidos permitan diseñar indicadores de medición que brinden resultados coherentes y comparables bajos diferentes entornos y puedan conllevar a establecer adecuadas estrategias de mejora del proceso.

Figura 5.
Modelo conceptual de la productividad



Nota. Adaptado de Capability Maturity Model Integration [CMMI] (2018).

Cabe señalar que antes de iniciar el proceso de medición de la productividad es necesario definir el objetivo de la medición, ya que ello permitirá establecer el nivel en el cual se quiere realizar la medición. Por ejemplo, se podría requerir medir la productividad de la fábrica de *software* para realizar una comparación con el mercado o se podría medir la productividad en solo algunos componentes de la fábrica de *software* para verificar posibles falencias que estaría generando cuellos de botella en la producción final o se podría medir la productividad de los roles que participan en el proceso de desarrollo de *software*.

En ese sentido, medir la productividad es una actividad compleja porque no hay un consenso sobre qué medir, además, incorpora diversos factores que en muchos casos no son tomados en cuenta (Asmild *et al.*, 2006; Cheikhi *et al.*, 2012; Khan et al., 2014; Machek *et al.*, 2012; Moreira et al., 2010; Scacchi, 1995; Yilmaz y O'Connor, 2011). También, los diversos estudios no han definido el ámbito de medición y han dificultado la detección de oportunidades de mejora en las organizaciones, por lo cual se plantea la siguiente pregunta de investigación: ¿cómo se viene midiendo la productividad en las fábricas de *software*?

A continuación, se presentará una revisión sistemática de los diferentes modelos que miden la productividad en las fábricas de *software*, así como de los factores que la afectan, pero, a su vez, se clasificará de acuerdo a la unidad de producción de la fábrica de *software* en la cual interviene, de tal manera que permita a los investigadores sobre esta temática contar con información actualizada sobre los diferentes estudios realizados hasta la fecha.

4.3. Fábrica de software

De acuerdo a estudios realizados por Clements y Northrop (2001), el concepto de *fábrica de software* trae grandes ventajas para las áreas de desarrollo de *software*, tales como la disminución en costos de producción por producto hasta en un 60 %; el ahorro del tiempo de puesta en el mercado hasta en un 98 % y la reducción del requerimiento en mano de obra hasta en un 60 %.

Otra ventaja es el mejoramiento de la productividad en alrededor de diez (10) veces y la calidad de cada producto derivado en hasta diez (10) veces menos errores, lo que aumenta el portafolio de productos y servicios ofrecidos por una fábrica de *software* y la posibilidad de ganar nuevos mercados.

El término fábrica de *software* ha sido usado desde los años sesenta y setenta en Estados Unidos y Japón. Cusumano (1989) la define como:

> una empresa productora de software que responda a características como: producción de software en gran escala, estandarización de tareas, estandarización del control, división del trabajo, mecanización y automatización, puede ser considerada una factoría de software. El desarrollo de una factoría implica que las buenas prácticas de Ingeniería de Software sean aplicadas sistemáticamente. (p. 23)

Por otra parte, Basili *et al.* (1992) proponen dividir la fábrica de software en dos subáreas: producción de *software* y producción de componentes. La primera realiza las solicitudes de productos (componentes para la producción del *software*), datos (estadísticas para la estimación de costos y plazos) y planos (modelos, métodos para el análisis y diseño del *software*). La segunda posee una base de componentes reutilizables, los cuales pueden ser también estadísticas y datos históricos de los cuales se apoya para dar respuesta a las solicitudes hechas por la subárea de producción de *software*.

Por otro lado, Li et al. (2001) organizan la fábrica de software basado en el Modelo Integrado de Madurez de Capacidades (CMMI) y la Organización Internacional de Normalización 9001 (ISO) en cinco entidades: técnicas, proceso, trabajadores involucrados, gestión de la factoría y activos del proceso, herramientas y componentes de código.

Para Kruchten (2004), el proceso unificado de rational (RUP) describe el proceso de desarrollo de *software* en términos de disciplinas, las cuales estructura en modelado de negocios, requisitos, análisis y diseño, implementación, pruebas y despliegue; y procesos de soporte, tales como configuración y gestión del cambio, gestión del proyecto y entorno.

Por otro lado, Rockwell y Gera (1993) adaptan el modelo Eureka (proceso de desarrollo de software distribuido) para el desarrollo de componentes en dos capas (servicios e interfaz), donde establecen un desarrollo de software por capas. Asimismo, Fernandes y Teixeira (2004) proponen un modelo que clasifica la factoría de acuerdo al alcance o fases de desarrollo definidas en el proceso. De esta manera, ofrecen una idea sobre cuál debe ser el ciclo de vida de un proyecto y lo clasifican en factoría de proyectos ampliada, factoría de proyectos de software, factoría de proyectos físicos y factoría de programas.

Yanosky (2005) plantea que una fábrica de *software* debe poseer un modelo de organización de la producción, una unidad de producción de componentes y una unidad de producción de *software*. Sus deficiencias radican en que no se contempla la gestión de proyecto y no utiliza estándares de calidad.

Fabri et al. (2004a; 2004b; 2005; 2007a; 2007b; 2007c; 2007d) y Pessoa *et al.* (2004) definen el

proceso de producción de *software* como aquel con características de fabricación, así, establecen que una fábrica de software posee dos unidades: producción de software y producción de componentes. Esta división es parecida a la propuesta por Basili *et al.* (1992) mencionada anteriormente.

Nomura *et al.* (2006) plantean una estructura de fábrica de *software* basada en los conceptos de reutilización de la fábrica de experiencia de Basili *et al.* (1992), la gestión operacional y división organizacional de la fábrica de software de Fernandes y Teixeira (2004), la aplicación de actividades de ingeniería de *software* citada por Swanson *et al.* (1991) y en la mejora de los métodos de trabajo citada por Cusumano (1989), así como prácticas de ingeniería mencionadas en la Guía de Fundamentos para la Dirección de Proyecto (PMBOK), los conceptos y las terminologías de RUP.

Asimismo, Trujillo (2007) propone un modelo que considera los principales elementos a definir para establecer un proceso de desarrollo con base en una metodología acorde a la magnitud del proyecto y al personal con que cuenta el mismo.

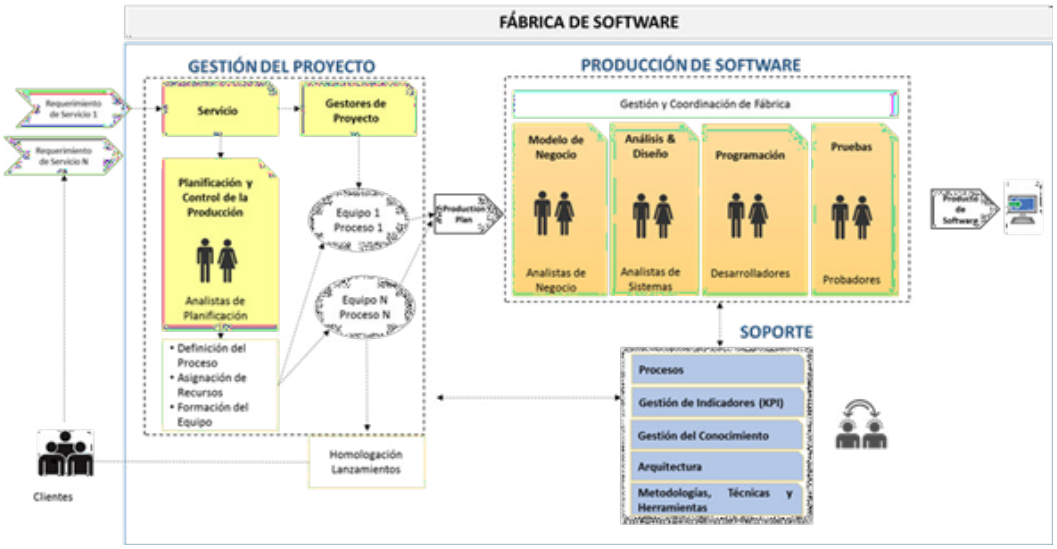
En la Tabla 2 se resumen los estudios sobre la estructura organizacional de la fábrica de *software* realizado a la fecha y una analogía entre los diferentes modelos propuestos.

Autor	Unidad de trabajo							
Basili *et al.* (1992)	Producción de *software*							Producción de componentes
Li *et al.* (2001)	Gestión de la factoría y activos del proceso	Proceso y trabajadores involucrados						Técnicas, herramientas y componentes de código
Rockwell y Gera (1993)		Servicios e interfaz						
Kruchten (2004)	*Project Management*	Modelado de negocio	Requisitos	Análisis y diseño	Implementación	Pruebas		Configuración y gestión del cambio, despliegue y entorno
Fernandes y Teixeira (2004)		Arquitectura proyecto de solución conceptual	Especificación lógica	Proyecto detallado	Construcción y pruebas	Pruebas de integración	Pruebas aceptación	
Yanosky (2005)		Organización de la producción	Producción de *software*					Producción de componentes
Nomura *et al.* (2006)		*Project Management*	Producción de *software* negocio	Arquitectura e ingeniería	Codificación y pruebas			
Trujillo (2007)		Fases de metodología de desarrollo de *software*						
Fabri *et al.* (2004a, 2004b, 2005, 2007a, 2007b, 2007c, 2007d); Pessoa *et al.* (2004)		Producción de *software*						Producción de componentes

Producción de componentes

Respecto a la Tabla 2, se puede mencionar que el modelo de Nomura *et al.* (2006) plantea una estructura de fábrica de *software* que cubre las distintas disciplinas del desarrollo de *software*, tal como se visualiza en la Figura 6.

Figura 6.
Estructura de una fábrica de *software*



Nota. Adaptado de Nomura et al. (2006)

Tal como plantean Nomura et al. (2006), una fábrica de *software* tiene tres grandes componentes: gestión de proyecto, producción de *software* y soporte. El primero se encarga de la relación con el cliente, la negociación, la planificación, la elaboración y el gerenciamiento de contratos de acuerdo con el planeamiento estratégico de la empresa, así como de la gerencia, elaboración, programación y distribución de las órdenes de servicio y la asignación de los recursos necesarios para la ejecución del servicio. De esta manera, se estructura en las siguientes unidades de trabajo: servicio, gestión de proyectos, planificación y control de la producción.

El componente de producción de *software* es el principal en el cual se desarrollan todas las actividades propias del proceso de desarrollo. Este componente está estructurado en las siguientes unidades de trabajo: modelo de negocio, análisis y diseño, programación y pruebas. Por otro lado, el componente de soporte brinda apoyo para la integración y el desarrollo de actividades del proceso de desarrollo, que incluye a la gestión de procesos, la infraestructura y el soporte, el diseño de interfaces, la inseguridad, la documentación, entre otros.

Este trabajo estuvo orientado a la búsqueda de estudios que se han realizado en el componente de producción de software, principalmente en las unidades de trabajo de análisis y diseño, programación y pruebas.

4.4. Planificación de la revisión

La búsqueda de la respuesta a la pregunta principal de la investigación se hizo con base en los siguientes planteamientos sobre la productividad en las fábricas de *software*:

- Q1: ¿Qué es la productividad de fábricas de *software*?
- Q2: ¿Qué factores influyen en la medición de la productividad y cómo se clasifican?
- Q3: ¿Qué modelos se han desarrollado?
- Q4: ¿Qué factores incluyen los modelos?

Por otra parte, se consideraron las bases de datos IEEE Explore, ACM Digital Library, ScienceDirect, Springer, DOAJ (Directory of Open Access Journals), Taylor y Francis, entre otros

(SCOPUS, Web of Knowledge). Los trabajos seleccionados responden a las cadenas de búsqueda dadas en la Tabla 3, que han sido aplicados al título, resumen y palabras clave, en el período comprendido de enero 2005 a abril 2017. Además, estos términos fueron adaptados para que coincidan con cada una de las preguntas de investigación y las necesidades individuales de cada uno de los buscadores.

Tabla 3. Cadenas de búsqueda utilizada en las bases de datos

Bases de datos	Cadena de búsqueda
IEEE Explore	(("*software* factory" OR "*software* process" OR "*software* development" OR "*software* development process" OR "*software* engineering") AND (productivity OR "development efficiency" OR "development effectiveness" OR "development performance" OR "models performance" OR "project performance" OR "productivity model" OR "productivity factors"))
	and refined by Content Type: Conference Publications Journals & Magazines Year: 2005-2017
ACM Digital Library	"query": {(+("*software* +factory" +OR +"*software* +process" +OR +"*software* +development" +OR +"*software* +development +process" +OR +"*software* +engineering") +AND +(productivity +OR +"development efficiency" +OR +"development effectiveness" +OR +"development performance" +OR +"models performance" +OR +"project +performance" +OR +"productivity +model" +OR +"productivity +factors"))}
	"filter": {"publicationYear":{ "gte":2005 }}, {owners.owner=ACM
ScienceDirect	("*software* factory" OR "*software* process" OR "*software* development" OR "*software* development process" OR "*software*
Springer DOAJ	engineering") AND (productivity OR "development efficiency" OR "development effectiveness" OR "development performance" OR
Taylor & Francis	"models performance" OR "project performance" OR "productivity
SCOPUS, Web of Knowledge	model" OR "productivity factors" OR "efficiency" OR "effectiveness" OR "performance")

4.4.1. Criterios de selección

Conducción de la revisión

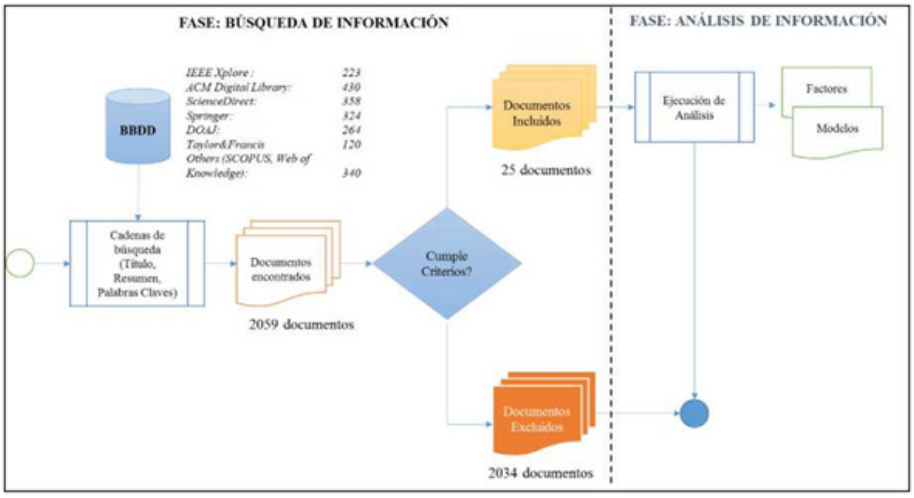
Los estudios primarios potenciales identificados en el proceso de búsqueda de acuerdo a la estrategia planteada fueron sometidos a un proceso de selección considerando los criterios de inclusión y exclusión establecidos (Tabla 4). Para ello fue necesario realizar una revisión previa a su contenido para poder determinar su relevancia en el presente estudio y, principalmente, si estos se aplican a los aspectos de la productividad de fábricas de *software*.

Tabla 4. Criterios de inclusión y exclusión

Criterios de inclusión	Criterios de exclusión
• Artículos publicados en fuentes con factor de impacto SJR. • Artículos que presentan modelos, factores, técnicas o metodologías sobre productividad de *software*. • Artículos que responden directamente a las preguntas de investigación.	• Estudios orientados a costos y estimación de esfuerzo. • Artículos que mencionan productividad de *software*, pero no aportan al tema planteado. • Artículos que no están dentro del contexto de fábrica de *software*. • Libros, tesis, pósteres, relatorios técnicos.

La mayoría de los artículos fueron descartados porque correspondían a la estimación de costos y esfuerzos para el desarrollo de *software*, a los factores para la estimación de esfuerzo o no tenían factor de impacto SJR. El proceso aplicado se representa en la Figura 7. Luego, se procedió a analizar los artículos para contestar las preguntas de investigación.

Figura 7.
Proceso de revisión de la literatura



El desarrollo de la revisión sistemática dio como resultado 2059 artículos de los cuales 25 fueron seleccionados de acuerdo a los criterios de inclusión y exclusión (Tabla 5).

Tabla 5. Estudios potencialmente elegibles y seleccionados

Base de datos	n.° de estudios potencialmente elegibles	n.° de estudios seleccionados
IEEE Xplore	223	1
ACM Digital Library	430	2
ScienceDirect	358	5
Springer	324	5
DOAJ	264	3
Taylor & Francis	120	0
Others	340	9
Total	**2059**	**25**

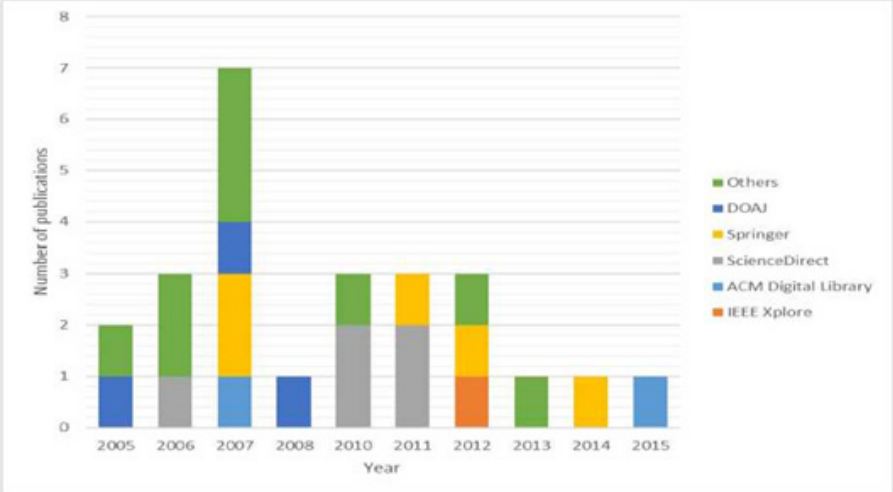
De la tabla anterior, ScienceDirect y Springer son las bases que contienen la mayor cantidad de publicaciones, la cual concentra el 40 %. Asimismo, en Taylor y Francis no se seleccionaron artículos, debido a que no cumplían con los criterios de inclusión.

4.5. Estadísticas

4.5.1. Tendencia temporal de publicaciones

Si se revisan los estudios de productividad de *software* que se han realizado a través de los años, se puede notar que se han venido elaborando publicaciones en las diferentes bases de datos (Figura 8), lo cual demostraría el interés que los investigadores han dado al tema de la productividad en la industria del *software* y, en particular, a la fábrica de *software* en las empresas.

Figura 8.
Número de publicaciones sobre productividad de software por año y base de datos



4.5.2. Número de publicaciones seleccionadas

En la Tabla 6 se muestra la distribución de los estudios por tipo de publicación y base de datos. Se observa que la mayor cantidad de publicaciones están en revistas y, respecto a base de datos, se han realizado mayores publicaciones en ScienceDirect y Springer, con 40 % del total de publicaciones de las cuales el 48 % corresponde a actas y conferencias.

Tabla 6. Distribución por tipo de publicación y base de datos

Tipo de publicación	Bases de datos						
	IEEE Xplore	ACM Digital Library	ScienceDirect	Springer	DOAJ	Otros	Total
Revistas	1	1	4	2	2	3	**13**
Actas	0	1	1	2	0	4	**8**
Conferencias	0	0	0	**1**	1	2	**4**
Total	**1**	**2**	**5**	**5**	**3**	**9**	**25**

4.6. Análisis

4.6.1. Q1: ¿Qué es la productividad de fábricas de software?

Banker y Kauffman (1991) establecen que la productividad de *software* es definida por la siguiente relación: Productividad = (Tamaño de aplicación desarrollada)/(Mano de obra consumida durante el desarrollo).

Por otra parte, de acuerdo a la IEEE (1992), se define a la productividad de *software* como la relación entre el producto del trabajo y el esfuerzo de trabajo. En adición, Fenton y Pfleeger (1997) discuten la medición de la productividad basado en la siguiente definición de productividad de *software*: Productividad = Tamaño/Esfuerzo.

El concepto que muchos autores (Basili et al., 1996; Blackburn *et al.*, 1996; Briand et al., 1998; Jeffery *et al.*, 2001; Maxwell, 2001; Nwelih y Amadin, 2008; Scacchi, 1995; Wagner y Ruhe, 2008b) toman en cuenta para medir la productividad es el que está basado en el modelo CRS, para lo cual se ha definido que: Productividad = Salida/Entrada.

En dicha definición, salida está representado por puntos de función (FP) y líneas de código (SLOC); mientras que entrada es el esfuerzo del proyecto en hora persona (PH) o mes persona (PM). De acuerdo a Wagner y Ruhe (2008b), el número de SLOC escrito o el número de FP implementados por hora persona del desarrollador es la medida de la productividad. También Card (2006) define la productividad como "la relación entre los productos producidos y los recursos consumidos" (p. 1).

En síntesis, los estudios realizados sobre la productividad de una fábrica de *software* se enfocan en la medición de la productividad en la unidad de trabajo de programación de *software*, como se aprecia en las entradas y salidas evaluadas (SLOC, FP, PH del desarrollador, entre otros), lo cual ha conllevado a que se desarrollen muchos modelos de medición del desempeño inexactos.

Sin embargo, los estudios no han revisado de manera integral a la fábrica de *software*, ya que su estudio considera, además, otras unidades de trabajo como análisis y diseño, y pruebas, asimismo, su productividad depende de diversos factores que influyen en el proceso, contexto, entre otros (Arcudia-Abad *et al.*, 2007; Nomura *et al.*, 2006).

4.6.2. Q2: ¿Qué factores influyen en la productividad y cómo se clasifican?

Premraj *et al.* (2005) muestran que el sector de negocio influye en la productividad de las personas y los sectores más influyentes son la banca, las telecomunicaciones y los seguros. Por otro lado, Moreira *et al.* (2010) plantean un modelo de desempeño basado en CMMI-Six Sigma y su aplicación para la predicción de la productividad en proyectos de una organización de *software*, que considera los siguientes factores: densidad de defectos en pruebas sistémicas, nivel de inestabilidad de los requerimientos, nivel de utilización de integración continua, nivel de experiencia, entorno de desarrollo, porcentaje de defectos en revisiones técnicas y cobertura de pruebas unitarias.

Por otra parte, Rodger *et al.* (2011) estudian la vigencia de los factores propuestos en Subramanian y Zarnich (1996) y Foss (1993), de esta manera, demuestran que los factores más significativos que afectan la productividad son el tamaño del equipo, la volatilidad en tipo de desarrollo de *software* (nuevo, mantenimiento), la plataforma de desarrollo y el tipo de lenguaje de programación. Asimismo, se determina que las herramientas de ingeniería de *software* asistida por computadora (CASE) no

tienen impacto en la productividad del *software*.

Según Rodríguez *et al.* (2011), sí existe una correlación estadística significativa entre tamaño del equipo, esfuerzo, productividad y duración del proyecto. De igual manera, Yilmaz y O'Connor (2011) demuestran la correlación entre la productividad de *software*, la productividad social y el capital social, así, se establecen los siguientes factores: motivación, proceso, complejidad, reutilización, tamaño del equipo, liderazgo del equipo, resultados colectivos, conocimiento de la información, transparencia de comunicación, relaciones sociales y reuniones frecuentes.

Por otro lado, Çetin y Alabaş-Uslu (2015) realizaron un estudio en compañías de *software* de Turquía y a través del análisis de regresión múltiple demostraron que existe una relación entre el tipo de proyecto y los factores severidad, el número total de defectos y el tamaño del equipo del proyecto. También demostraron la relación entre la duración del proyecto y los factores de metodología, el número de líneas bases, la diferencia entre fechas actuales y planificadas (línea base), la duración de análisis, la duración de desarrollo, la duración de pruebas, el número total de defectos, el número total de riesgos y la complejidad del proyecto.

Otros factores que impactan en la productividad son la complejidad (Khan *et al.*, 2014; Machek *et al.*, 2012; Yilmaz y O'Connor, 2011; Zhan *et al.*, 2012) y el nivel de experiencia (Moreira *et al.*, 2010). Por otro lado, existen trabajos que realizan un *ranking* de los factores, tal como se menciona en Paiva *et al.* (2010), donde se demuestra que la motivación y el compromiso tienen una alta influencia positiva.

Wagner y Ruhe (2008b) realizan la clasificación de factores en ocho categorías, las cuales al final se agrupan en factores blandos y técnicos. Los autores definen a los factores blandos como "todos los factores no técnicos que influyen en la productividad. Estos factores provienen principalmente del equipo y su entorno de trabajo" (p. 12), y a los factores técnicos, como "contiene todos los factores que tienen una relación directa con el producto, los aspectos técnicos del proceso y las herramientas que el desarrollador utiliza en el proyecto" (p. 12).

Asimismo, Trendowicz (2007) identifica varios contextos asociados a los factores, tales como factores del contexto en general, factores específicos del modelo, factores específicos del tipo de desarrollo, factores específicos del dominio y factores específicos de reutilización, a los cuales se define como "factores que se incluyen en el modelo para explicar la variación de la productividad dentro de un contexto determinado" (Trendowicz, 2007, p. 10).

Tal como se puede apreciar en la Tabla 7, la clasificación de Wagner y Ruhe (2008b) contempla categorías que no han sido cubiertos por Trendowicz (2007), como el entorno de desarrollo, la cultura corporativa, la cultura del equipo y entorno. Sin embargo, estas categorías son necesarias, ya que agrupan a factores que son transversales a toda la organización y no son exclusivas del producto o proyecto. Por ello, como base para el análisis, se utilizó la categoría propuesta por Wagner y Ruhe (2008b) para la agrupación de factores.

Tabla 7. Clasificación de factores por categoría

	Categorías de Wagner y Ruhe (2008a)	Categorías de Trendowicz (2007)	Descripción
Técnicos	Producto	Producto	"[...] la categoría producto contiene todos los factores que tienen una relación directa con el producto" (Wagner y Ruhe, 2008, p. 12). "Características de los productos de *software* que se desarrollan a lo largo de todas las fases de desarrollo. Estos factores se refieren a productos tales como código de *software*, requisitos, documentación, etc., y sus características, como complejidad, tamaño, volatilidad, etc." (Trendowicz, 2007, p. 10).
	Proceso	Proceso	"La categoría proceso se compone de los aspectos técnicos del proceso" (Wagner y Ruhe, 2008a, p.12). "Características de los procesos de *software*, así como métodos, herramientas y tecnologías aplicadas durante un proyecto de desarrollo de *software*. Incluyen, por ejemplo, la efectividad del aseguramiento de la calidad, la calidad de la prueba, la calidad del análisis y los métodos de documentación, la calidad y el uso de la herramienta, la calidad de la gestión del proceso o el grado de participación del cliente" (Trendowicz, 2007, p. 10).
	Entorno de desarrollo		"La categoría entorno de desarrollo contiene factores acerca de las herramientas que el desarrollador usa en el proyecto" (Wagner y Ruhe, 2008a, p. 12).
	Cultura corporativa		"Contiene los factores que están en un nivel más amplio para toda la empresa, mientras que..." (Wagner y Ruhe, 2008a, p. 14).
	Cultura del equipo		"[...] denota factores similares a nivel de equipo" (Wagner y Ruhe, 2008a, p. 14).
Blandos	Capacidades y experiencia	Personal	"[...] en Capacidades y Experiencia son factores resumidos que está relacionados con las personas" (Wagner y Ruhe, 2008a, p. 14). "Características del personal involucrado en el Proyecto de desarrollo de *software*. Estos factores usualmente consideran la experiencia y capacidades de las partes interesadas del proyecto, como miembros del equipo de desarrollo (por ejemplo, analistas, diseñadores, programadores, gerentes de proyecto), así como usuarios de *software*, cliente, subcontratistas, etc." (Trendowicz, 2007, p. 10).
	Entorno		"Representa las propiedades del entorno de trabajo" (Wagner y Ruhe, 2008, p. 14).
	Proyecto	Proyecto	"Factores específicos del proyecto." (Wagner y Ruhe, 2008a, p. 14). "Considerar las diversas cualidades de la gestión y organización del proyecto, las limitaciones de desarrollo, las condiciones de trabajo o la rotación del personal" (Trendowicz, 2007, p. 10).

En este contexto, se recopiló de la literatura existente 74 factores que afectan la productividad, los cuales han sido clasificados en las siete categorías propuestas por Wagner y Ruhe (2008b): producto, proceso, entorno de desarrollo, cultura corporativa, cultura del equipo, capacidades y experiencia, proyecto (Tabla 8).

Tabla 8. Cantidad de factores por componente y categoría que impactan en la productividad

Unidad de trabajo	Categoría	n.° de Factor	Referencias
Programación	Producto	17	Rodger *et al.* (2011); Machek *et al.* (2012); Yilmaz y O'Connor (2011); Cheikhi *et al.* (2012); Khan *et al.* (2014); Çetin y Alabaş-Uslu (2015); López *et al.* (2010); Pai *et al.* (2015); Unlutürk y Kurtel (2015)
	Proceso	13	Machek *et al.* (2012); Yilmaz y O'Connor (2011); Khan *et al.* (2014); Cheikhi *et al.* (2012); Çetin y Alabaş-Uslu (2015)
	Entorno de desarrollo	1	Moreira *et al.* (2010)
	Cultura corporativa	3	Premraj *et al.* (2005); Khan *et al.* (2014); Çetin y Alabaş-Uslu (2015)
	Cultura del equipo	8	Yilmaz y O'Connor (2011); Khan *et al.* (2014)
	Capacidades experiencia	6	Yilmaz y O'Connor (2011); Moreira *et al.* (2010); Khan *et al.* (2014)
	Entorno	0	
	Proyecto	12	Rodger *et al.* (2011); Yilmaz y O'Connor (2011); Khan *et al.* (2014); Çetin y Alabaş-Uslu (2015); Machek *et al.* (2012); Cheikhi *et al.* (2012)
Pruebas	Producto	3	Moreira *et al.* (2010); Pai *et al.* (2015); Unlutürk y Kurtel (2015)
	Proceso	2	Moreira *et al.* (2010)
Análisis y diseño	Producto	9	Cao *et al.* (2012)

4.6.3. Q3: ¿Qué modelos se han desarrollado?

Cardoso *et al.* (2010) afirman que un modelo es una representación simplificada de la realidad que a través de los diferentes componentes y procesos que forman parte del sistema en estudio se definen. Se han identificado diez modelos que miden la productividad en el desarrollo de *software*, los cuales están basados en diversos enfoques. En la Tabla 9 se resumen los modelos DEA, CMMI y Six Sigma, Lógica Difusa, Reutilización del Software, SEM, Factor Total de Productividad, IEEE Std. 1045 & ISO 9216-4.

Por otro lado, también existen trabajos que presentan modelos de evaluación de productividad de *software*, pero que no han sido seleccionados por no cumplir los criterios del presente estudio. Entre ellos se encuentran los modelos descritos por Sudhakar *et al.* (2012), como el modelo simple de productividad descrito por Card (2006); la tasa de entrega de productividad normalizada (PDR) de Jiang *et al.* (2007) y SEM de Khan *et al.* (2014), los cuales se presentan en la Tabla 10.

Tabla 9. Modelos de medición de la productividad de software

Id	Enfoque	Modelo	Contexto	Consideraciones	Referencias
M1	Análisis Envolvorio de Datos (DEA)	***Productividad*** = $e^{1.95}$ FP $^{0.7}$ (*) Donde: • Productividad: La productividad es el esfuerzo del trabajo • FP: Puntos de función	Grandes proyectos en bancos de Canadá.	Los autores miden la productividad considerando como salida a los puntos de función (FP) y la entrada, el esfuerzo de desarrollo, utilizando DEA como técnica de medición.	Asmild *et al.* (2006)
M2	CMMI & Six Sigma	***Productividad*** = 32.087 – 3.637 DDST + 11.71 LRU – 9.451 LCIU – 0.8187 LEX DENV Donde: • DDST: Densidad de Defectos en Pruebas Sistémicas DDST = 1.8955 - 0.5087 PDTR - 1.6020 UTC • LRU: Nivel de Inestabilidad de los Requerimientos • LCIU: Nivel de Utilización de Integración Continua • LEX: Nivel de Experiencia • DENV: Entorno de Desarrollo • PDTR: Porcentaje de Defectos en Revisiones Técnicas • UTC: Cobertura de Pruebas Unitarias	5 proyectos del Atlantic Institute| una empresa mediana, evaluada en CMMI nivel 3, cuyo objetivo real es alcanzar el nivel de madurez 5.	Los autores proponen un modelo de productividad basado en CMMI y Six Sigma. En el artículo, ellos definen la productividad como la Productividad General de los Proyectos (GPP) y usan la técnica de regresión lineal para establecer su modelo.	Moreira *et al.* (2010)
M3	Lógica Difusa	***Productividad*** = 10.3 + 0.31701* N&C N&C: Código Nuevo y modificado. Se considera como líneas físicas de código (LOC). N&C es compuesta de código agregado y modificado.	Conjunto de datos de 140 pequeños programas desarrollados con prácticas basadas en el Proceso de *Software* Personal (PSP)	Los autores proponen que un modelo de lógica difusa se podría usar para estimar y predecir la productividad del desarrollo de *software*.	López *et al.* (2010)
M4	Modelo de Ecuaciones Estructurales (SEM)	***Productividad*** = *f* (Motivación, Proceso, Complejidad, Reutilización, Tamaño del Equipo, Productividad Social, Capital Social) • Productividad Social = *f* (Liderazgo del Equipo, Resultados Colectivos, Conocimiento de la Información) • Capital Social= *f* (Transparencia de la Comunicación, Relaciones Sociales, Reuniones Frecuentes)	El análisis se realizó con 227 participantes. Alrededor del 63 % de los participantes eran posgraduados.	Los autores proponen un modelo empíricamente validado para medir el capital social, la productividad social y la productividad del desarrollo de *software*.	Yilmaz y O'Connor (2011)
M5	Factor Total de Productividad	• Definición de salidas (y) • Definición de entradas (x) • Determinación de precios de entradas y salidas (p, w)	No se explica en el artículo.	Modelo de medición de la productividad total del proceso de *software* basado en el factor total de productividad.	Machek *et al.* (2012)

		Cálculo del factor total de productividad (Índices de Productividad de Fisher o Törnqvist)			
M6	ISO 9126-4 IEEE Std. 1045	Modelo de productividad consolidado ***Productividad*** = *f* (Eficiencia, Completitud, Precisión) Donde: • Eficiencia = f (tiempo de tareas, costo de tarea, frecuencia de error, eficiencia del usuario, eficiencia del usuario en comparación con un experto, eficiencia de la tarea (tiempo), eficiencia de la tarea (costo), proporción del tiempo que el usuario es productivo) • Completitud = f (% de tareas realizadas, % de usuarios que tuvieron éxito en completar la tarea, logro del objetivo de porcentaje para cada unidad de tiempo) • Precisión = f (% de tareas logradas correctamente, % de usuarios que logran la tarea correctamente, ¿Con qué frecuencia encuentra el usuario resultados inexactos al realizar la tarea? ¿Con qué frecuencia realiza la tarea el usuario con una precisión inadecuada?)	No se explica en el artículo.	Los autores revisan los estándares internacional IEEE Std. 1045 e ISO 9216-4 y definen cómo ambos estándares pueden usarse juntos para la medición de la productividad.	Cheikhi *et al.* (2012)
M7	DEA con rendimientos variables de escala (DEA VRS, por las siglas del inglés Data Envelopment Analysis Variable Returns to Scale)	***Productividad*** = *f* (n (OT), n (RT), n (PT), PO (MT, o), PXO (MT), PR (MT, *e*), PXR (MT), RO (MT, o), RO (MT)) / (hora de trabajo variable) • n (O_T): Recuento de tipos de objeto por técnica • n (R_T): Recuento de tipos de relación por técnica • n (P_T): Recuento de tipos de propiedad por técnica • P_O (M_T, o): Recuento de número de propiedades para un tipo de objeto dado • P_{XO} (M_T): Promedio de número de propiedades para un tipo de objeto dado • P_R (M_T, e): n.° de propiedades de un tipo de relación y los tipos de roles que lo acompañan • P_{XR} (M_T): Promedio de propiedades por tipo de relación • R_O (M_T, o): Cantidad de tipos de relaciones que se pueden conectar a un determinado tipo de objeto • R_O(M_T): n.° promedio de tipos de relaciones que se pueden conectar a un determinado tipo de objeto	Información sobre 25 proyectos SAD basados en UML. Esta información se obtuvo de diferentes empresas.	Un enfoque para evaluar la eficiencia relativa de los proyectos de *software* utilizando estas medidas de complejidad como salidas. El modelo se aplica en el proceso de análisis y diseño de sistemas (SAD).	Cao *et al.* (2012)
M8	Reutilización del *Software*	Productividad = $\sum_{i=1}^{n}(r_i + f_i + l_i + e_i) / \sum t$ Donde: r_i = Reutilización f_i = Funcionalidad	En Nigeria, por instancia, con referencias especiales a la industria de TI.	Este modelo considera factores como reutilización, complejidad, longitud, funcionalidad y esfuerzo.	Nwelih y Amadin (2008)

		l_i = Longitud c_i = Complejidad ΣI = Esfuerzo			
M9	DEA VRS	***Productividad*** = FP, Calidad/Esfuerzo Total (Esfuerzo de desarrollo + EoC + EoNC) • Esfuerzo de conformidad (EoC) = (Evaluación + Prevención) Costos. Ejemplo: esfuerzo de revisión • Esfuerzo de no conformidad (EoNC) (Interna + Externa) Costos de fallas. Ejemplo: El esfuerzo de retrabajo, incluido el esfuerzo de falla • Calidad = f(Defectos)	El modelo se aplica a los datos recogidos en 79 proyectos de desarrollo de *software* de una organización CMMI nivel 5.	Los autores desarrollan un modelo para medir la productividad de los proyectos de *software* e identificar las mejores prácticas. Este estudio muestra que incluir EoC y EoNC como entradas tiene un impacto positivo en la frontera de mejores prácticas.	Pai *et al.* (2015)
M10	Técnica Estadística	Productividad = $\sum_{i=1}^{n}(Calidad_i * Cantidad_i)/$ $Horas\ Netas\ de\ Tareas_i) * Métodos\ Ponderados$ • Calidad = 1- ((10 * total # de Defectos Serios + 3 * total # de Defectos Medianos + total # de Defectos Triviales) / LOC) • Cantidad = LOC • Métodos Ponderados = (3 * (total# de constructores + total# de destructores) + 5 * total # de selectores + 9 * total # de iteradores + 15 * total # de modificadores) * (1/N) Donde: N es el número total de métodos para un programador individual	Durante esta fase, tres estudiantes de nivel sénior trabajaron en el proceso de medición de dos proyectos.	Un método para medir la productividad de los programadores de *software* de manera individual. Este proporciona una opinión común para comprender, controlar y mejorar las prácticas de ingeniería de *software*.	Unlurturk y Kurtel (2015)

Nota. *Para una mejor comprensión del modelo, se ha simplificado la fórmula.

Tabla 10. Otros modelos de medición de la productividad de software no seleccionados

ID	Técnica/Modelo	Fórmula/Descripción	Consideraciones	Referencias
1	Modelo simple de productividad	• Productividad física = n.° de LOC/horas hombre o días o meses • Productividad funcional = n.° de puntos de función/horas hombre o días o meses • Productividad económica = valor/costo Donde: Valor = f(precio, tiempo, calidad, funcionalidad)	Este modelo considera las entidades tales como producto, proceso o subproceso, requisitos, valor, costo y esfuerzo.	Card (2006)
2	Tasa de entrega de productividad normalizada	log (PDR) = 2.8400+0.3659 x log (Team Size)-0.6872 x I(3GL) - 1.2962 x I(4GL) – 1.3225 x I(ApG) – 0.1627 x I(MR) – 0.4189 x I(Multi) – 0.3201 x I (PC) – 0.4280 x I(OO) – 0.2812 x I (Event) + 0.7513 x I(OO:Event) – 0.2588 x I(Business) – 0.0805 x I(Regression) + 1.0506 x I(Business:Regression) Normalizada PDR = (Esfuerzo de Trabajo Normalizado) / (Puntos de Función Ajustados)	Utiliza dos variables continuas tamaño promedio del equipo y PDR; seis variables categóricas como tipo de lenguaje, tipo de desarrollo, plataforma de desarrollo, técnicas de desarrollo, herramienta CASE utilizada y metodología.	Jiang *et al.* (2007)
3	Modelo de ecuaciones estructuradas	***Productividad*** = f (tecnología, cultura de trabajo, interés en trabajo individual, complejidad, tamaño del equipo, productividad humana) Productividad humana = f (habilidades del gerente, unidad de equipo, vida social, frecuencia de reunión)	Los autores desarrollan un modelo de productividad empírico, así, muestran la correlación entre la productividad de *software* y la productividad humana.	Khan *et al.* (2014)

Nota. Adaptado de Sudhakar *et al.* (2012)

Para la evaluación de los modelos presentados se consideraron los siguientes criterios: la formulación de la productividad que explicita una función para medir la productividad; la cobertura del modelo que especifica si el modelo cubre parte o todas las unidades de trabajo de la fábrica de *software* (análisis y diseño, programación, y pruebas de *software*), y el sustento teórico que especifica si el modelo presenta sustento o fundamento en teorías.

Otros criterios considerados fueron la validación que especifica si el modelo evaluado ha sido probado en proyectos reales de una organización o por repositorios de datos, y el preprocesamiento que es descrito por el modelo como un procedimiento para la captura y preprocesamiento de los datos para medir la productividad. En la Tabla 11 se pueden visualizar los criterios de los modelos de productividad.

Tabla 11. Evaluación de modelos de productividad de *software*

Criterio	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10
Formulación de productividad	√	√	√	No	No	No	No	Sí	No	√
Cobertura del modelo	PRO	PRO - TES	PRO	PRO	PRO	PRO	AD	PRO	PRO - TES	PRO - TES
Sustento teórico	√	√	√	√	√	√	√	√	√	√
Validación	NE	√	√	NE	NE	NE	√	NE	√	√
Preprocesamiento	NE	NE	NE	NE	NE	NE	NE	NE	√	NE

Nota. √ Si cumple el criterio establecido; NE: No especifica; PRO: Programación; TES: Pruebas; AD: Análisis y diseño.

Por otra parte, en la Tabla 12, se agrupan los modelos revisados en cada uno de los componentes que conforman la fábrica de *software*.

Tabla 12. Modelos de medición de la productividad de *software* por componente

Programación	M1	Análisis Envoltorio de Datos (DEA)	Asmild *et al.* (2006)
	M2	CMMI & Six Sigma	Moreira *et al.* (2010)
	M3	Lógica Difusa	López *et al.* (2010)
	M4	Modelo de Ecuaciones Estructuradas	Yilmaz y O'Connor (2011)
	M5	Factor Total de Productividad	Machek *et al.* (2012)
	M6	ISO 9126-4 IEEE Std.1045	Cheikhi *et al.* (2012)
	M8	Reutilización de *Software*	Nwelih y Amadin (2008)
	M9	DEA con rendimientos variables de escala (DEA VRS)	Pai *et al.* (2015)
	M10	Técnicas Estadísticas	Unluturk y Kurtel (2015)
Pruebas	M2	CMMI & Six Sigma	Moreira *et al.* (2010)
	M9	DEA con rendimientos variables de escala (DEA VRS)	Pai *et al.* (2015)
	M10	Técnicas Estadísticas	Unluturk y Kurtel (2015)
Análisis y diseño	M7	DEA con rendimientos variables de escala (DEA VRS)	Cao *et al.* (2012)

4.6.4. Q4: ¿Qué factores incluyen los modelos?

Tomando en consideración la información encontrada en la literatura, en la Tabla 13 se visualizan los factores incluidos en los distintos modelos, los cuales se clasifican por componentes.

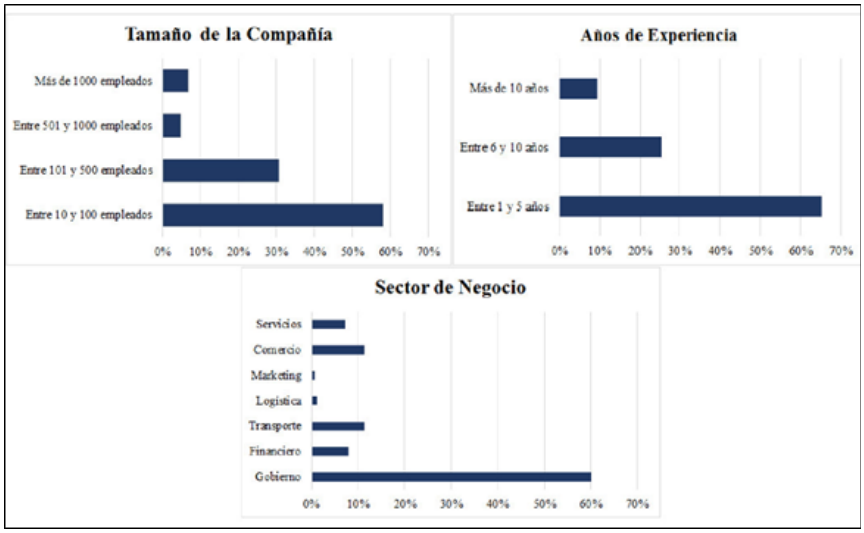
Tabla 13. Factores incluidos en los modelos de productividad de *software* por componente

Unidad de trabajo	Categoría	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10
	Producto			F10, F15, F16	F3, F4, F17	F1, F2, F3, F4, F5, F6, F7, F8, F17	F9, F10		F4, F5, F6, F17	F10	F9
	Proceso	F35, F37, F38, F39			F31, F32	F30, F34	F33				
	Entorno de desarrollo cultura corporativa		F45								
Programación	Cultura de equipo				F49, F50, F51, F52, F53, F54						
	Capacidades y experiencia		F58		F57, F59, F60, F61						
	Proyecto					F64, F65, F66, F67, F68, F69, F70, F71, F73	F66, F72				
Pruebas	Producto		F18, F19, F20							F18, F19	F18, F19
	Proceso		F43, F44								
Análisis y diseño	Producto							F21, F22, F23, F24, F25, F26, F27, F28, F29			

a. Describir cómo los factores de productividad afectan a las unidades de trabajo de análisis y diseño de las fábricas de *software*

Una vez aplicado el cuestionario vía *online*, al realizar su análisis y sistematización, este arrojó los siguientes resultados: el 88.7 % de los participantes laboran en fábricas de *software*, cuyo tamaño oscila entre 10 y 500 personas, donde solo el 11.3 % de los participantes que laboran en empresas tienen más de 500 personas, tal como se visualiza en la Figura 9.

Figura 9.
Datos de la caracterización de la muestra



Por otro lado, el 60 % de los participantes laboran en fábricas de *software* que brindan servicios al sector estatal, seguido de los sectores transporte y servicios con el 11.3 % cada uno, y el 17.4 % se encuentra dividido entre los demás sectores. Asimismo, el 65.3 % de los participantes tiene entre uno y cinco años de experiencia en fábricas de *software* y el 34.7 % de los participantes cuenta con más de cinco años de experiencia.

b. Sector de negocio

Se identificaron tres nuevos factores que influyen en la productividad de la fábrica de *software*: uso de reglas y eventos, heterogeneidad del equipo y comunicación osmótica. Para este fin, se revisaron teorías sobre la perspectiva de la acción del lenguaje y la teoría de la memoria transactiva, así como metodologías para sustentar estos factores. Para validarla, se hizo un estudio empírico en fábricas de *software* de Perú.

Cusumano (1989) define al *software factory* como una empresa que tiene características de producción de *software* en gran escala, estandarización de tareas y de control, división del trabajo, mecanización, automatización y aplicación sistemática de buenas prácticas de ingeniería de *software*.

En ese sentido, el implementar una fábrica de *software* trae grandes ventajas, tales como la disminución en costos de producción por producto hasta en un 60 %, el ahorro del tiempo de puesta en el mercado hasta en un 98 %, la reducción del requerimiento en mano de obra hasta en un 60 %, el mejoramiento de la productividad en alrededor diez veces y la calidad de cada producto derivado en hasta diez veces menos errores, lo cual aumenta el portafolio de productos y servicios ofrecidos, así como la posibilidad de ganar nuevos mercados (Clements y Northrop, 2001).

Por lo tanto, al medir la productividad, una fábrica de *software* cuenta con indicadores que le permiten compararse en el mercado, de tal manera que le ayuda a plantearse acciones que incrementen su eficiencia global, al tiempo que utiliza todos los recursos de una manera eficaz y eficiente para obtener los mejores resultados posibles. En otras palabras, es necesario saber cómo la organización se está desempeñando en comparación con períodos anteriores y sus competidores. De esta manera, surgen las siguientes preguntas: ¿se está incrementando, disminuyendo, avanzando o se está retrocediendo? ¿Cuál es la magnitud de ese avance o de ese retroceso? ¿Son eficaces las estrategias implementadas?

Todos los modelos encargados de medir la productividad en una fábrica de *software* se basan en

diferentes elementos, tales como procesos, recursos, unidad de medida, entre otros, pero dejan de lado el hecho de cómo afecta el recurso y cuál es su repercusión en la medición de la productividad. Por ejemplo, la motivación y la confianza en el equipo son factores que podrían tener un impacto positivo en la productividad, mientras que, en un equipo desmotivado, podría tener un efecto contrario (Yilmaz y O'Connor, 2011).

Este es el motivo por el cual se están desarrollando estudios para identificar los factores que afectan la productividad; sin embargo, se evidencia que (i) los factores identificados se orientan a la unidad de trabajo de programación, lo que no se podría generalizar en la fábrica de *software*, dado que esta contempla otras unidades de trabajo y cada unidad tiene sus particularidades; (ii) existen factores que influyen en la productividad en otros dominios de conocimiento, pero no han sido analizados en la fábrica de *software*.

Además, no se ha considerado que la productividad depende de diversos factores que van más allá del *input y output*, los cuales influyen en el proceso, contexto, entre otros aspectos (Arcudia-Abad *et al.*, 2007; Nomura *et al.*, 2006).

En esta parte, se procedió a incluir nuevos factores que afectan la productividad en las fábricas de *software*, las cuales están sustentadas en las teorías de perspectiva de la acción del lenguaje (LAP), teoría de la memoria transactiva (TMT) y las buenas prácticas del agilismo como el *timeboxing*.

También se estudió cómo afectan dichos factores en las unidades de trabajo como son a) análisis y diseño, b) programación y c) pruebas. Para la validación de la propuesta, se realizaron 150 encuestas. El presente trabajo es parte de una investigación sobre modelos de productividad para fábricas de *software*.

4.7. Modelo conceptual

A continuación, se muestran las unidades de trabajo y los factores que la afectan.

4.7.1. Unidades de trabajo

En la Tabla 14, se pueden identificar tres grandes componentes en la organización de la fábrica de software: gestión del proyecto, producción de software y soporte. Asimismo, cada componente puede estar dividido en unidades de trabajo como se puede observar en la Tabla 15. Estas fueron obtenidas del análisis y emparejamiento realizado en los trabajos sobre estructura de la fábrica de software, ya que en muchos casos los autores traslapan las unidades o, en su defecto, le han dado otra denominación.

Tabla 14. Componentes de la fábrica de *software*

Componente	Descripción	Referencias
Gestión de proyecto	Responsable de la relación con el cliente, negociación, planificación, elaboración y gerenciamiento de contratos de acuerdo con el planeamiento estratégico de la empresa, así como de la gerencia, elaboración, programación y distribución de las órdenes de servicio y asignación de los recursos necesarios para la ejecución del servicio, lo cual se estructura en las siguientes unidades de trabajo: servicio, gestión de proyectos y planificación y control de la producción.	Li *et al.* (2001); Kruchten (2004); Yanosky (2005); Nomura *et al.* (2006)
Producción de *software*	Responsable de actividades propias del proceso de desarrollo. En esta componente se incluyen las unidades de trabajo de modelado de negocio, arquitectura de solución, proyecto conceptual, requisitos, análisis y diseño, implementación y pruebas.	Basili *et al.* (1992); Li *et al.* (2001); Rockwell y Gera (1993); Kruchten (2004); Fernandes y Teixeira (2004); Yanosky (2005); Nomura *et al.* (2006); Trujillo (2007); Fabri *et al.* (2004a, 2004b, 2005, 2007a, 2007b, 2007c, 2007d); Pessoa *et al.* (2004)
Soporte	Proporciona apoyo para la integración y desarrollo de actividades del proceso de desarrollo, incluyéndose entre otros: la gestión de procesos, la infraestructura y soporte, el diseño de interfaces, la seguridad, la documentación, etc.	Basili *et al.* (1992); Li *et al.* (2001); Kruchten (2004); Yanosky (2005); Nomura *et al.* (2006); Fabri *et al.* (2004a, 2004b, 2005, 2007a, 2007b, 2007c, 2007d); Pessoa *et al.* (2004)

Tabla 15. Unidades de trabajo de los componentes de la fábrica de *software*

Componente	Unidad de trabajo	Descripción	Referencias
Gestión del proyecto	Servicio	Definición del tipo de servicio que se brindará al cliente.	Nomura *et al.* (2006)
	Gestores de proyecto	Asignación de la capa de gestión que estará a cargo de le ejecución del proyecto.	
	Planificación y control de la producción	Definición del equipo que atenderá los requerimientos, así como la información necesaria para el plan de producción.	
Producción de *software*	Modelo de negocio	Definición y elaboración del modelado de negocio del producto a desarrollar.	Li *et al.* (2001); Kruchten (2004); Yanosky (2005); Nomura *et al.* (2006)
	Análisis y diseño	Definición de la arquitectura base y la especificación funcional del requerimiento.	
	Programación	Implementación de la solución requerida por el cliente.	Kruchten (2004); Fernandes y Teixeira (2004); Nomura *et al.* (2006)
	Pruebas	Definición y ejecución de casos de prueba que son corroborados con los criterios de aceptación definidos por el cliente.	
Soporte	Procesos	Establecimiento de los procesos clave para la atención de los requerimientos de acuerdo con el tipo de servicio a brindar al cliente.	Li *et al.* (2001)
	Gestión de indicadores	Definición y medición de los KPI que se utilizarán en el proyecto.	CMMI (2016)
	Gestión del conocimiento	Políticas y activos organizacionales.	
	Arquitectura	Soporte de activos de programación o componentes de *software*.	Basili *et al.* (1992); Li *et al.* (2001); Yanosky (2005)
	Metodologías, técnicas y herramientas	Teniendo como referencia el proceso definido, se determinan las metodologías, técnicas y herramientas que darán soporte al proyecto.	Li *et al.* (2001); Kruchten (2004)

La Figura 6 ilustra el funcionamiento de una fábrica de *software* en la cual se puede apreciar que los clientes solicitan diferentes tipos de requerimientos, los cuales son atendidos por la componente de gestión del proyecto (PM) en la que, tomando en consideración la capacidad de la fábrica, se acuerdan los requerimientos a ser atendidos, se estima y planifica la atención en períodos acordados con el cliente, y se genera el plan de producción.

Luego, los requerimientos aceptados en la PM son trasladados a la componente de producción de *software* donde se realizan las actividades claves para el desarrollo del producto final, tales como análisis y diseño, programación y pruebas. La componente de soporte acompaña y brinda soporte en las diferentes actividades que se realizan a través del ciclo de producción de *software*.

El 90 % del esfuerzo de la fábrica de *software* es absorbida por el componente de producción de software, y de esta, las unidades de trabajo análisis y diseño, programación y pruebas consumen el 85 % del esfuerzo de esta componente (Jacobson *et al.*, 2000). Por ello, este estudio de investigación está orientado a la búsqueda de los factores que afectan la productividad en esta componente y las

unidades de trabajo de análisis y diseño, programación y pruebas.

4.7.2. Factores propuestos

Se revisaron 40 teorías que han sido aplicados en la formación de equipos de trabajo, comportamiento de equipos, comportamiento organizacional y comunicación efectiva para conocer su relación con la productividad, de manera que se descartan aquellas que no tenían relación. Posteriormente, se identifican los factores que afectan a la productividad con base en estas teorías y otras fuentes, varias de ellas ya identificadas en la literatura, y se obtienen cuatro factores nuevos y dos factores existentes que solo han sido aplicados a la unidad de trabajo de programación, que se describen a continuación.

Primero, se encontró al factor uso de reglas y eventos (RULE). Este se refiere al cumplimiento de normas y reglas de comportamiento que son definidas por el mismo equipo, lo cual permite que puedan cumplir metas y objetivos de manera clara y específica, a su vez, estos pueden ser medidos. Una de las buenas prácticas que se implementan en estas normas es el *timeboxing* que asegura que los miembros del equipo no ocupen demasiado o muy poco trabajo durante un período de tiempo particular, asimismo, no gasten su tiempo y energía en trabajos para los que tienen poca claridad (Guide to the Scrum Body of Knowledge [SBOK], 2016).

Algunas de sus ventajas son el proceso de desarrollo eficiente, menos gastos generales y la alta velocidad de los equipos, lo cual conlleva a una mejora de la productividad. Por ejemplo, el equipo debería participar de un evento llamado "reunión diaria", que les permita ir revisando el avance del producto. Esta reunión está determinada en un timeboxing de 15 minutos, a fin de que sea efectiva.

En adición, como las actividades de análisis involucran una adecuada planificación y constante interacción con el cliente y el equipo, de tal manera que se realice una buena elicitación de los requerimientos y, a su vez, estos se transformen en el producto deseado; es necesario que el analista se enfoque en delimitar el alcance del análisis. Por lo tanto, deben establecerse actividades prioritarias que ayuden al logro del objetivo y no se distraiga en actividades que pudiesen retrasar; por todo ello, se requiere establecer reglas y eventos que den soporte al cumplimiento de estas actividades (SBOK, 2016).

Asimismo, se encontró al factor que se refiere a la mejora del desempeño de heterogeneidad del equipo (HETE) a través de la formación de equipos multifuncionales, de tal manera que facilita el aprovechamiento de conocimiento de todos los roles participantes. Al respecto, Ancona y Caldwell (1992) demuestran que la diversidad funcional en equipos puede influenciar el desempeño a través de proceso internos y comunicación externa. Asimismo, los autores señalan que la diversidad produce procesos que facilitan el desempeño, pero también lo impiden. Por ello, Cohen y Bailey (1997) definen que la diversidad del equipo resulta en un mejor uso del conocimiento, mejor comunicación y cooperación en equipos, lo que conlleva a una mejor productividad del equipo.

Otro factor importante es la diversidad o heterogeneidad en el equipo, dado que es una buena manera de incorporar nuevas perspectivas que pueden ayudar a los integrantes a superar problemas de distintos ángulos. Por ello, si en el equipo de programación se cuenta con diversos perfiles, el resultado conllevará a una mejor comprensión de los requerimientos y una forma diferente de plantear alternativas de solución. Asimismo, al ser la programación una actividad que incorpora diversas disciplinas, se requiere que el equipo sea multifuncional, además de tener habilidades y capacidades que permitan ser más eficientes en el desarrollo del producto.

En el caso del factor tipo de comunicación "cara a cara" en el equipo (FACE), este hace referencia a la forma en cómo las personas interactúan para mejorar el entendimiento de los requerimientos, así como facilitan el proceso de aprendizaje y colaboración en el equipo. Por lo tanto, Simmons (1991) define que la comunicación tiene gran influencia en la productividad y, de acuerdo a la teoría de la perspectiva de la acción del lenguaje (Flores y Ludlow, 1980), considera que el lenguaje es la dimensión principal de la actividad de cooperación humana. Por ejemplo, podría considerarse la importancia que le han dado a este factor en las metodologías ágiles, por lo que es uno de los principios fundamentales del manifiesto ágil que "el método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara" (SBOK, 2016).

En ese sentido, la comunicación "cara a cara" es esencial para crear un vínculo entre los participantes del proyecto, por lo que se convierte en el método más eficiente y eficaz de comunicación con y dentro de un equipo de desarrollo, ya que produce un intercambio de energía que dinamiza la experiencia humana y facilita la interacción entre muchas personas al mismo tiempo. Estos propicia mayor fluidez, efectividad y rapidez respecto al diálogo, la discusión y la toma de decisiones.

Por otro lado, el factor tipo de comunicación "osmótica" en el equipo (OSMO) se refiere a la información que fluye en el fondo auditivo de los miembros del equipo, de manera que ellos captan la información relevante, como si fuera por ósmosis, lo que se produce normalmente por estar en el mismo ambiente de trabajo. Así, cuando una persona hace una pregunta, otros en el ambiente pueden sintonizar o desconectarse y, de esta forma, se contribuye a la discusión o continúan con su trabajo.

En la teoría de la memoria transactiva, Wegner (1987) define que los miembros son capaces de beneficiarse de los conocimientos y experiencias de otros para desarrollar una buena comprensión a través del compartir en el grupo, por ello, es necesario entender qué tan importante es la comunicación osmótica para la adquisición de conocimientos en ambientes de colaboración, lo cual ayuda al incremento de la productividad.

También cabe señalar que este factor podría tener un efecto negativo en caso de que el equipo de trabajo requiera una mayor tranquilidad y concentración. Los equipos de desarrollo normalmente trabajan en lugares colocalizados, por lo cual este tipo de comunicación es necesaria en el equipo y se requiere que todos conozcan las actividades, a fin de evitar la dependencia y el equipo pueda avanzar.

Un factor que también se halló fue el tamaño del equipo (SIZE) que se refiere a la cantidad de participantes dentro del equipo de trabajo, el cual debe ser necesario para asegurar las habilidades y capacidades requeridas, al tiempo que facilite la colaboración entre los miembros del equipo, ya que mientras mayor sea la cantidad de canales de comunicación, será más complejo el traslado de información hacía los diferentes niveles.

Según Rodríguez *et al.* (2011), el tamaño del equipo de software es considerado uno de los *drivers* de productividad del proyecto. Asimismo, muchos autores han realizado estudios acerca de este factor y han determinado su impacto en la productividad del software, pero solo ha sido aplicado en la unidad de trabajo de programación (Abdel-Hamid y Madnick, 1991; Abdel-Hamid, 1989; Alba y Chicano, 2005, 2007; Brooks, 1995; Chicano *et al.*, 2011; Di Penta *et al.*, 2007; Gueorguiev et al., 2009; Harman y Jones, 2001; Kremmel et al., 2010, 2011; Montgomery, 1981; Norden, 1958, 1977; Pendharkar y Rodger, 2007; Pillai y Sukumaran, 1997; Smith *et al.*, 2001; Trendowicz y Munch, 2009; Verner *et al.*, 2007). Por tal motivo, se plantea una interrogante acerca de su influencia en la unidad de trabajo de análisis y diseño.

Por lo tanto, el tamaño del equipo es un factor crítico en el desarrollo del *software*, ya que mientras más integrantes tengan un equipo, más canales de comunicación se generan. Por ello, es necesario que el equipo sea lo suficientemente grande para asegurar las habilidades adecuadas, pero lo suficientemente pequeño para facilitar la comunicación.

El último factor encontrado fue la confianza en el equipo (TRUS) que se refiere a la capacidad de los miembros del equipo para confiar entre ellos, lo cual contribuye a una mayor cohesión y, en consecuencia, garantiza una mejor productividad. Al respecto, Sridhar *et al.* (2007) encuentran que la confianza entre los miembros del equipo y sus comunicaciones efectivas tienen correlación positiva con el éxito del proyecto, lo cual afecta el desempeño del equipo de desarrollo. Asimismo, diversos autores han demostrado la importancia de la confianza en la productividad (Sudhakar *et al.*, 2011; Yilmaz y O'Connor, 2011).

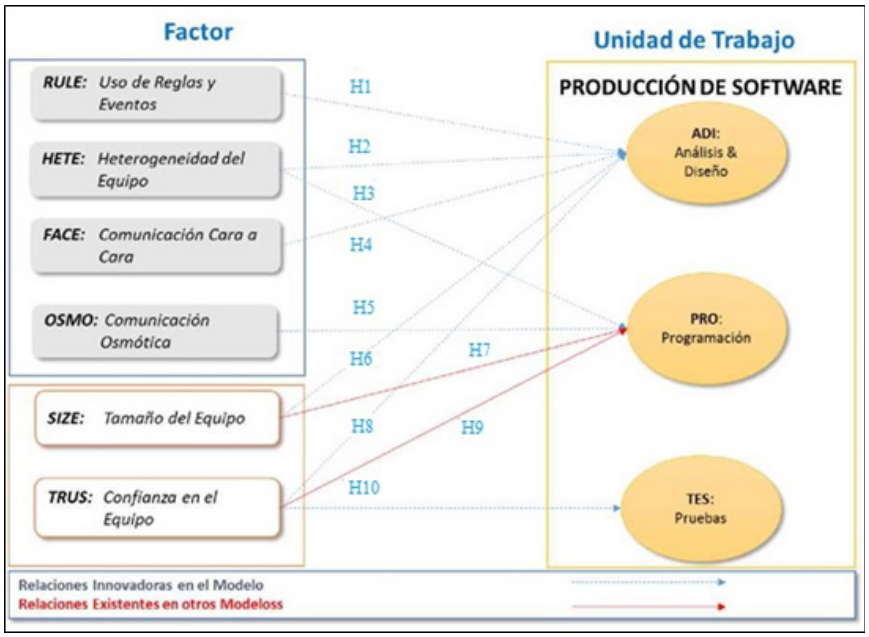
Un equipo de trabajo exitoso tiene sus cimientos en la base de la confianza. Cada miembro del equipo debe establecerla a través de sus acciones, sus palabras y su trabajo para mantenerla. Cada miembro también necesita ser capaz de confiar en los miembros de su equipo para hacer un compromiso con ellos y sus objetivos. Una síntesis de los factores introducidos se presenta en la Tabla 16.

Tabla 16. Nuevos factores que impactan en la productividad

ID	Factor	Descripción	Soporte
RULE	Uso de reglas y eventos	Cumplimiento de normas y reglas de comportamiento que son definidas por el mismo equipo.	Metodologías Ágiles (SBOK, 2016)
FACE	Tipo de comunicación "cara a cara" en el equipo	Método de comunicación que permite que las personas interactúen para mejorar el entendimiento de los requerimientos.	Metodologías Ágiles (SBOK, 2016); Perspectiva de la Acción del Lenguaje (Flores y Ludlow, 1980; Simmons, 1991)
OSMO	Tipo de comunicación "osmótica" en el equipo	Aprendizaje a través de información que fluye en el fondo auditivo de los miembros del equipo, de tal forma que ellos captan la información relevante.	Teoría de la memoria transactiva (Wegner, 1987)
HETE	Heterogeneidad del equipo	Formación de equipos de trabajo multifuncionales.	Ancona y Caldwell (1992); Cohen y Bailey (1997)
SIZE	Tamaño del equipo	n.º de participantes en el equipo de trabajo.	Rodríguez *et al.* (2011); Verner *et al.* (2007); Brooks (1995); Abdel-Hamid y Madnick (1991); Abdel-Hamid (1989); Montgomery (1981); Smith *et al.* (2001); Trendowicz y Münch (2009); Norden (1958, 1977); Pillai y Sukumaran (1997); Harman y Jones (2001); Di Penta *et al.* (2007); Alba y Chicano (2005, 2007); Guerguiev *et al.* (2009); Chicano *et al.* (2011); Kremmel *et al.* (2010); Kremmel *et al.* (2011); Pendharkar y Rodger (2007)
TRUS	Confianza en el equipo	Capacidad de los miembros del equipo para confiar entre ellos.	Yilmaz y O'Connor (2011); Sridhar *et al.* (2007); Sudhakar *et al.* (2011)

Luego de revisar los diferentes factores, y con base en los criterios establecidos, en la Figura 10 se puede visualizar el modelo conceptual de los factores propuestos y su incidencia en la productividad en las diferentes unidades de trabajo de la fábrica de *software*.

Figura 10.
Modelo conceptual inicial de los factores que afectan la productividad



4.7.3. Hipótesis

Como se mencionó, para la realización de esta propuesta, se consideraron algunas hipótesis:

- Hipótesis 1 (H1): El factor uso de reglas y eventos influye en la unidad de trabajo de análisis y diseño.
- Hipótesis 2 (H2): El factor heterogeneidad del equipo influye en la unidad de trabajo de análisis y diseño.
- Hipótesis 3 (H3): El factor heterogeneidad del equipo influye en la unidad de trabajo de programación.
- Hipótesis 4 (H4): El factor tipo de comunicación "cara a cara" influye en la unidad de trabajo de análisis y diseño.
- Hipótesis 5 (H5): El factor tipo de comunicación "osmótica" influye en la unidad de trabajo de programación.
- Hipótesis 6 (H6): El factor tamaño del equipo influye en la unidad de trabajo de análisis y diseño.
- Hipótesis 7 (H7): El factor tamaño del equipo influye en la unidad de trabajo de programación.
- Hipótesis 8 (H8): El factor confianza en el equipo influye en la unidad de trabajo de análisis y diseño.
- Hipótesis 9 (H9): El factor confianza en el equipo influye en la unidad de trabajo de programación.
- Hipótesis 10 (H10): El factor confianza en el equipo influye en la unidad de trabajo de pruebas.

Asimismo, como se observa en la Tabla 17, se analizó aquellos hipótesis que se consideraron como factores que podrían afectar la productividad en el desarrollo de *software*.

Tabla 17. Hipótesis planteadas para la definición de factores que afectan la productividad en el desarrollo de *software*

Factor	Unidad de trabajo		
	Análisis y diseño (ADI)	Programación (PRO)	Pruebas (TES)
RULE	H1		
HETE	H2	H3	
FACE	H4		
OSMO		H5	
SIZE	H6	H7	
TRUS	H8	H9	H10

a. Prueba de hipótesis: Regresión Lineal Múltiple

Para el análisis de los resultados, se tuvo en cuenta que si la significación de t-test es menor que 0.05, implicaba que la variable independiente se relaciona de forma significativa con la variable dependiente, por tanto, influye sobre ella y es explicativa (Ramanathan, 2002).

b. Componente: ADI-Análisis y Diseño de *software*

Mediante la técnica de Regresión Lineal Múltiple, se explicará el nivel de significancia entre la variable dependiente (ADI) y las seis variables independientes (RULE, HETE, FACE, OSMO, SIZE, TRUS), a fin de determinar cuáles son las que tienen mayor impacto en la productividad de dicha componente.

Tabla 18. Regresión de la variable dependiente (ADI) con todas las variables independientes

Factor	B	Error estándar	t	Sig.
RULE	0.154	0.069	2.240	0.027
HETE	-0.141	0.061	-2.312	0.022
FACE	0.068	0.087	0.781	0.436
OSMO	-0.005	0.059	-0.087	0.931
SIZE	-0.056	0.072	-0.778	0.438
TRUS	0.235	0.078	3.009	0.003

Tal como se muestra en la Tabla 18, las variables más significantes son las siguientes: RULE, HETE y TRUS. Además, para tener un modelo predictivo, y teniendo en consideración estas variables significativas, se realiza la regresión lineal, de esta manera, se obtienen los resultados de la Tabla 19.

Tabla 19. Regresión de la variable dependiente (ADI) con las variables independientes significativas

Factor	B	Error estándar	t	Sig.
RULE	0.163	0.054	2.991	0.003
HETE	-0.129	0.053	-2.436	0.016
TRUS	0.243	0.049	5.002	0.000

El modelo predictivo se muestra a continuación:

$$ADI = 0.163\ RULE - 0.129\ HETE + 0.243\ TRUS$$

Interpretando el modelo se puede decir que cada vez que el factor RULE se incremente en un punto (intensidad), ADI se incrementará en 0.163; es decir, el mayor uso de reglas y eventos conllevará a un mejor desempeño de las actividades del equipo. Asimismo, al incrementarse en una unidad el nivel de influencia por efecto de la heterogeneidad del equipo (HETE), entonces, el impacto sobre el componente de análisis y diseño decrece en 0.129 unidades; es decir, a menos heterogeneidad en el equipo, se genera mayor eficiencia en los resultados.

De igual manera, cada vez que el factor TRUS se incremente en un punto (intensidad), ADI se incrementará en 0.243; es decir, una mayor confianza en el equipo conllevará a un mejor desempeño en el logro de resultados.

c. Componente: PRO-Programación de *software*

Mediante la técnica de Regresión Lineal Múltiple, se explicará el nivel de significancia entre la variable dependiente (PRO) y las seis variables independientes (RULE, HETE, FACE, OSMO, SIZE, TRUS), a fin de determinar cuáles son las que tienen mayor impacto en la productividad en el componente de programación de *software*.

Tabla 20. Regresión de la variable dependiente (PRO) con todas las variables independientes

Factor	B	Error estándar	t	Sig.
RULE	-0.048	0.052	-0.923	0.357
HETE	0.100	0.048	2.062	0.041
FACE	0.061	0.047	1.301	0.195
OSMO	0.120	0.051	2.182	0.032
SIZE	0.125	0.056	2.225	0.028
TRUS	0.170	0.058	2.932	0.004

Como se observa en la Tabla 20, las variables más significantes son HETE, OSMO, SIZE y TRUS. Asimismo, para tener un modelo predictivo, y teniendo en consideración estas variables significativas, se realiza la regresión lineal, a partir de la cual se obtienen los resultados de la Tabla 21.

Tabla 21. Regresión de la variable dependiente (PRO) con las variables independientes significativas

Factor	B	Error estándar	t	Sig.
HETE	0.076	0.040	1.894	0.060
OSMO	0.115	0.042	2.282	0.001
SIZE	0.147	0.045	3.282	0.001
TRUS	0.157	0.046	3.418	0.001

El modelo predictivo se muestra a continuación:

PRO = 0.076 HETE + 0.115 OSMO + 0.147 SIZE + 0.157 TRUS

Tal como se puede visualizar, las cuatro variables influyen positivamente en la productividad de la unidad de trabajo de programación.

d. Componente: TES-Pruebas de Software

Mediante la técnica de Regresión Lineal Múltiple, se explicará el nivel de significancia entre la variable dependiente (TES) y las seis variables independientes (RULE, HETE, FACE, OSMO, SIZE, TRUS), a fin de determinar cuáles son las variables que tienen mayor impacto en la productividad en el componente de pruebas de *software*.

Tabla 22. Regresión de la variable dependiente (TES) con todas las variables independientes

Factor	B	Error estándar	t	Sig.
RULE	0.039	0.034	1.151	0.252
HETE	0.040	0.035	1.159	0.248
FACE	-0.006	0.034	-0.174	0.862
OSMO	-0.022	0.033	-0.678	0.499
SIZE	-0.004	0.034	-0.125	0.901
TRUS	-0.085	0.037	-2.271	0.025

Según los datos de la Tabla 22, la variable más significante es TRUS. Asimismo, para tener un modelo predictivo, y teniendo en consideración estas variables significativas, se realiza la regresión lineal, así, se obtienen los resultados de la Tabla 23.

Tabla 23. Regresión de la variable dependiente (TES) con las variables independientes significativas

Factor	B	Error estándar	t	Sig.
TRUS	0.044	0.007	6.071	0.000

El modelo predictivo se muestra a continuación:

$$TES = 0.044\ TRUS$$

Como se observa, cada vez que el factor TRUS se incremente en un punto (intensidad), TES se incrementará en 0.044; es decir, una mayor confianza en el equipo conllevará a un mejor desempeño en el logro de resultados.

4.8. Discusión

De acuerdo a la literatura, existen diversas publicaciones sobre productividad de *software* que se han estado revisando en los últimos años, lo cual puede ser justificado por el interés que los investigadores han dado al tema de la productividad en la industria del *software*. Sin embargo, es muy escasa o casi nula la literatura acerca de la medición de la productividad en fábricas de *software*, cuya utilización del modelo ha traído grandes ventajas, tal como lo demuestra el estudio de Clements y Northrop (2001).

Si bien existen modelos de medición de la productividad del *software* (Asmild *et al.*, 2006; Cheikhi *et al.*, 2012; Machek *et al.*, 2012; Moreira *et al.*, 2010; Nwelih y Amadin, 2008; Yilmaz y O'Connor, 2011), la mayoría ha considerado solo la unidad de trabajo de programación, lo cual conlleva a un sesgo en la medición, que trae como consecuencia que no se cuente con indicadores de medición que faciliten la toma de decisiones en la organización.

En ese sentido, los estudios han sido orientados en dos vertientes principales: técnicas o modelos que miden la productividad y los factores que la afectan. Estos estudios representan el proceso de desarrollo como unidad y dificultan la medición si se quisiera realizar de manera independiente por cada uno de los componentes que participan en el ciclo de vida de desarrollo del *software*. Además, no se han encontrado evidencias de modelos que midan la productividad en todas las fases de este proceso de desarrollo. Asimismo, no existe un consenso de los autores sobre los factores que generan mayor impacto en la productividad y la unidad de medida a tener en cuenta para la medición de la misma.

Del total de documentos, más del 70 % están enfocados en los factores que afectan la productividad. A partir de ello, se puede inferir la importancia que han dado los investigadores al aspecto factor, de esta manera, se busca explicar el impacto que tiene este en la medición. Existen pocos estudios (27.41 %) enfocados a determinar modelos que midan la productividad en las diferentes unidades de la fábrica de *software*; por ello, se han planteado algunas respuestas a las interrogantes de esta investigación.

La primera interrogante que se planteó fue ¿qué es la productividad de fábricas de software? Si bien los estudios explican la productividad de *software* y, a la vez, se enfocan en la unidad de trabajo de programación; la información acerca de la productividad en la fábrica de *software* como un todo es casi nula.

De igual manera, en los estudios se habla muy poco de las fábricas de *software* a pesar de ser un modelo que se viene trabajando en la industria hace muchos años y, por ende, no se tiene una definición clara de la productividad.

Por lo tanto, si la medición de la productividad es muy importante para el desarrollo de una organización, esta se hace compleja si no se tiene una definición clara. En este contexto, es necesario definir la productividad de las fábricas de *software* como un indicativo de cómo los recursos utilizados para la producción del producto final son eficientes en las diferentes unidades de trabajo de la fábrica de *software*. A partir de dicho concepto, la productividad exige una buena gestión de los recursos asignados a fin de conseguir resultados que vuelvan eficientes todas las labores desarrolladas dentro de la organización no solo respecto a la producción del *software*, sino también a los métodos utilizados y la relación interna de la organización.

Otra de las interrogantes planteada fue ¿qué factores influyen en la medición de la productividad y cómo se clasifican? En la investigación, se han identificado 74 factores, los cuales han sido agrupados de acuerdo a las categorías propuestas por Wagner y Ruhe (2008a), ya que contemplan algunas que en muchos casos no han sido cubiertas por Trendowicz (2007), tales como entorno de desarrollo, cultura corporativa, cultura del equipo y entorno. Sin embargo, estas categorías son necesarias, ya que agrupan factores transversales a toda la organización y no son exclusivas del producto o proyecto.

Por otro lado, respecto a la interrogante ¿qué modelos se han desarrollado?, se han identificado diez modelos de medición de la productividad de *software* que están basados en DEA (Asmild et al., 2006; Cao *et al.*, 2012; Pai *et al.*, 2015), CMMI y Six Sigma (Moreira *et al.*, 2010), Lógica difusa (López *et al.*, 2010), modelo de ecuaciones estructuradas (Yilmaz y O'Connor, 2011), reutilización de software (Nwelih *et al.*, 2008), factor total de productividad (Machek *et al.*, 2012), ISO 9126-4 IEEE Std.1045 (Cheikhi *et al.*, 2012) y otras técnicas estadísticas planteadas por Unluturk y Kurtel (2015). Si bien se definen cómo los modelos de medición de la productividad, muchos de ellos plantean una función; pero no derivan a una fórmula (Cheikhi *et al.*, 2012; Machek *et al.*, 2012; Yilmaz y O'Connor, 2011), solo mencionan diversos factores para considerar en la medición.

Estos modelos solo han sido diseñados para realizar una medición de la productividad interna, pero, a pesar de ello, son incompletos, ya que se han orientado en su mayoría para la unidad de trabajo de programación (Asmild *et al.*, 2006; Cheikhi *et al.*, 2012; Machek *et al.*, 2012; Moreira et al., 2010; Nwelih y Amadin, 2008; Yilmaz y O'Connor, 2011), y muy pocos para pruebas y análisis y diseño (Cao *et al.*, 2012; Moreira *et al.*, 2010; Pai *et al.*, 2015; Unluturk y Kurtel, 2015), lo cual genera un sesgo en la medición, ya que no permitiría establecer la productividad en los otros componentes de la fábrica de *software*.

Asimismo, los modelos no miden la productividad externa; es decir, no realizan una evaluación comparativa de la productividad, lo cual podría traer consecuencias negativas para la organización, ya que, si solo se compara de manera interna, tendría una información sesgada de cómo se está comportando el mercado.

Algunos de los modelos, como los de Asmild et al. (2006) y Moreira *et al.* (2010), se han validado en un entorno real; sin embargo, la información utilizada para su diseño se obtuvo de la misma organización, lo cual dificulta su generalización si se quisiera comparar con otras organizaciones. También, se plantea un modelo cualitativo (Cheikhi et al., 2012; Machek *et al.*, 2012; Yilmaz y O'Connor, 2011) que, si bien se podría considerar la precisión del modelo, esto no podría generalizarse

debido a que los modelos no presentan una aplicación práctica o no han sido aplicados en casos de estudio.

Finalmente, la última pregunta planteada fue ¿qué factores incluyen los modelos? Respecto a ello, los factores que más se han utilizado en los modelos son los siguientes: tamaño de equipo, reutilización y complejidad. Sin embargo, existen muchos otros que no han sido tomados en cuenta en los modelos, tales como números totales de lanzamientos, problemas, riesgos, solicitud de cambio, ocurrencia de espera, duración de espera, motivo de espera.

Si se toma en cuenta los factores que existen en la literatura y los modelos que se han elaborado hasta la fecha, se observa que no existe una homogeneización de los factores por considerar en un modelo y, en muchos casos, no se han considerado algunos de los factores, por ello, se ha afectado la precisión del modelo, ya que se considera que cuántos más factores cubra el modelo, se podrá establecer más aspectos de la realidad.

Tal como se ha podido observar en los diferentes estudios, no existe un estándar acerca de cómo se debe organizar una fábrica de *software*, lo cual dificulta el análisis de la medición de la productividad. Normalmente, los estudios delimitan la fábrica de *software* solo a la producción de *software*, pero ello tiene varias desventajas, pues no se contemplaría la gestión de proyecto y otras unidades que interactúan en ella, lo que podría sesgar la interpretación de la productividad.

Por otra parte, se identificó que una fábrica de *software* se organiza en tres grandes componentes: gestión del proyecto, producción de *software* y soporte. Asimismo, estos pueden estar divididos en unidades de trabajo (servicio, gestores de proyecto, planificación y control de la producción, modelo de negocio, análisis y diseño, programación, pruebas, procesos, gestión de indicadores, gestión del conocimiento, arquitectura, metodologías, técnicas y herramientas); es decir, las áreas en las cuales se realizan las actividades necesarias para el cumplimiento del objetivo del componente.

De igual manera, se ha podido determinar que en Perú las fábricas de *software* se han implementado en su mayor parte en entidades públicas (60 %). También, se puede apreciar que el enfoque de la fábrica de *software* en el Perú es muy reciente, tal como lo corrobora el 65.3 % de los participantes, quienes cuentan en promedio con cinco años de experiencia.

Por lo tanto, a partir de los datos obtenidos en este estudio, se ha evidenciado que el factor uso de reglas y eventos (RULE) tiene una influencia significativa positiva en la unidad de trabajo de análisis y diseño, debido a que el analista requiere establecer una adecuada planificación de las actividades a realizar y estas deben estar alineadas al alcance del requerimiento, por lo que el analista no debería distraerse en actividades accesorias que impidan el cumplimiento del objetivo. Por ello, es de mucha importancia establecer reglas que den soporte y precisen estas actividades.

Asimismo, el factor heterogeneidad del equipo (HETE) tiene influencia negativa en la unidad de trabajo de análisis y diseño; sin embargo, es significativamente positivo en la programación, es decir, para las tareas de análisis en general se requiere especialistas en tipos de negocios, por lo que involucrar más especialistas reduciría la productividad. No obstante, en las tareas de programación, se requiere diferentes habilidades y capacidades para el desarrollo de un producto.

El factor tipo de comunicación "cara a cara" en equipo (FACE) no tiene un impacto significativo en la unidad de trabajo de análisis y diseño, por lo que se descarta la hipótesis inicial que había sido sustentada con base en la teoría de la perspectiva de la acción del lenguaje (Flores y Ludlow, 1980).

Ello se podría explicar por el incremento del uso de los medios tecnológicos para la interacción, la cual en el desarrollo de software es muy importante para el logro de los objetivos, pero la versión presencial está siendo superada por los medios virtuales.

El factor tipo de comunicación osmótica en equipo (OSMO) tiene una influencia significativa positiva en la unidad de trabajo de programación, debido a que las actividades de programación requieren diferentes habilidades y capacidades, lo cual genera que, mientras más personas tengan conocimiento de lo que están haciendo los demás integrantes del equipo, más fácil será la integración del producto y mayor contingencia se tendrá en caso de ausencia de algún miembro del equipo, por lo que conllevará a una mayor productividad.

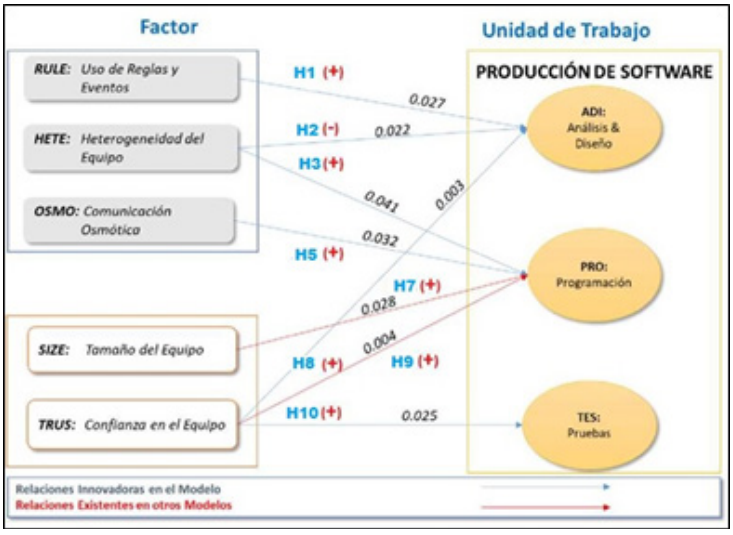
Por otra parte, se corrobora el impacto del factor tamaño del equipo (SIZE) en la unidad de trabajo de programación (Abdel-Hamid y Madnick, 1991; Abdel-Hamid, 1989; Alba y Chicano, 2005, 2007; Brooks, 1995; Chicano *et al.*, 2011; Di Penta *et al.*, 2007; Gueorguiev *et al.*, 2009; Harman y Jones, 2001; Kremmel *et al.*, 2010, 2011; Montgomery, 1981; Norden, 1958, 1977; Pendharkar y Rodger, 2007; Pillai y Sukumaran, 1997; Smith *et al.*, 2001; Trendowicz y Munch, 2009; Verner *et al.*, 2007). Sin embargo, no hay significancia de este factor sobre la unidad de trabajo de análisis y diseño.

El factor confianza en el equipo (TRUS) tiene una influencia significativa positiva en las tres unidades de trabajo, lo cual refuerza la necesidad de diseñar estrategias que mejoren la confianza entre los miembros del equipo, con el fin de alcanzar el objetivo, que es la entrega del producto final, la cual se da a través del trabajo integrado entre todas las unidades de las fábricas.

Si bien los otros factores analizados tienen un nivel de influencia positiva o negativa en la productividad de la fábrica de *software*, los valores obtenidos no son lo suficientemente significativos para ser considerados. Esto implica que, a pesar de que estos factores sean gestionados correcta o incorrectamente, su influencia en la productividad no va a generar un alto impacto. Por ejemplo, el uso del tipo de comunicación osmótica en el equipo (OSMO) genera una mejora en la productividad en la unidad de trabajo de análisis y diseño; pero esta influencia no será muy significativa.

Finalmente, en la Figura 11 se puede visualizar el modelo conceptual final con los factores que tienen un mayor impacto en cada una de las unidades de trabajo de la fábrica de *software*.

Figura 11. Modelo final de factores que afectan la productividad



CAPÍTULO V

MODELO DEA PROPUESTO PARA LA MEDICIÓN DE LA PRODUCTIVIDAD DE LA FÁBRICA DE SOFTWARE

La productividad es definida como la razón entre una salida que se produce y la entrada que se usa para producirla (Coelli *et al.*, 2005). Asimismo, no hay un estándar en la definición de la productividad; sin embargo, los autores tienen un aspecto común, ya que consideran que la productividad es la ratio del producto obtenido entre el esfuerzo. Por otro lado, los estudios realizados sobre la productividad de una fábrica de *software* se orientan a medir la productividad en la unidad de trabajo de programación de *software*, tal como se puede apreciar en las entradas y salidas que han sido evaluadas (SLOC, FP, PH, PM).

Esto ha conllevado a que se desarrollen muchos modelos de medición del desempeño que han llegado a ser inexactos. No obstante, los estudios no han revisado de manera integral a la fábrica de *software* y han obviado otras unidades de trabajo también importantes como el análisis y diseño, y las pruebas.

En este contexto, se define la productividad de las fábricas de *software* como un indicador de la eficiencia de los recursos utilizados en diferentes unidades de trabajo de estas fábricas para la consecución del producto final. Se ha planteado esta definición porque considera todas las unidades de trabajo que participan en el desarrollo de *software* y, además, permite evaluar los diferentes factores que influyen en todo el ciclo de su producción.

Cardoso *et al.* (2010) afirman que un modelo es una representación simplificada de la realidad y se define a través de los diferentes componentes y procesos que forman parte del sistema en estudio. En ese sentido, se han identificado trece modelos que miden la productividad en el desarrollo de *software*, los cuales están basados en los siguientes enfoques: DEA, CMMI & Six Sigma, Reutilización de *software*, SEM, Factor total de productividad, IEEE Std. 1045 & ISO 9216-4.

5.1. Modelos de productividad DEA

DEA es una técnica de programación matemática ampliamente utilizada que fue desarrollada por Charnes *et al.* (1978) y extendida por Banker *et al.* (1984) para incluir retornos de escala variable. Este modelo generaliza la medida de eficiencia técnica de Farrell (1957) para evaluar la eficiencia relativa de unidades pares con respecto a múltiples medidas de rendimiento (Charnes *et al.*, 1994; Cooper *et al.*, 2006).

Las unidades bajo evaluación en DEA se llaman toma de decisiones (DMU). Una de estas se considera eficiente cuando ninguna otra unidad puede producir más salidas usando una cantidad igual o menor de entradas. Por lo tanto, la DEA es una técnica de medición de la eficiencia basada en la obtención de una frontera de eficiencia a partir del conjunto de observaciones que se considere sin la estimación de ninguna función de producción (Charnes *et al.*, 1994).

Por otro lado, a diferencia de los métodos paramétricos que requieren un conocimiento detallado del proceso, DEA no es paramétrico y no requiere una forma funcional explícita relacionada con las entradas y salidas (Cook y Seiford, 2009; Cooper *et al.*, 2006). Desde su introducción, la investigación en DEA ha ido evolucionando, tanto en el ámbito teórico como en la práctica, y se ha encontrado trabajos en diversas áreas, tales como la medición del desempeño de instituciones educativas, el

benchmarking de procesos logísticos, la comparación de sucursales de oficinas regionales de bancos, la regulación de servicios públicos, la medición de productividad investigativa y docente en departamentos académicos, así como en la industria del software, como se puede apreciar en el resumen de la Tabla 24 en el cual se actualiza la revisión de literatura elaborada por Pai *et al.* (2015).

Tabla 24. Estudios de proyectos de software aplicando DEA

Id	Entrada	Salida	Modelo	Unidad de trabajo	Tamaño de Muestra	Referencia
1	Esfuerzo	Puntos de función	DEA Constant Returns to Scale	Programación	8 conjunto de datos, cada uno con 15 – 63 proyectos	Banker y Kemerer (1989)
2	Esfuerzo	Puntos de función, calidad	Stochastic DEA	Programación	65 proyectos	Banker *et al.* (1991)
3	Esfuerzo	Puntos de función	DEA-based F-tests	Programación	11 conjunto de datos, cada uno con 15 - 63 proyectos	Banker *et al.* (1994)
4	Esfuerzo	Comment Change, Change, Routines Change	Additive DEA Code Constant Returns to Scale	Programación	78 proyectos	Mahmood *et al.* (1996)
5	Costo del proyecto	Tamaño (puntos de duración, tiempo de comercialización en días)		(Programación-pruebas)	11 proyectos	Paradi *et al.* (1997)
6	Esfuerzo	Usuarios, EDI, conversiones	DEA Variable Returns to Scale	Soporte	30 proyectos COTS (por las siglas del inglés Commercial-off-the-shelf)	Myrtveit y Stensrud (1999)
7	Esfuerzo	Puntos de función, ubicaciones, unidades de negocio, usuarios concurrentes	DEA Variable Returns to Scale	(Programación-soporte)	114 proyectos	Flitman (2003)
8	Esfuerzo	Usuarios, EDI, conversiones	DEA Variable Returns to Scale	Soporte	30 proyectos Enterprise Resource Planning (ERP)	Stensrud y Myrtveit (2003)
9	Esfuerzo, costos del vendedor, duración	Líneas de código fuente		Programación	70 proyectos	Yang y Paradi (2004)
10	Esfuerzo	Puntos de función		Programación	144 proyectos	Asmild *et al.* (2006)
11	Esfuerzo	Puntos de función, Líneas de código fuente	DEA & Regression Analysis	Soporte	10 proyectos ERP (Enterprise Resource Planning)	Parthasarathy y Anbazhagan (2008)
12	Desarrolladores, incidencias	Kilobytes por descarga, descargas, rango del esfuerzo del proyecto, puntos de función como una variable contextual		Programación	34 proyectos	Wray y Mathieu (2008)
13	Esfuerzo, tamaño del equipo	Tamaño (puntos de función)	DEA Variable Returns to Scale	Programación	540 proyectos	Pendharkar y Rodger (2009)
14	Horas de labor variable	Métricas de complejidad UML	DEA Variable Returns to Scale	(Análisis y diseño)	25 proyectos	Cao *et al.* (2012)
15	Esfuerzo de desarrollo (DE), esfuerzo de revisión y pruebas (EoC), y esfuerzo de retrabajo (EoRC)	Puntos de función, calidad (f(defectos))	DEA Variable Returns to Scale	(Programación-pruebas)	79 proyectos	Pai *et al.* (2015)

A partir de los datos presentados en la tabla anterior, se aprecia que la mayoría de los trabajos que evalúan la productividad en proyectos de desarrollo de *software* usando DEA consideran el modelo VRS y solo la unidad de trabajo de programación, lo cual sesga la medición de la productividad, ya que para el desarrollo de *software* se involucran otras unidades de trabajo muy importantes como el análisis y diseño, y las pruebas.

Por ello, se consideró utilizar la metodología DEA porque desarrolla un proceso de optimización para cada observación individual con el objetivo de calcular una frontera discreta determinada por las DMU eficientes. Además, no requiere supuesto alguno sobre la forma funcional de la frontera; las DMU ineficientes se proyectan sobre la frontera mediante una combinación convexa de las eficientes más próximas y permite identificar las fuentes y el nivel de ineficiencia de los *inputs y outputs*.

5.2. Metodología

El propósito de la DEA es construir una frontera no paramétrica por tramos sobre el conjunto de datos, de manera que todos los puntos observados se encuentren en la frontera de producción o debajo de ella para calcular las eficiencias relativas de todas las DMU con respecto a la frontera. Cada DMU no posicionada en la frontera se reduce a una combinación convexa de DMU en la faceta fronteriza más cercana (Charnes *et al.*, 1978). Los diferentes aspectos que se han tenido en cuenta para la definición del modelo de la DEA y los datos que se deben considerar para su evaluación se analizan con mayor detalle en la subsección siguiente.

5.2.1. Elección del modelo de la DEA

Para la elección del modelo DEA se tomaron algunas consideraciones, entre las cuales la primera fue la isotonicidad. Esta es una regla fundamental para los parámetros DEA, se trata de un aumento en una variable de entrada que debe mejorar cada una de las salidas (Chung *et al.*, 2008), a esto se le denomina propiedad de isotonicidad de parámetros DEA. Se debe realizar un análisis de correlación para asegurar las relaciones positivas entre las entradas y las salidas. En caso de correlaciones negativas, se pueden aplicar técnicas de transformaciones.

Sin embargo, varios problemas relacionados con la transformación de variables para satisfacer la propiedad de isotonicidad los han descrito Liu *et al.* (2010). Por lo tanto, en lugar de conducir a la transformación de parámetros, las correlaciones negativas pueden ser una indicación de que uno o más de los parámetros deben ser excluidos del modelo.

Para el presente caso de estudio se consideraron dos variables de entrada (esfuerzo, costo del proyecto) y una de salida (puntos de función), que ya han sido utilizadas en estudios anteriores (Asmild *et al.*, 2006; Banker y Kemerer, 1989; Banker *et al.*, 1991, 1994; Flitman, 2003; Mahmood *et al.*, 1996; Myrtveit y Stensrud, 1999; Pai *et al.*, 2015; Paradi *et al.*, 1997; Parthasarathy y Anbazhagan, 2008; Pendharkar y Rodger, 2009; Stensrud y Myrtveit, 2003; Yang y Paradi, 2004), por lo que no es necesario establecer la isotonicidad de los parámetros.

Otra consideración fue la orientación del modelo DEA. Se sabe que este proporciona dos básicas orientaciones: maximización de la salida y minimización de la entrada. Un modelo orientado a maximizar la salida determina el máximo aumento proporcional de las salidas en relación con los valores de entrada, que es adecuado para establecer un conjunto de valores de salida objetivo. Por otro lado, un modelo orientado a minimizar la entrada determina la cantidad por la cual los valores de entrada se pueden disminuir mientras se siguen produciendo las mismas salidas. Para esta

investigación se decidió adoptar un modelo orientado a las entradas para determinar la eficiencia basada en el uso de parámetros de entrada para producir los valores de calidad y productividad resultantes.

Por último, se consideró la selección del modelo DEA. La importancia de seleccionar adecuadamente qué modelo se utilizará se debe a que mediante DEA se buscará determinar si se puede crear una fábrica de *software* ideal (virtual) que sea mejor que una o más de las fábricas de *software* reales. Cualquier fábrica dominada será una fábrica ineficiente.

Como resultado del proceso de selección, para el caso del presente estudio, se tomó en cuenta el modelo DEA retornos de escala variable (DEA VRS), ya que se consideró que algunas fábricas de *software* de tamaño diferente al de las eficientes pueden no ser capaces de conseguir la productividad de estas. Por ello, el estudio se realizó mediante la eficiencia técnica, que toma en cuenta que cada DMU debe tomar como referencia a la productividad mayor de entre las de su tamaño.

Debido a que los contratos establecidos con las fábricas de *software* implican la entrega de productos definidos al inicio del servicio, se consideró la orientación de entrada por el hecho de que las fábricas de *software* deben alcanzar la productividad de la unidad de referencia a costa de reducir las horas de esfuerzo y el costo del proyecto real. Adicionalmente, se considera que las variables de entrada pueden ser controladas.

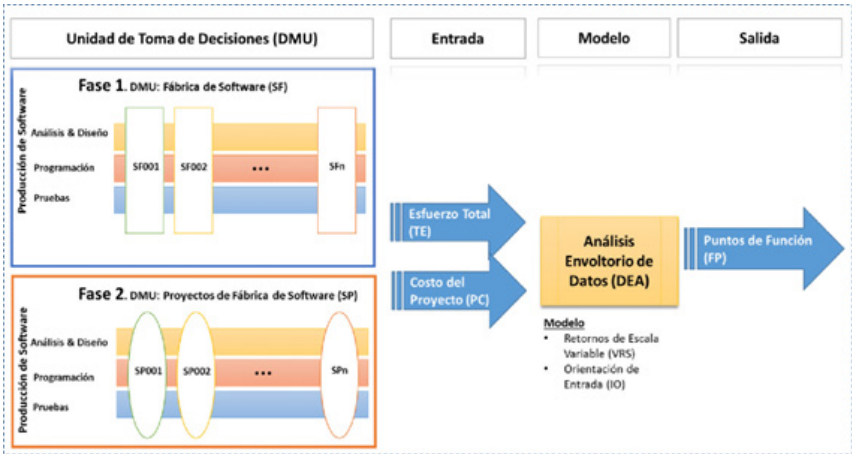
5.2.2. Parámetros para el modelo DEA

La Figura 12 muestra los parámetros que componen el modelo DEA: DMU, entrada, modelo, salida, fases. Por otra parte, la fábrica fue evaluada en dos fases: la fase 1 buscaba determinar la productividad de las fábricas de *software*, con el fin de conocer la más eficiente y, también, la brecha existente y las buenas prácticas que se iban aplicando, de modo que sean replicadas a las otras fábricas de *software* y conlleven a una mejora de su eficiencia.

Así también, la fase 2 determinaría la productividad de los proyectos de fábrica de *software*, para obtener buenas prácticas o lecciones aprendidas de los proyectos más eficientes y se conviertan en modelos a seguir por los proyectos que están en su categoría y que podrían lograr la eficiencia de estos.

Figura 12.

Parámetros para el modelo DEA



A continuación, se describen los parámetros del modelo DEA.

a. Unidad de toma de decisiones (DMU)

Se define como cualquier organización que genera productos o servicios que consumen ciertos recursos, con la capacidad de modificar tanto los recursos consumidos (entradas) como la producción creada (salidas). Para el caso planteado, se consideró la evaluación en dos fases: la fase 1, DMU-Fábrica de *software*, considera la información de las diferentes organizaciones que vienen trabajando en el modelo de fábrica de *software*. Por otro lado, la fase 2, DMU-Proyectos de Fábrica de *software*, considera la información de los proyectos desarrollados dentro de cada fábrica. Para ambas fases, se recolectó la información de las actividades que se ejecutaron dentro de la componente de producción de *software*, así, se tuvo en cuenta solo las siguientes unidades de trabajo: análisis y diseño, programación, pruebas.

b. Entrada

Es la variable de entrada que fue evaluada por el modelo DEA. En el caso planteado se consideró dos variables para las dos fases, que han sido ampliamente estudiadas para la evaluación de la eficiencia en proyectos de desarrollo de *software*. La primera fase es el esfuerzo total (TE), que es el trabajo medido en horas (h) utilizado en las fábricas de *software* y en sus respectivos proyectos, donde se ha considerado el esfuerzo de las unidades de trabajo de análisis y diseño, programación, pruebas.

Esta variable ha sido ampliamente utilizada en los estudios de Banker y Kemerer (1989), Banker *et al.* (1991), Banker *et al.* (1994), Mahmood *et al.* (1996), Myrtveit y Stensrud (1999), Flitman (2003), Stensrud y Myrtveit (2003), Yang y Paradi (2004), Asmild *et al.* (2006), Parthasarathy y Anbazhagan (2008), Pendharkar y Rodger (2009) y Pai *et al.* (2015).

La segunda fase es el costo del proyecto (PC) que es expresado en dólares americanos (USD) e incluye los costos incurridos en las fábricas de *software* y sus proyectos a través de las unidades de trabajo de producción de *software*. Esta variable ha sido analizada y utilizada en Paradi *et al.* (1997).

c. Salida

Es la variable de salida que será evaluada por el modelo DEA. En el caso planteado se ha considerado la variable puntos de función (FP), que representa al tamaño del *software*, por ser la unidad de medida que se utiliza en las diferentes fábricas de *software* y sus proyectos. Esta se ha obtenido a través de los equipos de estimación (responsables de contabilizar los FP al final del proyecto).

El proceso de desarrollo especifica que los FP son estimados al inicio y final del proyecto para evidenciar los cambios en este. Esta variable ha sido ampliamente utilizada en los estudios de Banker y Kemerer (1989), Banker *et al.* (1991), Banker *et al.* (1994), Flitman (2003), Asmild *et al.* (2006), Parthasarathy y Anbazhagan (2008), Pendharkar y Rodger (2009) y Pai *et al.* (2015).

d. Modelo

Para el estudio se utilizará el modelo DEA VRS-I (Banker *et al.*, 1984), ya que se busca minimizar las entradas (esfuerzo total y costo del proyecto), así como mantener las salidas actuales (número de puntos de función) de las fábricas de *software* y sus respectivos proyectos.

e. Fases

Se ha considerado identificar las eficiencias en dos fases: fase 1, fábrica de *software*; fase 2, proyectos de fábrica de *software*.

- Fase 1. En esta fase se busca identificar las fábricas de software más eficientes a través de la minimización del esfuerzo total y el costo del proyecto consumidos en un período determinado por las fábricas de software, pero se debe mantener la cantidad de puntos de función, por lo que se utiliza DEA VRS-I, donde las entradas son minimizadas y las salidas se mantienen en su nivel actual. Este modelo asume dos entradas, una salida y ene DMU's, tal como se muestra a continuación:

$$Minimizar\ \theta$$

$$\sum_{j=1}^{6} \lambda_j * x_{ij} \leq \theta\, x_{i0} \quad i = 1, 2, \ldots, n;$$

$$\sum_{j=1}^{6} \lambda_j * y_j \geq y_0;$$

$$\sum_{j=1}^{6} \lambda_j = 1;$$

$$\lambda_j \geq 0 \ \ j = 1, 2, \ldots, n;$$

Donde θ representa la medida de productividad o eficiencia de la fábrica de *software*; x_{ij}, la *i*th entrada (x_{ij} es el esfuerzo total y x_{2j} el costo del proyecto) de la fábrica *j* (DMUj); *yj* y *Aj*, la salida (puntos de función) y el coeficiente o peso asignado de la fábrica *j*, respectivamente; *xi0* y *y0*, la *i*th entrada y la salida de la fábrica bajo evaluación (DMU0), respectivamente.

El modelo busca determinar la eficiencia de la fábrica de *software* a través de la minimización del esfuerzo total y costo del proyecto (1) en el DMU virtual. Por ello, si Θ=*1*, el DMU virtual requiere tantas entradas como lo hace una DMU real bajo evaluación, por lo cual no se evidencia ineficiencia. Sin embargo, si Θ<1, el DMU virtual requiere menos entradas para obtener el nivel de salida logrado por el DMU real bajo evaluación; es decir, es ineficiente.

Se consideran restricciones que buscan encontrar la combinación de las entradas y los pesos ponderados (A) que determinan la fábrica de *software* más eficiente (2), y que las salidas sean mayores o se mantengan constantes (3). Asimismo, el total de pesos ponderados que afectan a las entradas y salidas debería ser igual que 1 (4), al tiempo que cada peso ponderado debe tener un valor positivo (5). A continuación, en la Tabla 25, se muestra la información para describir un ejemplo.

Tabla 25. Información acerca de fábricas de *software*

DMU	Sector	Esfuerzo total (TE)	Costo del proyecto (PC)	Puntos de función (FP)
SF001	Banca	54 788.36	$ 2 161 797.26	1700.00
SF002	Banca	17 268.00	$ 663 962.77	594.00
SF003	Gobierno	79 646.99	$ 3 187 837.24	2736.00
SF004	Gobierno	62 700.98	$ 1 779 787.46	1500.00
SF005	Gobierno	63 595.12	$ 2 588 293.33	2183.00
SF006	Gobierno	109 362.63	$ 4 232 423.03	3760.00
Total		**387 362.08**	**$ 14 614 101.08**	**12 473.00**

El modelo DEA de este ejemplo para la DMU SF001 es el siguiente:

$$Minimizar\ \theta$$

$$\lambda_1(54{,}788.36) + \lambda_2(17{,}268.00) + \ldots + \lambda_6(109{,}362.63) \leq \theta(54{,}788.36)$$

$$\lambda_1(2{,}161{,}197.26) + \lambda_2(663{,}962.77) + \ldots + \lambda_6(4{,}232{,}423.03) \leq \theta(2{,}161{,}797.26)$$

$$\lambda_1(1{,}700) + \lambda_2(594) + \ldots + \lambda_6(3{,}760) \geq 1{,}700$$

$$\lambda_1 + \lambda_2 + \ldots + \lambda_6 = 1$$

$$\lambda_1, \lambda_2 \ldots, \lambda_6 \geq 0$$

• Fase 2. El objetivo de esta fase es identificar los proyectos de fábricas de *software* más eficientes a través de la minimización del esfuerzo total y el costo del proyecto consumidos en un período determinado por los proyectos, pero se debe mantener la cantidad de puntos de función. Este modelo asume dos entradas, una salida, y eme DMU, tal como se muestra a continuación:

$$Minimizar\ \theta$$

$$\sum_{j=1}^{160} \lambda_j * x_{ij} \leq \theta\, x_{i0} \quad i = 1, 2, \ldots, n;$$

$$\sum_{j=1}^{160} \lambda_j * y_j \geq y_0;$$

$$\sum_{j=1}^{160} \lambda_j = 1\,;$$

$$\lambda_j \geq 0 \quad j = 1, 2, \ldots, m;$$

Donde θ representa la medida de productividad o eficiencia de los proyectos de fábrica de software; x, la ith entrada (x_{ij} es el esfuerzo total y *x2j* el costo del proyecto) del proyecto *j* (DMUj); *yj* y *Áj*, la salida (puntos de función) y el coeficiente o peso asignado del proyecto j, respectivamente; *xi0* y *y0*, la ith entrada y la salida del proyecto bajo evaluación (DMU0), respectivamente.

5.3. Fuente de datos

5.3.1. Recopilación de datos

Para el presente estudio, se consideró la información obtenida de los diferentes proyectos de la fábrica de *software* de Palo Alto II (ex GMD S. A.), empresa de Outsourcing de Procesos de Negocios, Tecnología de la Información (TI) y Transformación Digital con mayor confiabilidad y 33 años de experiencia en el Perú. Esta cuenta con un *staff* de 3000 profesionales e infraestructura adecuada que lo convierte en la fábrica de *software* más grande en el país. Además, tiene certificaciones de calidad como ISO 9001, ISO 27001, OHSAS 18001, ISO 20000, ISO 22301, NTP 392-030 y metodologías CMMI-5, ITIL y PMI (GMD, 2018); de acuerdo a la consultora International Data Corporation [IDC] (2017), es la empresa líder de Outsourcing en Perú.

Esta empresa se enfoca en desarrollar soluciones de *software* para el sector público y privado bajo el modelo de fábrica de *software*; así, implanta este modelo de manera local en cada cliente y para ello posee su propio personal, pero sigue los lineamientos de la fábrica de *software* matriz. Por ello, se han seleccionado para el presente estudio solo seis fábricas de *software* de las dieciocho existentes por ser estas las que cumplen los criterios de realizar actividades de la componente producción de *software*.

Asimismo, se ha considerado como parte de la muestra a 160 proyectos que están distribuidos entre las seis fábricas de *software* seleccionadas, los cuales han sido desarrollados en el período de noviembre 2015 a octubre 2017 y han tenido una duración aproximada de 5 a 7 meses. Inicialmente, se recolectó la información de 200 proyectos; sin embargo, se procedió a realizar una depuración debido a que se presentaban inconsistencias en la data. Por lo tanto, se obtuvo solo 160 proyectos que han sido utilizados para el análisis respectivo. La información obtenida de estos está referida al tamaño del proyecto, tipo de sector, esfuerzo y costo del proyecto.

5.3.2. Análisis descriptivo de la muestra

Fase 1

Como se puede observar en la Tabla 26, se consideró la información de las fábricas de *software* del sector banca y estatal (la información se encuentra disponible en https://tinyurl.com/ycfyp5po). Por otro lado, las estadísticas descriptivas para entradas y salidas de las fábricas de *software* se muestran en la Tabla 27.

Tabla 26. Fábricas de *software* considerados en la muestra

DMU	Sector	Número de proyectos	Esfuerzo total (TE)	Costo del proyecto (PC)	Puntos de función (FP)
SF001	Banca	23	54 788.36	$ 2 161 797.26	1700.00
SF002	Banca	7	17 268.00	$ 663 962.77	594.00
SF003	Gobierno	33	79 646.99	$ 3 187 837.24	2736.00
SF004	Gobierno	22	62 700.98	$ 1 779 787.46	1500.00
SF005	Gobierno	21	63 595.12	$ 2 588 293.33	2183.00
SF006	Gobierno	54	109 362.63	$ 4 232 423.03	3760.00
Total		**160**	**387 362.08**	**$ 14 614 101.08**	**12 473.00**

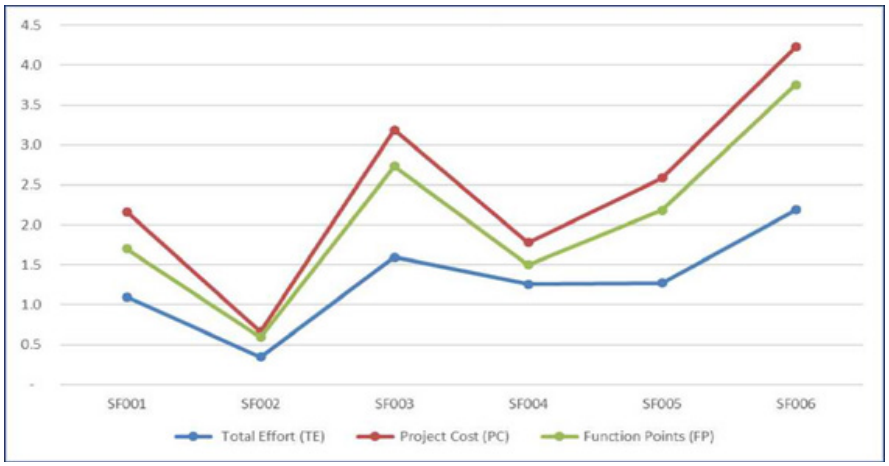
Tabla 27. Estadísticas descriptivas para variables de entrada y salida-Fábricas de *software*

Estadísticas	Esfuerzo total (TE)	Costo del proyecto (PC)	Puntos de función (FP)
Media	64 560.35	$ 2 435 683.51	2 078.83
Mediana	63 148.05	$ 2 375 045.29	1 941.50
Desviación estándar	30 236.73	$ 1 221 597.77	1 091.09
Mínimo	17 268.00	$ 663 962.77	594.00
Máximo	109 362.63	$ 4 232 423.03	3 760.00

A partir de la información de las fábricas de *software*, se puede notar que en la Figura 13, si solo se considera la cantidad de FP producidos, las fábricas SF006 y SF003 son las que presentan mayor productividad. Sin embargo, también presentan esfuerzos similares a las fábricas SF004 y SF005, que al mismo tiempo generan menores costos, a fin de determinar si dichos productos han sido implementados con eficiencia. Entonces, un análisis de la productividad, considerando solo los FP, no es suficiente.

Figura 13.

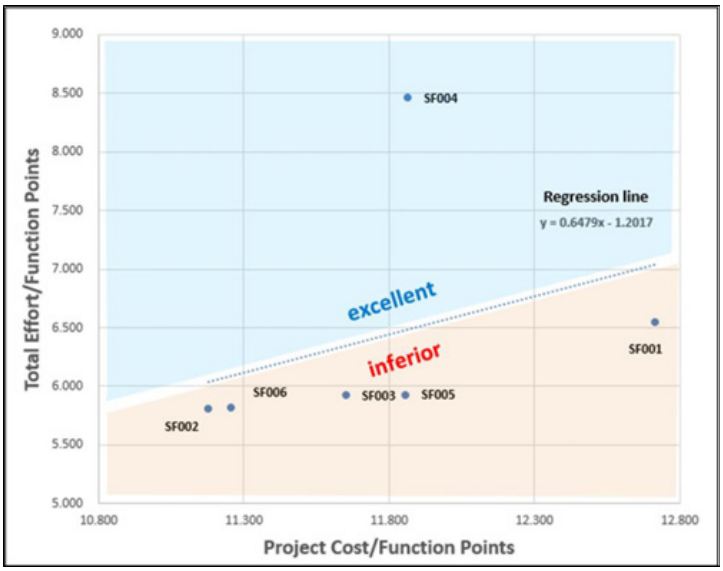
Fábricas de *software* consideradas en la muestra



Asimismo, se observa en la Figura 14 que, si se considera la técnica estadística de regresión lineal enfocada en promedios, se podría asumir que la fábrica SF004 es la más productiva, ya que está sobre la línea de tendencia; mientras que las demás (SF001, SF002, SF003, SF005, SF006) deberían mejorar su productividad por encontrarse debajo del promedio. Por lo tanto, un análisis de la productividad, considerando solo los promedios de productividad, no es suficiente.

Figura 14.

Línea de regresión estadística para fábricas de *software*



Fase 2

En esta fase se consideró 160 proyectos desarrollados por las seis fábricas seleccionadas de la fase 1. En la Tabla 28 se muestran 10 de los 160 proyectos de fábrica de *software* que se han utilizado para el análisis. La información se encuentra disponible en la ruta: https://tinyurl.com/ycfyp5po. Por otro lado, en la Tabla 29 se presentan las estadísticas descriptivas para variables de entrada y salida en los proyectos de *software*.

Tabla 28. Proyectos de fábricas de *software* considerados en la muestra

DMU	Esfuerzo total (TE)	Costo del proyecto (PC)	Puntos de función (FP)
SP001	1 427.84	$ 45 267.01	49.00
SP002	1 279.90	$ 41 195.87	28.00
SP003	1 063.74	$ 41 162.56	25.00
SP004	1 043.02	$ 40 567.90	25.00
SP005	1 358.80	$ 52 708.51	36.00
SP006	1 385.02	$ 52 679.84	10.00
SP007	1 991.80	$ 77 004.22	69.00
SP008	3 888.00	$ 178 606.37	120.00
SP009	1 530.50	$ 69 905.58	50.00
SP010	2 304.00	$ 81 217.63	62.00

Tabla 29. Estadísticas descriptivas para variables de entrada y salida-Proyectos de fábricas de *software*.

Estadísticas	Entradas		Salidas
	Esfuerzo total (TE)	Costo del proyecto (PC)	Puntos de función (FP)
Media	2421.01	$ 91 338.13	77.96
Mediana	1781.07	$ 73 292.85	61.00
Desviación estándar	1672.68	$ 54 284.23	47.47
Mínimo	1012.00	$ 34 196.03	10.00
Máximo	13 524.05	$ 328 033.66	282.00

5.4. Resultados empíricos e interpretaciones

Para evaluar la eficiencia relativa de las fábricas de *software* y sus respectivos proyectos, con múltiples entradas (esfuerzo total, costo del proyecto) y una salida (puntos de función), se utilizó el modelo DEA VRS-I. Los resultados fueron computados con DEAP v2.1 *software*. En la Tabla 30 se muestra una lista de abreviaturas que se utilizaron en la interpretación de los resultados.

Tabla 30. Tabla de abreviaturas

Acrónimo	Descripción
DEA	Análisis envoltorio de datos
CRS	Retornos de escala constante
VRS	Retornos de escala variable
TE	Eficiencia técnica
CRSTE	Eficiencia técnica con retornos de escala constante
VRSTE	Eficiencia técnica con retornos de escala variable
SE	Escala de eficiencia
IRS	Retornos de escala creciente
DRS	Retornos de escala decreciente
VRSEFF	Proyectos eficientes VRS

A continuación, se describen los resultados obtenidos para las dos fases en evaluación.

Fase 1

De las seis fábricas de *software* analizadas, hay dos DMU eficientes (VRSTE = 1), lo que satisface la "regla de un tercio", que indica si el número de DMU eficientes no es más de un tercio del número total de DMU en la muestra, el tamaño de esta es aceptable (Cooper et al., 2004); en este caso, se

cumple. En la Tabla 31 se observa la eficiencia (VRSTE) de las fábricas de *software* para el modelo DEA VRS-I.

Tabla 31. Eficiencia DEA VRS-I, fábrica de *software*

DMU	VRSTE
SF001	0.893
SF002	**1.000**
SF003	0.977
SF004	0.937
SF005	0.996
SF006	**1.000**

De acuerdo a la Figura 15, en promedio, los puntajes de eficiencia de las fábricas de *software* son a) 96.6 % para CRSTE, en general, las fábricas de *software* podrían reducir sus entradas en un 3.4 % para obtener el mismo número de FP; b) 96.7 % para VRSTE, una mejor fábrica de *software* podría ser capaz de reducir el consumo de entradas en un 3.3 %; c) 99.9 % para SE, al ajustar la escala, las fábricas de *software* podrían reducir sus entradas en un 0.1 %.

Figura 15.
Resumen de eficiencias

```
EFFICIENCY SUMMARY:

 firm  crste  vrste  scale

SF001  0.893  0.893  1.000  -
SF002  1.000  1.000  1.000  -
SF003  0.977  0.977  1.000  -
SF004  0.933  0.937  0.995 drs
SF005  0.996  0.996  1.000  -
SF006  1.000  1.000  1.000  -

mean  0.966  0.967  0.999

Note: crste = technical efficiency from CRS DEA
      vrste = technical efficiency from VRS DEA
      scale = scale efficiency = crste/vrste

Note also that all subsequent tables refer to VRS results
```

En la Figura 16, se nota que SF004 tiene una eficiencia técnica del 93.7 % y una escala de eficiencia del 99.5 %, con una tendencia a retornos de escala decreciente (DRS). Al mejorar la operación de la fábrica de *software*, se podría reducir el uso de entradas en un 6.3 % (100-93.7). Por otra parte, al ajustar la fábrica de *software* a su tamaño óptimo, se podría reducir el uso de entradas en 0.5 % (100-99.5).

Figura 16.

Resultados para SF004

```
Results for SF004:
Technical efficiency = 0.937
Scale efficiency     = 0.995   (drs)
 PROJECTION SUMMARY:
  variable          original        radial         slack     projected
                       value      movement      movement         value
 output Points          1.500         0.000         0.000         1.500
 input  Effort         62.700        -3.959       -15.538        43.203
 input    Cost       1780.000      -112.388         0.000      1667.612
 LISTING OF PEERS:
  peer   lambda weight
 SF006       0.281
 SF002       0.719
```

La columna "original value" contiene los valores originales de las variables de estudio de la fábrica de *software*: SF004 produce 1.5 FP con 62.7 horas de esfuerzo y un costo total de $1780. Sin embargo, SF004 podría producir el mismo número de FP con menos entradas: 43.20 horas en vez de 62.7; $1667.612 en lugar de 1780 (ver la columna "projected value").

La reducción en las entradas TE y PC es 6.31 % de los valores originales:

((–3.959/62.7) x 100) para la entrada TE y ((–112.388/1780) x 100) para la entrada PC. Sin embargo, en el caso de la entrada TE, para que llegue a ser eficiente no solo debe reducirse en 6.31 % (menos 3.959 de la columna "radial movement"), sino también en un 15.538 adicional (columna "slack movement"); es decir, para que la entrada TE llegue a ser eficiente debe reducirse en un 31.1 %.

Para mejorar su eficiencia, SF004 tiene que analizar las prácticas de la SF006 y SF002, las cuales son identificadas como sus pares. Para ser un par (o *benchmark*), una fábrica de *software* debe tener una eficiencia técnica del 100 %. El peso ponderado (!) asociado con cada par corresponde a su importancia relativa entre el grupo de pares. Idealmente, SF004 debería analizar las mejores prácticas desde una fábrica de *software* compuesta por 28.1 % de SF006 y 71.9 % de SF002. Como una fábrica de *software* virtual no existe, SF004 debería concentrase en analizar las mejores prácticas del par asociado con el mayor valor de X, por lo cual debería compararse con la SF002.

La Tabla 32 muestra las fábricas de *software* de referencia VRSTE considerada en la muestra. Una de estas siempre es seleccionada desde un conjunto de fábricas establecidas en la frontera eficiente. En el caso del modelo VRSTE, una fábrica de referencia siempre será eficiente y similar tamaño a las fábricas que la consideran como referencia (Stensrud y Myrtveit, 2003).

Tabla 32. Eficiencia DEA VRS-I, fábrica de *software*

DMU[1]	VRSTE Orientación de entrada[2]	Benchmark (X óptimo)	
		SF006*	SF002*
SF001	0.893	0.344	**0.656**
SF002	**1.000**		1.000
SF003	0.977	**0.656**	0.344
SF004	0.937	0.281	**0.719**
SF005	0.996	**0.500**	**0.500**
SF006	**1.000**	1.000	

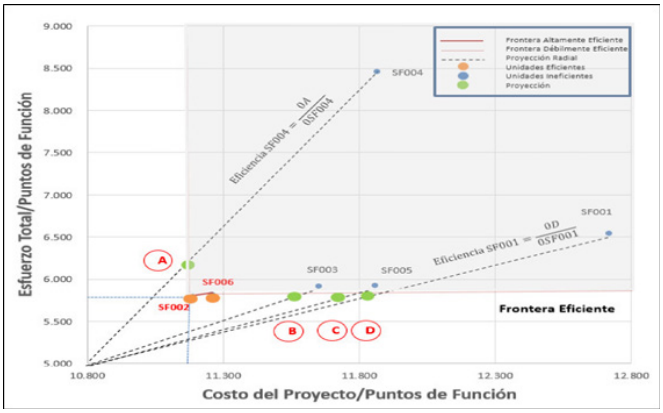
Nota. [1] lista de todas las fábricas; [2] eficiencia de las fábricas; * pesos asociados a las fábricas de referencia.

Por ejemplo, cuando se revisa la fábrica SF004, se observa que, para mejorar su eficiencia, debería acercarse a la frontera eficiente y el pareto más cercano es dado por la combinación ponderada (pesos) de SF002 y SF006 (Virtual Software Factory A).

También se observa que la SF002 es más adecuada como fábrica de referencia, porque genera mayor productividad con menor esfuerzo y costo que SF006 (71.9 % vs. 28.1 %), tal como se aprecia en la Figura 17, donde las fábricas SF002 y SF006 se encuentran en la frontera de eficiencia VRS.

Figura 17.

Conjunto de referencias VRS para la fábrica de *software*



Fase 2

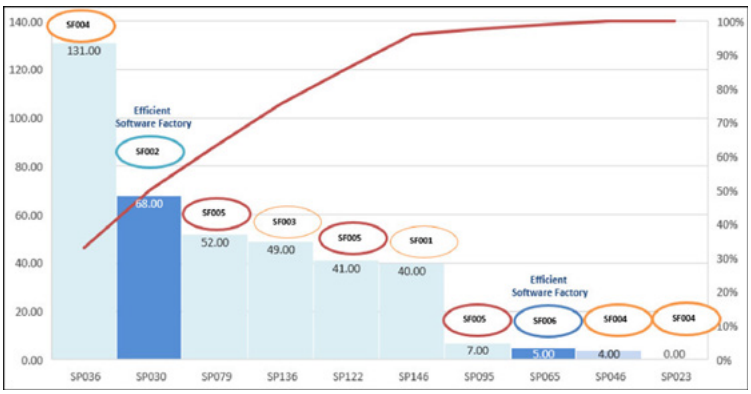
En esta fase se identificaron 10 proyectos eficientes de 160 proyectos evaluados (Tabla 33). Además, en la Figura 18, se presentan los proyectos eficientes que más se toman en cuenta como proyectos que aplican buenas prácticas y reflejan una mayor productividad, entre ellos están el SP036 (SF004) y el SP030 (SF002). También se observa que el proyecto más eficiente no está dentro de las fábricas de *software* más eficientes.

Tabla 33. Eficiencia DEA VRS-I, Proyectos de fábrica de *software*

DMU	VRSTE	Fábrica de *software* (SF)	Frecuencia
SP023	1	SF004	0
SP030	1	SF002	**68**
SP036	1	SF004	131
SP046	1	SF004	4
SP065	1	SF006	**5**
SP079	1	SF005	52
SP095	1	SF005	7
SP122	1	SF005	41
SP136	1	SF003	49
SP146	1	SF001	40

Por ello, se podría mencionar que, si los proyectos de fábrica de *software* eficientes replicarán las buenas prácticas o estrategias que realiza el proyecto SP036, se podría tener una mayor productividad. Asimismo, se observa que las fábricas de *software* que tienen la mayor cantidad de proyectos eficientes son la SF005 y la SF004.

Figura 18.
Cantidad de veces que un proyecto es referencia de otro proyecto

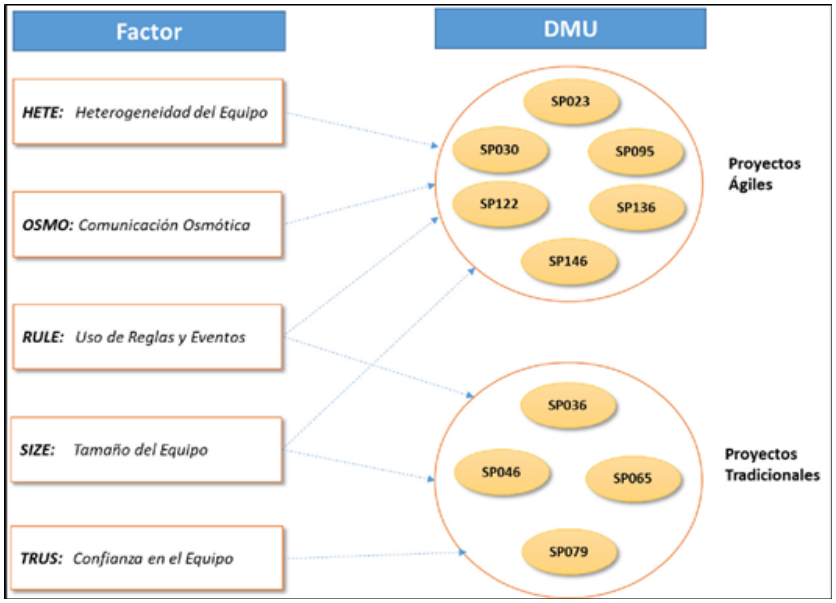


Como se muestra en la Tabla 34, en promedio, los puntajes de eficiencia de las fábricas de *software* son a) 90.6 % para CRSTE, en general, los proyectos de fábrica de *software* podrían reducir sus entradas en un 9.4 % para obtener el mismo número de FP; b) 94 % para VRSTE, un mejor proyecto de fábrica de *software* podría ser capaz de reducir el consumo de entradas en un 6 %; c) 96.3 % para SE, al ajustar la escala, proyectos de fábrica de *software* podrían reducir sus entradas en un 3.7 %.

Tabla 34. Eficiencia promedio DEA VRS

En la Figura 19 se puede identificar qué factores han influenciado para que los proyectos de fábrica de *software* sean eficientes.

Figura 19.
Factores y su influencia en los proyectos de fábrica de *software* eficientes



En la Tabla 35 se detallan algunas características de los proyectos de fábrica de *software* eficientes y cómo los factores identificados en el capítulo anterior han influenciado para el logro del objetivo.

Tabla 35. Caracterización de los proyectos de fábrica de *software*.

Factor	Tipo de metodología	Justificación
Heterogeneidad del equipo	Ágil	En estos proyectos se ha identificado la participación de múltiples roles con diferentes habilidades y capacidades para el desarrollo de un producto, que han resaltado por ser proyectos de innovación donde los requerimientos son cambiantes, lo que implica la necesidad de incorporar nuevas perspectivas que ayudan a los miembros del equipo a superar problemas desde diferentes ángulos.
Comunicación osmótica	Ágil	Los miembros de los proyectos se encuentran localizados en ambientes de trabajo llamados Cuartos de Guerra, lo cual permite que la información fluya en el fondo auditivo de los miembros del equipo, de tal forma que ellos captan la información relevante, facilitan el aprendizaje y reducen los tiempos de curva de aprendizaje, ya que todo el equipo conoce las actividades que vienen realizando cada uno de sus miembros.
Uso de reglas y eventos	Ágil tradicional	En los proyectos eficientes, se identificó que en los equipos de trabajo se han establecido reglas para facilitar la colaboración. Por ejemplo, el uso del Log de impedimentos para reportar cualquier problema que esté dificultando la culminación de actividades comprometidas y hace que esta sea visible a todo el equipo. También se ha identificado la implementación de *timebox* en los eventos, lo cual evita que el equipo se distraiga en actividades accesorias que impidan el cumplimiento del objetivo.
Tamaño del equipo	Ágil tradicional	Los proyectos utilizan equipos de trabajo pequeños, que oscilan entre 6 a 10 personas, lo cual ha facilitado los niveles de comunicación.
Confianza en el equipo	Ágil	En los proyectos se han implementado prácticas ágiles, tal como el proceso de estimación, que se realiza por los miembros del equipo que van a ejecutar las tareas, las cuales son propuestas en los planes de trabajo. Esto ha permitido que el equipo se sienta comprometido y genere niveles de confianza entre los diferentes niveles de la organización.

De los 160 proyectos analizados, solo 10 son DMU eficientes (VRTSE = 1), lo que corresponde aproximadamente al 6.3 % de las DMU. Además, la "regla de un tercio" indica que el tamaño de la muestra es aceptable si el número de DMU eficientes no es más de un tercio del número total de DMU en la muestra (Cooper *et al.*, 2004). Como el número de DMU es 160 y la cantidad de proyectos eficientes 10, entonces, se satisface la regla. Una DMU con una eficiencia menor que 1 es relativamente ineficiente con respecto a sus puntos de referencia.

En la Tabla 36 se presentan los proyectos de referencia VRS para cada proyecto de fábrica de *software* considerado en la muestra. Un proyecto de referencia siempre es seleccionado desde un conjunto de proyectos establecidos en la frontera eficiente. En el caso del modelo DEA VRS, dicho proyecto siempre será eficiente y de similar tamaño a los proyectos que la consideran como referencia (Stensrud y Myrtveit, 2003).

Tabla 36. Conjunto de referencias VRS para los proyectos de fábrica de *software*.

DMU	Eficiencia	*Benchmark* (κ óptimo)							
	VRS-I SP023	SP030	SP036	SP046 SP065	SP079	SP095	SP122	SP136	SP146
SP001	0.98687	0.029	0.123		0.848				
SP002	0.98617				0.274				0.726
SP003	0.99381			0.396		0.604			
SP004	0.98791			0.473		0.527			
SP005	0.99168							0.545	0.455
SP006	0.99404			0.139		0.861			
SP007	0.99997	0.050	0.743		0.207				
SP008	0.99455		0.757				0.243		
SP009	0.9952		0.438					0.563	
SP010	0.99207	0.066	0.389		0.545				
SP151	0.97853							1.000	
SP152	0.99198		0.879				0.121		
SP153	0.99173		0.771				0.229		
SP154	0.99791		0.219					0.103	0.678
SP155	0.99504	0.142	0.757		0.101				
SP156	0.9899	0.005	0.562		0.433				
SP157	0.99979	0.003	0.389		0.608				
SP158	0.99489	0.025	0.485		0.490				
SP159	0.98488							0.220	0.780
SP160	0.99771	0.532	0.212				0.256		

Por ejemplo, cuando se revisa el proyecto SP001, se observa que tiene tres proyectos en su conjunto de referencia: proyectos SP030, SP036 y SP079. También, el proyecto SP079 es más importante que SP036 y SP030 (84.8 % vs. 12.3 % y 2.9 %, respectivamente).

5.5. Procedimiento de medición

El procedimiento a seguir para la aplicación del modelo de medición de la productividad basado en DEA inicia con el alcance del procedimiento que está orientado a las fábricas de *software* y los proyectos implementados en estas. Luego, se deben establecer las responsabilidades de los diferentes roles que participan en el proceso de medición de la productividad tal como se describe en la Tabla 37:

Tabla 37. Responsabilidades de los participantes en el procedimiento

Rol	Componente	Responsabilidades
Gestor del proyecto	Gestión del proyecto	Responsable de la gestión del proyecto Asignación de los recursos necesarios para la ejecución del proyecto. Seguimiento de las actividades ejecutadas por el equipo del proyecto.
Equipo de fábrica	Producción de *software*	Ejecución de las actividades del proyecto. Reporte de actividades diarias.
Equipo de arquitectura	Soporte	Validación de la información reportada por los diferentes equipos de trabajo que vienen ejecutando proyectos.
Gestor de indicadores	Soporte	Integración de la información de los proyectos de fábrica de *software*. Aplicación del modelo de medición de la productividad. Consolidación y difusión de los resultados de la aplicación del modelo. Implementación de estrategias de mejora continua. Mejoras y ajustes en el modelo DEA

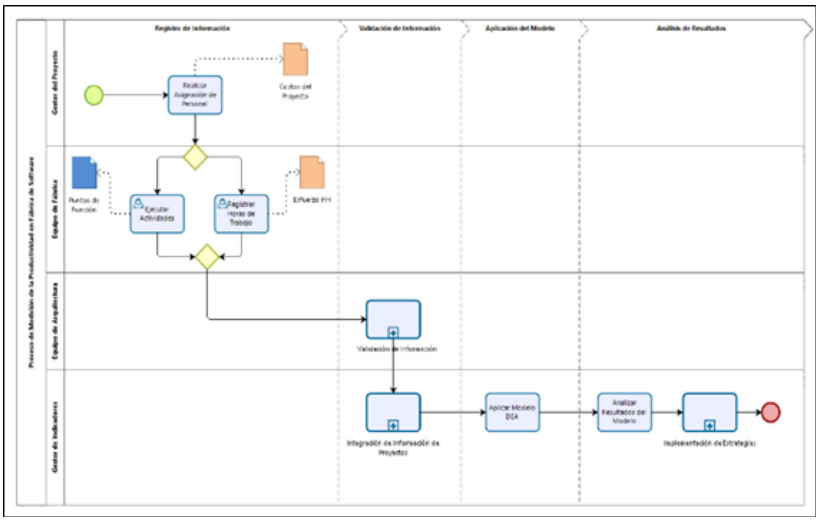
Como parte del procedimiento, se registran las entradas y salidas. En las entradas se debe considerar el insumo y su origen; en el caso de las salidas, el servicio o producto y el destino, como se observa en la Tabla 38.

Tabla 38. Entradas y salidas

Entradas	
Insumo	**Origen**
Esfuerzo	Equipo de fábrica
Costos del proyecto	Gestor del proyecto
Salidas	
Servicio o producto	**Destino**
Puntos de función	Cliente

En la Figura 20 se visualiza el proceso y cada una de las actividades para la medición de la productividad.

Figura 20.
Proceso para la medición de la productividad



En la Tabla 39 se detallan las actividades a realizarse durante el proceso de medición de la productividad.

Tabla 39. Actividades del proceso de medición de la productividad

Rol	Actividad
Gestor del proyecto	Realizar la asignación del personal que será parte del equipo de fábrica.
Equipo de fábrica	Ejecutar actividades del proyecto. Registrar horas de trabajo que se han utilizado para la ejecución de las tareas o actividades.
Equipo de arquitectura	Validar la información reportada por los diferentes equipos de trabajo. En caso exista alguna incidencia u observación, esta será reportada al gestor de proyecto para su respectiva validación.
Gestor de indicadores	Integrar la información de los diferentes proyectos para su respectiva evaluación. Depurar la información inconsistente del proyecto y reportar al gestor de proyecto respectivo. Introducir valores (entradas y salidas) en las herramientas del modelo DEA. Evaluar los resultados obtenidos con los diferentes gestores de proyectos, a fin de definir las buenas prácticas utilizadas en los proyectos que han obtenido mejores resultados. Definir estrategias para la mejora continua. Difundir información a los proyectos.

Para la medición se deben realizar modificaciones y revisiones al presente documento, por lo que se debe tener en cuenta el formato presentado en la Tabla 40. Asimismo, considerar las referencias, los instructivos de trabajo y los procedimientos de cada organización.

Tabla 40. Formato de revisión

Cláusula (*)	Cambio
(n.° de cláusula)	(Breve descripción del cambio)
	(Agregue las filas que sean necesarias. Eliminar la tabla cuando es Revisión 0)

Nota. Cuando en una misma revisión se realiza un cambio que afecte varias cláusulas, esta se menciona una sola vez anotando las cláusulas afectadas. Solo se registran los últimos tres cambios.

CAPÍTULO VI

IMPORTANCIA DE LA MEDICIÓN DE LA PRODUCTIVIDAD EN LAS FÁBRICAS DE SOFTWARE

Durante las últimas décadas, los productos de *software* han ido posicionándose en el mercado económico de tal manera que han alcanzado grandes avances en áreas como marketing, finanzas, recursos humanos, entre otros. Por otra parte, las fábricas de *software* lograron favorecer el sistema digital de un sinfín de empresas, hecho que se ha visto reflejado no solo en un ingreso económico superior de ganancias, sino también en un servicio eficiente por parte de las organizaciones a sus clientes.

Ahora bien, si se enfoca esta serie de ventajas desde una perspectiva empresarial interna, se observa que entre los logros alcanzados por las empresas se encuentra el de la especialización de los trabajadores, quienes al delegar algunas funciones básicas a ciertos sistemas *software* han logrado especializarse y obtener un mayor rendimiento laboral. Esto le ha permitido al trabajador ser más eficiente al tiempo que ha brindado resultados satisfactorios a los clientes.

Por lo tanto, debido a los resultados positivos que las fábricas de *software* han ido generando, comenzó a surgir una mayor demanda que devino en un incremento en la formación de fábricas de software, así como de su producción. En otras palabras, el número de fábricas dispuestas a cumplir con las demandas de los diversos clientes que el ser informático sostiene comenzó a ser cada vez mayor. En relación con ello, surge de parte de los fabricantes de *software* una serie interrogantes sobre la productividad de los desarrolladores de software y cómo hacerle frente a la competitividad, la optimización de recursos, entre otros.

Al respecto, cabe señalar que la elaboración de un producto *software* no puede ser medida de la misma forma que la producción de elementos de rápida fabricación, pues esta puede ser medible en relación con la cantidad que se produce al día. Sin embargo, para medir la productividad de un sistema *software* no es suficiente establecer la ratio de salidas y entradas, sino que se debe tomar en cuenta otras consideraciones, tales como la dificultad, las habilidades intrínsecas del equipo del proyecto, la facilidad de interacción con el cliente y el nivel de requerimientos no funcionales expresado como confiabilidad y rendimiento, lo cual complejiza su medición.

Ante ello, surgen puntos de vista como los de los desarrolladores informáticos que señalan que la cantidad de códigos producidos por el sistema *software* puede ser una idea de medir la productividad. No obstante, para el cliente, esta es una observación en cierta medida superficial, pues podría ser interpretado como el grado de funcionalidad lograda por el sistema de *software*. A partir de ello se infiere cuán importarte es medir la productividad en las fábricas *software*, las cuales, recomiendan los expertos, deben estar enfocadas no solo a nivel de factoría, sino también de proyecto. Dos aspectos que no siempre son directamente proporcionales.

Ahora bien, para realizar un análisis cabal de la productividad es necesario, de acuerdo a Fontalvo *et al.* (2017), establecer la relación entre costo y gestión de calidad, así como el rol humano y su papel en conceptos de producción, los cuales deben ser calificados para dar a conocer los niveles de productividad existentes.

Por otro lado, Loayza (2016) explica que la productividad debe ser entendida por cuatro componentes principales que la conforman: 1) la innovación, que consiste en la creación de nuevas

tecnologías, productos y procesos, 2) la educación, que disemina la innovación y desarrolla tanto conocimientos como habilidades, 3) la eficiencia, que procura el uso y la distribución eficaz de los recursos productivos, 4) la infraestructura física e institucional, que otorga bienes y servicios públicos en apoyo a la economía.

Otros autores como Medina (2010, citado en Fontalvo *et al.*, 2017) conceptualizan el término con relación a cómo interactúan los factores de producción cuando un producto o servicio es elaborado; es decir, la relación entre los elementos de producción de las organizaciones, que resulta ser estratégico debido a que en esta fase las instituciones presentan mayor interés en crear la calidad de su producto o servicio con el fin de tener un agregado que lo diferencia de otras empresas. En esta etapa, el empresario buscará brindar la garantía de la excelencia de su producto o servicio. Al respecto, Fontalvo *et al.* (2017) afirma lo siguiente:

> Una empresa puede producir grandes cantidades de bienes y servicios, pero, este nivel de producción debe estar de acuerdo con los insumos consumidos. En muchas ocasiones el concepto de productividad se confunde con otros términos asociados a ella, tales como eficacia y eficiencia. (p. 50)

De lo expuesto, se confluye que medir la competitividad de un producto no es tarea fácil, pues su elaboración implica consideraciones que son difíciles de medir. Sin embargo, en la actualidad, existen propuestas que pretenden dar solución a esta problemática con el fin de establecer indicadores sobre el trabajo de desarrolladores informáticos.

Si bien hay diversos medios para conocer si la producción de una empresa está siendo la ideal, uno de los índices más comunes es la demanda del producto en el mercado; es decir, si el público busca adquirir más a menudo un bien o un servicio, se debe a que este satisface sus requerimientos. En esa línea se encuentra la productividad que, además de permitir una adecuada gestión de calidad, indica cuáles son los recursos que se están usando en producción y cómo se pueden optimizar. Asimismo, si la empresa logra posicionarse firmemente en el mercado, este puede devenir en mejoras salariales. Otro aspecto a tener en cuenta respecto a la productividad es que una mayor producción podrá ponerla en igualdad de condiciones frente a otras empresas.

CAPÍTULO VII
CONSIDERACIONES FINALES

Si una organización necesita conocer la eficiencia y la eficacia con las que se están desarrollando las actividades claves en el proceso de producción, es necesario que tengan clara la forma de cómo medir la productividad, la cual es una actividad compleja y no hay un consenso sobre los aspectos a tener en cuenta para su medición. La productividad incorpora diversos factores que en muchos casos no son tomados en cuenta, entre los más resaltantes en los modelos encontrados están los siguientes: tamaño del equipo, reutilización, complejidad, cohesión del equipo, unidad del equipo, relaciones sociales, vida social, reuniones regulares, frecuencia de reuniones. Sin embargo, existen muchos factores que no han sido considerados, lo cual afecta la precisión de los modelos.

Asimismo, no existen modelos para la medición de la productividad en las fábricas de *software* como un todo, solo se ha encontrado estudios en su mayoría orientados a la unidad de trabajo de programación y muy pocos a las de análisis y diseño, y pruebas. Estos estudios representan al proceso de desarrollo como una unidad y dificultan la medición si se quisiera realizar de manera independiente por cada uno de los componentes que participan en el ciclo de vida de desarrollo del *software*. Tampoco se han encontrado evidencias de modelos que midan la productividad en todas las fases del proceso de desarrollo de *software*.

No obstante, los modelos no realizan una medición de la productividad externa, pues solo se han orientado a medir la producción interna de la organización, lo cual conllevaría a tomar malas decisiones, ya que se podría considerar una alta productividad en la interna. Sin embargo, esto podría dejar la propuesta por debajo del estándar, la cual sería categorizada como ineficiente.

En este trabajo se ha introducido tres nuevos factores que influyen en la productividad de *software* sustentadas en teorías y se ha estudiado la influencia de los factores tamaño del equipo (SIZE) y confianza en el equipo (TRUS) en las diferentes unidades de trabajo de la componente de la fábrica de *software*.

Los resultados obtenidos de la encuesta administrada a 150 profesionales involucrados en actividades dentro de fábricas de *software* en el Perú corroboraron la influencia de los factores propuestos: en la unidad de trabajo RULE y en TRUS se tiene una influencia significativa positiva, y en HETE, negativa.

Por otro lado, en la unidad de programación, heterogeneidad del equipo (HETE), tipo de comunicación osmótica en equipo (OSMO), tamaño del equipo (SIZE) y confianza en el equipo (TRUS), estas tienen una influencia significativa positiva. También, en la unidad de trabajo de pruebas, se ha demostrado que confianza en el equipo (TRUS) tiene una influencia significativa positiva.

Además, se puede visualizar que el factor confianza en el equipo (TRUS) afecta a las tres unidades de producción. También se reafirma que el factor tamaño del equipo (SIZE) tiene una influencia significativa positiva en la unidad de trabajo de programación; sin embargo, no tiene este mismo efecto en las demás unidades.

Por otra parte, se hizo una exhaustiva revisión sistemática de la literatura sobre factores que afectan la productividad y los modelos para medirla. Basado en ello, se introdujeron tres nuevos

factores que influyen en la productividad de una fábrica de *software* y un modelo con base en DEA orientado a entrada con retorno de escala variable para medir la productividad en fábricas de *software*.

También se identificaron 74 factores, los cuales fueron agrupados de acuerdo a categorías, ya que contemplan algunas que en muchos casos no han sido cubiertas, tales como entorno de desarrollo, cultura corporativa, cultura del equipo y entorno. Sin embargo, estas categorías son necesarias, pues agrupan a factores que son transversales a toda la organización y no son exclusivas del producto o proyecto.

De igual manera, se han identificado diez modelos de medición de la productividad de *software* que están basados en el análisis envoltorio de datos, CMMI y Six Sigma, lógica difusa, modelo de ecuaciones estructurales, reutilización de *software*, factor de productividad total, ISO 91264 IEEE Std.1045 y otras técnicas estadísticas. Si bien es cierto que se definen como modelos de medición de la productividad, muchos de ellos plantean una función; pero no derivan a una fórmula, solo se mencionan diversos factores a tener en cuenta en la medición.

Los modelos solo han sido diseñados para realizar una medición de la productividad interna; a pesar de ello, son incompletos, ya que han sido orientados en su mayoría para la unidad de trabajo de programación y muy pocos para análisis y diseño o pruebas. Esto genera un sesgo en la medición, ya que no permitiría establecer la productividad en los otros componentes de la fábrica de *software*.

Ahora bien, no se han encontrado modelos que midan la productividad en toda la fábrica de *software*. No obstante, se han identificado 74 factores que influyen en la productividad, entre ellos los más utilizados son los siguientes: tamaño de equipo, reutilización y complejidad. También existen muchos factores que no han sido tomados en cuenta en los modelos, tales como números totales de lanzamientos, problemas, riesgos, solicitud de cambio, ocurrencia de espera, duración de espera, motivo de espera.

Si se toman en cuenta los factores que existen en la literatura y los modelos que se han elaborado hasta la fecha, se observa que no existe una homogeneización de los factores a considerar en un modelo. Además, en muchos casos, no se ha tomado en cuenta algunos de los factores, lo cual afecta la precisión del modelo, ya que se considera que cuantos más factores cubra el modelo, más aspectos de la realidad se podrán establecer.

Por lo tanto, los tres nuevos factores propuestos que influyen en la productividad de *software* sustentados en teorías comprenden los factores de tamaño del equipo (SIZE) y confianza en el equipo (TRUS) en las diferentes unidades de trabajo de la componente de la fábrica de *software*.

Este libro también propone un modelo para la medición de la productividad basado en el modelo DEA orientado a entrada y con retornos de escala variable, que contempla dos fases: 1) la evaluación de las fábricas de *software* considerando el esfuerzo total y costo del proyecto como entradas y los puntos de función como salida; 2) la evaluación de la productividad de los proyectos que realiza la fábrica de *software*.

Los resultados sobre seis fábricas de *software* y 160 proyectos implementados por las respectivas fábricas muestran que el modelo identifica la fábrica de *software* más eficiente, la cual sirve como modelo de fábricas de su mismo tamaño o complejidad y facilita la tarea de la gestión de proyectos, ya que la orientación estará dada a determinar qué prácticas eficientes se están realizando en esta fábrica para lograr convertirse en un modelo a seguir.

El modelo también permite identificar en la fase 2 los proyectos más eficientes. Asimismo, se ha podido comprobar que es necesario medir no solo a nivel de fábrica de *software*, sino a nivel de sus proyectos, ya que, como muestran los resultados, no necesariamente una fábrica de *software* modelo tiene los proyectos más eficientes.

Además, realizar un análisis en dos fases permitirá descubrir qué factores se encuentran inmersos en estos proyectos, los cuales podrían ser implementados en los proyectos de la fábrica más eficiente y conllevaría a mejorar su eficiencia. En ese sentido, las fábricas de *software* (donde se incluye las más eficientes) podrían mejorar su productividad si tomaran en cuenta las buenas prácticas que se vienen implementando en los proyectos.

Por lo tanto, este modelo permite quitar el sesgo de medir la productividad en comparación con otras técnicas estadísticas, tal como se puede notar en el análisis de regresión lineal, donde muchas veces se consideran proyectos eficientes cuando no necesariamente lo son. Por ello, para el presente estudio, se ha utilizado el modelo DEA con retornos de escala variable y orientación de entrada (DEA VRS-I), ya que se busca minimizar las entradas (esfuerzo total y costo del proyecto) y mantener la salida actual (puntos de función) de las fábricas de *software* y sus respectivos proyectos.

REFERENCIAS BIBLIOGRÁFICAS

Abdel-Hamid, T. K. (1989). The dynamics of *software* project staffing: a system dynamics based simulation approach. *IEEE Transactions on Software Engineering*, *15*, 109-119.

Abdel-Hamid, T. K. y Madnick, S. E. (1991). *Software Project Dynamics: An Integrated Approach*. Prentice-Hall, Inc.

Acosta, H., Llorens, S., Escaff, R., Díaz, J., Troncoso, S., Salanova, M. y Sanhueza, J. (2019). ¿Confiar o no Confiar?: El Rol Mediador de la Confianza entre el Trabajo en Equipo y el Work Engagement. *Revista Interamericana de Psicología Ocupacional*, *38*(1), 85-99. DOI: 10.21772/ripo.v38n1a07

Agosin, M., Atal, J., Blyde, J., Busso, M., Cavallo, E., Chong, A., Daude, C., Fernández-Arias, E., Galindo, A., Ibarrarán, P., Izquierdo, A., Llisterri, J., Lora, E., Ludeña, C., Madrigal, L., Maffioli, A., Mesquita, M., Navarro, J., Ñopo, H., … Zuñiga, P. (2010). *La era de la productividad: como transformar las economías desde sus cimientos*. Banco Interamericano de Desarrollo.

Alba, E. y Chicano, J. (2005). *Management of software projects with gas*. Proceedings of the 6th Metaheuristics International Conference (MIC'05) (pp. 13-18). Elsevier Science Inc., Vienna, Austria.

Alba, E. y Chicano, J. (2007). *Software* project management with GAs. *Information Sciences*, *177*, 2380-2401.

Alzate, A., Montoya, C. y Muñoz A. (2008). *Guía metodológica para la construcción de fábricas de software* [trabajo de grado, Universidad EAFIT]. https://repository.eafit.edu.co/bitstream/handle/10784/2769/Andres_Alzate_Carolina_Montoya_Arbey_Mu%C3%B1oz_2008.pdf?sequence=11

Ambrocio, T. (2017). *Modelo de fábrica de software para el desarrollo de sistemas de información en la empresa UCL Global Perú S.A.C., Callao 2017* [tesis de maestría, Universidad César Vallejo] Repositorio Institucional UCV. https://repositorio.ucv.edu.pe/bitstream/handle/20.500.12692/19337/Ambrocio_DT.pdf?sequence=1&isAllowed=y

Ancona, D. G. y Caldwell, D. F. (1992). Demography and design: predictors of new product team performance. *Organization Science*, *3*(3), 321-41.

Aoyama, M. (1996). Beyond *software* factories: concurrent-development process and an evolution of *software* process technology in Japan. *Information and Software Technology*, *38*(3), 133-143.

Aravena, C., Cavada, C. y Mulder, N. (2012). Contribución al crecimiento económico de las tecnologías de la información y las comunicaciones y de la productividad en la Argentina, el Brasil, Chile y México. *Cepal*, (76). https://repositorio.cepal.org/bitstream/handle/11362/4780/S1100964_es.pdf?sequence=1&isAllowed=y

Arcudia-Abad, C., Solís-Carcaño, R. G. y Cuesta-Santos, A. R. (2007). Propuesta tecnológica para incrementar la productividad en la construcción masiva de vivienda. *Ingeniería. Investigación y Tecnología (FI-UNAM)*, *8*(2), 11. http://www.scielo.org.mx/pdf/iit/v8n2/v8n2a1.pdf

Asmild, M., Paradi, J. C. y Kulkarni, J. C. (2006). Using data envelopment analysis in *software* development productivity measurement. *Software Process Improvement and Practice*, *11*(6), 561-572. DOI: 10.1002/spip.298

Banker, R. D., Charnes, A. y Cooper, W. W. (1984). Some models for estimating technical and scale inefficiency in data envelopment analysis. *Manag Sci*, *30*(9), 1078-1092.

Banker, R. D. y Kemerer, C. F. (1989). Scale economies in new *software* development. *IEEE Trans Softw Eng*, *15*(10), 1199-1205.

Banker, R. D., Datar, S. M. y Kemerer, C. F. (1991). A model to evaluate variables impacting the productivity of *software* maintenance projects. *Manag Sci*, *37*(1), 1-18.

Banker, R. D., Chang, H. y Kemerer, C. F. (1994). Evidence on economies of scale in *software* development. *Inf Softw Technol*, *36*(5), 275-282.

Banker, R. D. y Kauffman, R. J. (1991). Reuse and productivity in integrated computeraided *software* engineering: An empirical study. *MIS Quarterly*, *14*(3), 374-401.

Basili, V. R. (1989). *The Experience Factory: Packaging Software Experience.* The 14th Annual Software Engineering Workshop NASA Goddard Space Flight Center, Greenbelt, Maryland, USA.

Basili, V. R. (1993). *The Experience Factory and its Relationship to Other Improvement Paradigms.* The 4th European Software Engineering Conference - ESEC '93.

Basili, V. R., Briand, L. C. y Melo, W. L. (1996). How reuse influences productivity in object-oriented systems. *Communications of the ACM*, *39*(10), 104-116.

Basili, V. R., Caldiera, G. y Cantone, G. (1992). A reference architecture for the component factory. *ACM Transactions on Software Engineering and Methodology*, *1*(1), 53-80.

Behrens, C. A. (1983). Measuring the Productivity of Computer Systems Development Activities with Function Points, *IEEE Trans. on Softw. Eng.*, *9*, 649-658.

Bello, F. (2013). La investigación tecnológica: o cuando la solución es el problema. *Revista Faces*, 1-14. http://servicio.bc.uc.edu.ve/faces/revista/a6n13/6-13-3.pdf

Blackburn, J. D., Scudder, G. D. y Van Wassenhove, L. N. (1996). Improving speed and productivity of *software* development: A global survey of *software* developers. *IEEE Transactions on Software Engineering*, *22*(12), 875-888

Bonilla, E. (2012). La importancia de la productividad como componente de la competitividad. *Revista de Investigación. 5*(2), 158-163. https://repository.uamerica.edu.co/bitstream/20.500.11839/732/1/41584611-2012-2-EF.pdf

Briand, L. C., El-Emam, K. y Bomarius, F. (1998, 19-25 de abril). *COBRA: A hybrid method for software cost estimation, benchmarking, and risk assessment.* IEEE Computer Society Press (ed.) (390-399). The 20th International Conference on Software Engineering Kyoto, Japan.

Brooks, F. P. (1995). *The Mythical Man-Month.* Addison-Wesley Longman Publishing Co. Inc., Boston, MA, USA, anniversary ed.

Brunekreef, J. y Diertens, B. (2002). Towards a user-controlled *software* renovation factory. S*cience of Computer Programming*, *45*(2-3), 175-191.

Cabach, A. (2006). *Estructura y organización de una fábrica de software* [tesis de licenciatura, Pontificia Universidad Católica de Valparaíso]. http://opac.pucv.cl/pucv_txt/Txt-7500/UCI7511_01.pdf

Cao, Q., Ching Gu, V. y Thompson, M. A. (2012). Using complexity measures to evaluate software development projects: A nonparametric approach. *The Engineering Economist: A Journal Devoted to the Problems of Capital Investment*, *57*(4), 274-283. DOI: 10.1080/0013791X.2012.729878

Card, D. N. (2006). *The challenge of productivity measurement.* Proceedings of Pacific Northwest Software Quality Conference, Portland, USA.

Cardoso, C., Bert, F. y Podestá, G. (2010). Modelos Basados en Agentes (MBA): definición, alcances y limitaciones. *Landuse, biofuels and rural development in the La Plata Basin, IDRC Grant ID* 104783-001.

Carranza, Z. (2008). Fábrica académica de *software*. *Revista digital de la Facultad de Ingeniería de Sistemas, (30)*, 7-22. https://dialnet.unirioja.es/servlet/articulo?codigo=6043125

Castañeda P. y Mauricio D. (2018). A Review of Literature About Models and Factors of Productivity in the Software Factory, *International Journal of Information Technologies and Systems Approach, 11*(1), 48-71.

Castañeda, P. y Mauricio, D. (in press). New Factors Affecting Productivity Of The Software Factory. *International Journal of Information Technologies and Systems Approach.*

Castañeda, P. y Mauricio, D. (in press). Model Based on Data Envelopment Analysis for the Measurement of Productivity in the Software Factory. *International Journal of Information Technologies and Systems Approach.*

Celina, H. y Campo-Arias, A. (2005). Aproximación al uso del coeficiente alfa de Cronbach. *Revista Colombiana de Psiquiatría, 34*(4), 572-580. http://www.scielo.org.co/pdf/rcp/v34n4/v34n4a09.pdf

Cequea, M. y Rodríguez-Monroy, C. (2012). Productividad y factores humanos. Un modelo con ecuaciones estructurales. *Interciencia, 37*(2), 121-127. https://www.redalyc.org/pdf/339/33922717007.pdf

Çetin, F. y Alabaş-Uslu, Ç. (2015). Performance evaluation of projects in *software* development. *Journal of Aeronautics and Space Technologies, 8*(2), 1-6.

Cerón, K. y Arboleda, H. (2018). Modelo de relacionamiento estratégico entre factores organizacionales y técnicos en Fábricas de Software Colombianas. *Información Tecnológica, 29*(3), 29-38. http://dx.doi.org/10.4067/S0718-07642018000300029.

Charnes, A., Cooper, W., Lewin, A. y Seiford, L. (1994). *Data Envelopment Analysis: Theory, Methodology and Applications.* Springer Dordrecht. https://doi.org/10.1007/978-94-011-0637-5

Charnes, A., Cooper, W. W. y Rhodes, E. (1978). Measuring the efficiency of decision making units. *Eur J Oper Res, 2*, 429-444.

Cheikhi, L., Al-Qutaish, R. E. y Idri, A. (2012). Software productivity: Harmonization in ISO/IEEE software engineering standards. *Journal of Software, 7*(2), 462-470.

Chicano, J., Luna, F., Nebro, A. J. y Alba, E. (2011, 12-16 de julio). *Using multi-objective metaheuristics to solve the software project scheduling problem.* 13th Annual Conference on Genetic and evolutionary computation (GECCO'11) (pp.1915-1922), GECCO'11. ACM, New York, USA.

Chung, S. H., Lee, A. H., Kang, H. Y. y Lai, C. W. (2008). A DEA window analysis on the product family mix selection for a semiconductor fabricator. *Expert Syst. with Appl., 35*, 379-388.

Clements, P. C. y Northrop, L. M. (2001). *Software product lines: Practices and patterns.* Addison-Wesley Professional.

CMMI. (2016). Capability Maturity Model Integration. CMMI Institute. http://cmmiinstitute.com/

CMMI Institute. (2018). Published Appraisal Results. https://sas.cmmiinstitute.com/pars/

Coelli, T. J., Rao, D. P., O'Donnell, C. J. y Battese, G. E. (2005). *An introduction to efficiency and productivity analysis.* Springer Science + Business Media.

Coello, R. (2018). Análisis de las ventajas y desventajas del sistema de gestión de la seguridad de la información y su influencia en la competitividad de las empresas que utilizan Cloud Computing y Big Data en Ecuador. *Revista mensual de la UIDE extensión Guayaquil*, *3*(4), 181-195 https://revistas.uide.edu.ec/index.php/innova/article/view/562

Cohen, S. G. y Bailey, D. E. (1997). What makes teams work: group effectiveness research from the shop floor to the executive suite. *Journal of Management*, *23*(3), 239-290.

Cook, W. D. y Seiford, L. M. (2009) Data envelopment analysis (DEA)—thirty years on. *Eur J Oper Res*, *192*,1-17.

Cooper, W. W., Seiford, L. M. y Zhu, J. (2004). *Handbook on data envelopment analysis*. Springer.

Cooper, W. W., Seiford, L. M. y Tone, K. (2006). *Introduction to data envelopment analysis and its uses with DEA solver software and references*. Springer Science and Business Media.

Cusumano, M. A. (1989). The software factory: A historical interpretation. *IEEE Software*, *6*(2), 23-30.

Cusumano, M., Mac Cormack, M. A., Kemerer, C. F. y Crandall, B. (2003). Software development worldwide: The state of the practice. *IEEE Software*, *20*, 28-34.

Dale, C. J. y Van der Zee, H. (1992). Software productivity metrics: who needs them? *Info. & Softw. Technol.*, *34*, 731-738.

De los Reyes, A. (2020). *Modelo DEA para análisis de fusiones de entidades productivas* [tesis de bachiller, Universidad de Sevilla]. https://hdl.handle.net/11441/99704

Díaz-Contreras, C., Aguilera-Rojas, A. y Guillén-Barrientos, N. (2014). Lógica difusa vs. Modelo de regresión múltiple para la selección de personal. *Ingeniare. Revista Chilena de ingeniería*, *22*(4), 547-559. https://scielo.conicyt.cl/scielo.php?pid=S0718-33052014000400010&script=sci_arttext

Di Penta, M., Harman, M., Antoniol, G. y Qureshi, F. (2007). The effect of communication overhead on software maintenance project staffing: a search-based approach. IEEE International Conference on Software Maintenance (ICSM 2007), 315-324.

Di Penta, M., Harman, M., Antoniol, G. y Qureshi, F. (2007). *The effect of communication overhead on software maintenance project staffing: a search-based approach*. IEEE International Conference on Software Maintenance (ICSM 2007), 315-324.

Echeverry, J. C. y Hernández, M. (2005). Posibilidades y limitantes de un cambio en la productividad de los sectores colombianos: textiles-confecciones, avícola-porcícola, siderurgia-metalmecánica y galletería-confitería-chocolatería. *CEDE Universidad de los Andes*, (40), 1-127. http://hdl.handle.net/1992/7946

Fabri, J. A., Trindade, A. L., Begosso, L. R., Lerario, I., Silveira, F. L. y Pessoa, M. S. (2004a). *Techniques for the development of a software factory: Case CEPEIN-FEMA*. 17th International Conference Software & Systems Engineering and their Applications, Paris, Francia.

Fabri, J. A., Trindade, A. L., L'errio, A., Pessoa, M. S. y Spinola, M. (2004b). *Desenvolvimento e Replicação de uma Fábrica de uma Software*. VI Simpósio Internacional de Melhoria de Processo de Software, SIMPROS, São Paulo, Brasil.

Fabri, J. A., Trindade, A. L., Durscki, R., Spinola, M. M. y Pessoa, M. S. (2005). *Proposta de um Mecanismo de Desenvolvimento e Customização de uma Fábrica de Software Orientada a Dominios*. XXXI Latin American Computing Conference, Cali, Colombia.

Fabri, J. A., Trindade, A. L., Begosso, L. R., Lerario, I. y Pessoa, M. S. (2007a). *A Organização de uma Máquina de Processo e a Melhoria do Processo de Produção de Software em um Ambiente de Fábrica.* VI Jornadas Iberoamericana de Ingeniería del Software e Ingeniería del Conocimiento, Lima, Perú.

Fabri, J. A., Trindade, A. L., Silveira, M. y Pessoa, M. S. (2007b). *O Papel do CMMI na Configuração de um Meta-Processo de Produção de Software com Características Fabris: Um Estudo de Caso.* VI Jornadas Iberoamericanas de Ingeniería del *Software* e Ingeniería del Conocimiento (pp. 375-383), Lima, Perú.

Fabri, J. A., Scheible, A. C., Moreira, P. M., Trindade, A. L., Begosso, L. R., Braun A. P. y Pessoa, M. S. (2007c). *Meta-process used for production process modeling of a software factory: The Unitech case.* Managing Worldwide Operations and Communications with Information Technology-IRMA 2007, Vancouver, Canadá.

Fabri, J. A., Trindade, A. L., Begosso, L. R., Pessoa, M. S. y L'errio, A. (2007d). *The use of the idef-0 to model the process in a software factory.* Managing Worldwide Operations and Communications with Information Technology-IRMA 2007, Vancouver, Canadá.

Farrell, M. J. (1957). The measurement of productive efficiency. *J R Stat Soc, 120,* 253-290

Fenton, N. E. y Pfleeger, S. L. (1997). Measuring productivity. En *Software metrics: A Rigorous & Practical Approach* (pp. 412-425).

Fernandes, A. A. y Teixeira, D. S. (2004). *Fábrica de software: Implantação e gestão de operações.* Editora Atlas.

Fernández, I. (2015). Felicidad Organizacional, como construir felicidad en el trabajo. Ediciones B

Fernández, Y. (2018). *Modelo computacional para la evaluación y selección de productos de software* [tesis de doctorado, Universidad de Granada]. https://repositorio.uci.cu/handle/123456789/7871.

Flitman, A. (2003). Towards meaningful benchmarking of *software* development team productivity. *Benchmarking Int J, 10*(4), 382-399.

Flores, F. y Ludlow, J. (1980). Doing and Speaking in the Office. In: G. Fick y R.H. Sprague (eds.), *Decision Support Systems: Issues and Challenges* (pp. 95-118). Pergamon Press.

Fontalvo, T., De La Hoz, E. y Morelos, J. (2017). La productividad y sus factores: incidencia en el mejoramiento organizacional. *Dimensión Empresarial, 15*(2), 47-60. DOI: http://dx.doi.org/10.15665/rde.v15i2.1375

Foss, W. B. (1993). Fast, faster, fastest development. *Computerworld, 27*(22), 81-83.

Francia, N. (2018). Análisis comparativo de frameworks open source para el nivel de productividad en los proyectos de desarrollo de software web [tesis de bachiller, Universidad Tecnológica del Perú].

Galindo-Rueda, F., y Haskel, J. (2005). Skills, Workforce Characteristics and Firm-Level Productivity: Evidence from the Matched ABI/Employer Skills Survey. *IZA Discussion Paper*, 1542. https://econpapers.repec.org/RePEc:iza:izadps:dp1542

Garces, R. (2016). *Estrategia de marketing para la introducción del servicio de fábrica de software de sysware S.A. en Guayaquil* [tesis de maestría, Universidad de Guayaquil] http://repositorio.ug.edu.ec/bitstream/redug/15634/1/EMSFSS%20V1.4.pdf

García, M. y Flores, E. (2017). Consecuencias de la cultura corporativa y rol de la comunicación interpersonal en el compromiso organizacional del personal de la Municipalidad Provincial de Arequipa-Perú. *Comuni@cción, 8*(2), 137-147. http://www.scielo.org.pe/scielo.php?script=sci_arttext&pid=S2219-71682017000200007&lng=es&tlng=es.

GMD. (2018). GMD. Lima, Perú. http://www.gmd.com.pe/

Gómez, S. y Moraleda, E. (2020). Aproximación de la ingeniería del software. Universitaria Ramón Areces.

Greenfield, J., Short, K., Cook, S., Kent, S. y Crupi, J. (2004). *Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools*. Wiley.

Griss, M. (1993). Software reuse: from library to factory. *IBM System Journal, 32*(4), 548-566.

Gueorguiev, S., Harman, M. y Antoniol, G. (2009). *Software project planning for robustness and completion time in the presence of uncertainty using multi objective search based software engineering*. The 11th Annual Conference on Genetic and Evolutionary Computation (GECCO'09) (pp. 1673-1680), ACM, Montreal, Canada (Best Paper Award).

Guisado, M., Vila, M., y Guisado, M. (2016). Innovación, capacidad productiva, formación en el puesto de trabajo y productividad. *Cuadernos de Gestión, 16*(2), 77-92. https://www.redalyc.org/articulo.oa?id=2743/274345383004

Harman, M. y Jones, B. F. (2001). Search-based software engineering. *Information and Software Technology, 43*, 833-839.

Hernández, A., Colomo R. y García, A. (2012). Productivity in software engineering: A study of its meanings for practitioners: Understanding the concept under their standpoint. *Information Systems and Technologies (CISTI)*.

International Data Corporation. (2017). *Latin America Semiannual IT Services Tracker*. http://www.gmd.com.pe/novedades/noticias/news-15 IEEE. (1992).

IEEE Standard for Software Productivity Metrics 1045, http://ieeexplore.ieee.org/servlet/opac?punumber=2858

Iribarren, I. (2020). *Eficiencia de los clubes de Atletismo en la Liga de División de Honor Masculina aplicando Dynamic DEA* [tesis de bachiller, Universidad de Sevilla]. https://hdl.handle.net/11441/101086

Jacobson I., Booch G. y Rumbaugh J. (2000). *El Proceso Unificado de Desarrollo de Software*. Pearson Educación S. A.

Jaimes, L., Luzardo, M. y Rojas, M. (2018). Factores determinantes de la productividad laboral en pequeñas y medianas empresas de confecciones del área metropolitana de Bucaramanga, Colombia. *Información tecnológica, 29*(5), 175-186. https://dx.doi.org/10.4067/S0718-07642018000500175

Jeffery, D. R. y Low, G. C. (1990). Calibrating estimation tools for *software* development, *Software Engineering Journal, 5*, 215-221.

Jeffery, R., Ruhe, M. y Wieczorek, I. (2001). *Using public domain metrics to estimate software development effort. Proceedings of METRICS'01*.

Jiang, Z., Naud´e, P. y Comstock, C. (2007). An investigation on the variation of software development productivity. *International Journal of Computer and Information Science and Engineering, 1*(2), 72-81.

Jorgensen, M., Indahl, U. y Sjoberg, D. I. (2003). Software effort estimation by analogy and 'regression toward the mean'. *J. of Systems & Software, 68*, 253-262.

Khan, R., Ahmed, I. y Faisal, M. (2014). An industrial investigation of human factors effect on software productivity: Analyzed by SEM model. *International Journal of Computer Science and Mobile Computing,*

3(5), 16-24.

Kitchenham, B. A. y Charters, S. (2007). *Guidelines for performing Systematic Literature Reviews in Software. Engineering Version 2.3*. EBSE Technical Report (EBSE-2007-01). Software Engineering Group, School of Computer Science and Mathematics, Keele University. http://www.elsevier.com/__data/promis_misc/525444systematicreviewsguide.pdf

Kremmel, T., Kubalik, J. y Biffl, S. (2010). *Multiobjective evolutionary algorithm for software project portfolio optimization*. The 12th Annual Conference on Genetic and Evolutionary Computation (GECCO'10), ACM, Portland, Oregon, USA.

Kremmel, T., Kubalik, J. y Biffl, S. (2011). Software project portfolio optimization with advanced multiobjective evolutionary algorithms. *Applied Soft Computing*, *11*, 1416-1426.

Kruchten, P. (2004). *The rational unified process: An introduction* (3.a ed.). Addison-Wesley.

Krugman, P. (1997). *El Internacionalismo Moderno*. Editorial Crítica. Li, C., Li, H. y Li, M. (2001). A software factory model based on ISO 9000 and CMM for Chinese small organizations. Second Asia-Pacific Conference on Quality Software (APAQS'01), Hong Kong, China.

Liu, W., Meng, W., Li, X. y Zhang, D. (2010). DEA models with undesirable inputs and outputs, *Ann. Operations Res.*, *173*, 177-194.

López, C., Kalichanin, I., Meda, M. E. y Chavoya, A. (2010, 12-14 de abril). *Software development productivity prediction of small programs using fuzzy logic*. Seventh International Conference on Information Technology, Las Vegas, Nevada, USA.

Loayza, N. (2016). La productividad como clave de crecimiento y el desarrollo en el Perú y el mundo. *Revista Estudios Económicos*, *31*(9). https://www.bcrp.gob.pe/docs/Publicaciones/Revista-Estudios-Economicos/31/ree-31-loayza.pdf

Machek, O., Hnilica, J. y Hejda, J. (2012). Estimating productivity of software development using the total factor productivity approach. *International Journal of Engineering Business Management*, *4*, 4-34. DOI:10.5772/52797

Madero, S. y Barboza, G. (2015). Interrelación de la cultura, flexibilidad laboral, alineación estratégica, innovación y rendimiento empresarial. *Revista Contaduría y Administración*, *60*, 735-756. http://dx.doi.org/10.1016/j.cya.2014.08.001

Mahmood, M. A., Pettingell, K. J. y Shaskevich, A. I. (1996). Measuring productivity of software projects: a data envelope analysis approach. *Decis Sci*, *27*(1), 57-80.

Manzano, A. (2017). Introducción a los modelos de ecuaciones estructurales. *Investigación en Educación Médica*, *7*(25), 67-72. https://doi.org/10.1016/j.riem.2017.11.002

Martínez, M. y Fierro, E. (2018). Aplicación de la técnica PLS-SEM en la gestión del conocimiento: un enfoque técnico práctico. *Revista Iberoamericana para la Investigación y el Desarrollo Educativo*, *8*(16), 1-35. http://dx.doi.org/10.23913/ride.v8i16.336.

Mascheroni, M., Cogliolo, M. y Irrazabal, E. (2016). Automatización de pruebas de compatibilidad web en un entorno de desarrollo continuo de software. *Universidad Nacional de la Plata*. http://sedici.unlp.edu.ar/bitstream/handle/10915/57081/Documento_completo.pdf-PDFA.pdf?sequence=1&isAllowed=y

Maxwell, K. D. (2001). Collecting data for comparability: Benchmarking software development productivity. *IEEE Software*, *18*(5), 22-25.

Montgomery Jr., P. (1981). A model of the software development process. *Journal of Systems and Software*, *2*, 237-255.

Moreira, C. I., Carneiro, C., Pires, C. S. y Bessa, A. (2010). A practical application of performance models to predict the productivity of projects. In T. Sobh (ed.), *Innovations and advances in computer sciences and engineering* (pp. 273-277).

Motta, J., Morero, H. y Borrastero, C. (2017). La industria del *software*: la generación de capacidades tecnológicas y el desafío de elevar la productividad sistémica. En M. Abeles, M. Cimoli y P. Lavarello (eds.), *Manufactura y cambio estructural: Aportes para pensar la política industrial en la Argentina*. Cepal. https://repositorio.cepal.org/bitstream/handle/11362/42624/S1601348_es.pdf?sequence=1&isAllowed=y

Myrtveit I. y Stensrud E. (1999). *Benchmarking COTS projects using data envelopment analysis*. Proceedings Sixth International *Software* Metrics Symposium, Washington, D.C.

Nemur, L. (2016). *Productividad: consejos y atajos de productividad para personas ocupadas*. Editorial Bebelcube. https://books.google.es/books?hl=es&lr=&id=sh0aDAAAQBAJ&oi=fnd&pg=PT4&dq=productividad&ots=LMHhlPu3jb&sig=K7gejhXJmofAhOEVdoR16-3CH4A#v=onepage&q=productividad&f=false

Neyra, L., Espinoza, M. y Samaniego, M. (2017, 3-5 de julio). *Gestión de requerimientos a través del modelo de fábrica de software aplicando metodologías ágiles*. Séptima Conferencia de Directores de Tecnología de Información, TICAL 2017 Gestión de las TICs para la Investigación y la Colaboración, San José, Costa Rica. https://documentos.redclara.net/bitstream/10786/1281/1/105%20Gesti%C3%B3n%20de%20Requerimientos%20a%20trav%C3%A9s%20del%20Modelo%20de%20F%C3%A1brica%20de%20Software%20aplicando%20Metodolog%C3%ADas%20%C3%81giles.pdf

Nomura, L., Spinola, M., Hikage, O. y Tonini, A. C. (2006, 9-11 de octubre). FS-*MDP: Um Modelo de Definição de Processos de Fábrica de Software*. XXVI Encontro Nacional de Engenharia de Producão, Fortaleza, Brazil. http://www.abepro.org.br/biblioteca/enegep2006_tr450304_7257.pdf

Nolasco, M., Durski, V. y Coelho, A. (2020). Análise entoltória de dados (DEA) aplicada a portos. O estado da arte. *Estratégia & Negócios*, *13*(2), 77-109. DOI: http://portaldeperiodicos.unisul.br/index.php/EeN/article/view/7553

Norden, P. (1977). *Project life cycle modeling: background and application of the life cycle curves*. U.S. Army Computer Systems Command.

Norden, P. V. (1958). Curve fitting for a model of applied research and development scheduling. *IBM Journal of Research and Development*, *2*, 232-248.

Noriega, R. (2015). *El proceso de desarrollo de software*. Editorial IT Campus Academy.

Nwelih E. y Amadin I. F. (2008). Modeling software reuse in traditional productivity model. *Asian Journal of Information Technology*, *7*, 484-488

Osorio, N. (2016). *Diseño e implementación de un sistema de matrícula web usando software libre en el centro educativo "España" distrito - Breña 2013* [tesis de licenciatura, Universidad de Ciencias y Humanidades]. https://repositorio.uch.edu.pe/bitstream/handle/uch/82/Osorio_Alvarez_Neil_Angelo.pdf?sequence=1&isAllowed=y

Pai, D. R., Subramanian, G. H. y Pendharkar, P. C. (2015). Benchmarking software development productivity of CMMI level 5 projects. *Information Technology and Management*, *16*(3), 235-251.

Paiva, E., Barbosa, D., Lima, R. y Albuquerque, A. (2010). Factors that influence the productivity of *software*

developers in a developer view. In: T. Sobh y K. Elleithy K. (eds.), *Innovations in Computing Sciences and Software Engineering* (pp. 99-104).

Paradi, J. C., Reese, D. N. y Rosen, D. (1997). Applications of DEA to measure the efficiency of software production at two large Canadian banks. *Ann Oper Res, 73*, 91-115.

Parthasarathy, S. y Anbazhagan, N. (2008). Evaluating ERP projects using DEA and regression analysis. *Int J Bus Inf Syst, 3*(2), 140-157.

Pendharkar, P. C. y Rodger, J. A. (2007). An empirical study of the impact of team size on *software* development effort. *Information Technology and Management, 8*, 253-262.

Pendharkar, P. C. y Rodger, J. (2009). The Relationship between software development team size and software development cost. *Commun ACM, 52*(1), 141-144.

Pérez, H. (2016). *El impacto de la Lean Six Sigma en organizaciones latinoamericanas y sus factores críricos de éxito* [tesis de doctorado, Universidad Antropológica de Guadalajara]. http://rei.iteso.mx/handle/11117/3873

Pessoa, M. S., Fabri, J. A., L'errio, A., Spinola, M. y Begosso, A. (2004). *Desenvolvimento e Replicação de uma Fábrica de uma Software.* IV Jornadas Iberoamericanas de Ingeniería del *Software* e Ingeniería del Conocimiento, Madrid, España.

Petersen, K. (2011). Measuring and predicting software productivity: A systematic map and review. *Information and Software Technology, 53*(4), 317-343.

Piattini, M. y Garzs, J. (2010). *Fábricas de software: Experiencias, Tecnologías y Organización* (2.ª ed.). RA-MA Editorial.

Piattini, M., Caballero, I. y García, F. (2006). *Calidad de los Sistemas Informáticos.* RA-MA Editorial.

Piattini, M. y Daryanani, S. (1995). *Elementos y Herramientas en el Desarrollo de los Sistemas de Información. Una visión actual de la tecnología CASE.* RA-MA Editorial.

Pillai, K. y Sukumaran, V. S. (1997). A model for software development effort and cost estimation. *IEEE Transactions on Software Engineering, 23*, 485-497.

Polo, M., Piattini, M. y Ruiz, F. (2003). *Advances in Software Maintenance Management: Technologies and Solutions.* Idea Group Publishing.

Premraj, R., Shepperd, M., Kitchenham, B. y Forselius, P. (2005, 19-22 de setiembre). *An empirical analysis of software productivity over time.* 11th IEEE International *Software.* Metrics Symposium (Metrics05), Como, Italy.

Ramanathan, R. (2002). *Introductory Econometrics with Applications* (5.ª ed). South-Western.

Reina, R. (2016). Productividad de recursos humanos, innovación de producto y desempeño exportador: Una investigación empírica. *Intangible Capital, 12*(2), 619-641. http://dx.doi.org/10.3926/ic.746

Reyes, J. (2010). El residuo de Solow [versión electrónica]. *Revista de Economía Institucional, 12*(23).

Rockwell, R. y Gera, M. H. (1993, 7-9 de julio). *The Eureka Software Factory CoRe: A conceptual reference model for software factories. Software.* Engineering Environments Conference, UK, United Kingdom.

Rodger, J. A., Pankaj, P. y Nahouraii, A. (2011). Knowledge management of software productivity and

development time. *Journal of Software Engineering and Applications*, *4*(11), 609-618.

Rodríguez, D., Sicilia, M. A., García, E. y Harrison, R. (2011). Empirical findings on team size and productivity in *software* development. *Journal of Systems and Software*, *85*(3), 562-570.

Romo, D. y Andel, G. (2005). Sobre el concepto de competitividad. *Revista Comercio Exterior*, *55*(3).

Sánchez, H. H. y Reyes, C. (2015). *Metodología y Diseños en la Investigación Científica.* Business Support.

Santamaría, J. (2016). El diseño como motor de productividad, *Revista 180*, (37). http://dx.doi.org/10.32995/rev180.Num-37.(2016).art-14

SBOK. (2016). *A Guide to the Scrum Body of Knowledge.* SCRUMStudy. https://www.scrumstudy.com/

Scacchi, W. (1995). Understandings *Software* Productivity. *Advances in Software Engineering and Knowledge Engineering*, *4*, 37-70. World Scientific Press.

Sentas, P., Angelis, L., Stamelos, I. y Bleris, G. (2005). Software productivity and effort prediction with ordinal regression. *Info. & Softw. Technol.*, *47*, 17-29.

Simmons, D. B. (1991). Communications: a *software* group productivity dominator. *Software Engineering Journal*, *6*(6), 454-462. DOI: 10.1049/sej.1991.0044

Smith, R. K., Hale, J. E. y Parish, A. S. (2001). An empirical study using task assignment patterns to improve the accuracy of *software* effort estimation. *IEEE Transactions on Software Engineering*, *27*, 264-271.

Sridhar, V., Paul, R., Nath, D. y Kapur, K. (2007, 25-27 de julio). *Analyzing factors that affect performance of global virtual teams.* 2nd International Conference on Management of Soft factors affecting performance 203 Globally Distributed Work 2007, Indian Institute of Management (IIM), Bangalore, India.

Stensrud, E. y Myrtveit, I. (2003). Identifying high performance ERP projects. *IEEE Trans Comput*, *29*(5), 398-416.

Swanson, K., McComb, D., Smith, J. y McCubbrey, D. (1991). The application software factory: Applying total quality techniques to systems development. *MIS Quarterly*, *15*(4), 567-579.

Subramanian, G. H. y Zarnich, G. E. (1996). An examination of some *software* development effort and productivity determinants in ICASE tool projects. J*ournal of Management Information Systems*, *12*(14), 143-160.

Sudhakar G., Farooq A. y Patnaik S. (2011). Soft factors affecting the performance of *software* development teams, *Team Performance Management*, *17*(3/4), 187-205

Sudhakar, G., Farooq, A. y Patnaik, S. (2012). Measuring productivity of software development teams. *Serbian Journal of Management*, *7*(1), 65-75.

Swanson, K., Kent, D., McComb, J. y Dave, D. (1991). The Application Software Factory: Applying Total Quality Techniques to Systems Development. *MIS Quarterly*, *15*(4), 567-579.

Tamayo, L., Lazo, Y. y Ramírez, J. (2019). La gestión de reutilización de *software* en el Modelo de la Calidad para el Desarrollo de Aplicaciones Informáticas en Cuba. *Gestión Integral de Proyectos.* https://www.researchgate.net/profile/Jose_Ramirez_Perez2/publication/330771198_Proceso_de_diseno_del_software_para_un_modelo_de_la_calidad_en_Cuba/links/5fc4d296a6fdcc6cc6852e0a/Proceso-de-diseno-del-software-para-un-modelo-de-la-calidad-en-Cuba.pdf

TechAmerica. (1999). ANSI-EIA-632 *Standard*. Processes for Engineering a System.

Tello, M. (2017). Innovación y productividad en las empresas de servicios y manufactureras: el caso de Perú. *Cepal*, (121).

Tonkay, G. L. (2008). *Productivity in Encyclopedia of Science & Technology*. McGraw-Hill.

Trendowicz, A. (2007). Factors influencing *software* development productivity—state of the art and industrial experiences. *Technical Report 08.07/*E, Fraunhofer IESE, Kaiserslautern, Germany.

Trendowicz, A. y Munch, J. (2009). Factors Influencing *Software* Development Productivity-State-of-the-art and Industrial Experiences. *Advances in Computers*, *77*, 185-241.

Trespalacios, J., Vázquez, R. y Bello, L (2005). *Investigación de mercados: métodos de recogida y análisis de la información para la toma de decisiones de marketing*. Ediciones Paraninfo.

Trujillo, Y. T. (2007). Modelo de factoría de *software* aplicando inteligencia. *Serie Científica de la Universidad de las Ciencias Informáticas*, *1*(1).

Unluturk, M. S. y Kurtel, K. (2015). Quantifying productivity of individual software programmers: Practical approach. *Computing and Informatics*, *34*(4), 959-972.

Vazquez, M. (2019). *Propuesta de mejora en el proceso de gestión del área de servicio al cliente y gestión de cobro de refinancia Colombia utilizando la metodología Lean Six Sigma* [tesis de pregrado, Universidad Católica de Colombia]. https://repository.ucatolica.edu.co/handle/10983/23988

Van den Brand, M., Sellink, A. y Verhoef, C. (2000). Generation of components for software renovation factories from context-free grammars. *Science of Computer Programming*, *36*(2-3), 209-266.

Verner, J., Evanco, W. y Cerpa, N. (2007). State of the practice: an exploratory analysis of schedule estimation and software project success prediction. *Information and Software Technology*, *49*, 181-193.

Villa, G. (2003). *Análisis por Envoltura de Datos (DEA): Nuevos Modelos y Aplicaciones* [tesis de doctorado, Universidad de Sevilla]. http://hdl.handle.net/11441/15412

Wagner, S. y Ruhe, M. (2008a). *A structural review of productivity factors in software development*. Technical Report, Technische Universität München.

Wagner, S. y Ruhe, M. (2008b). *A systematic review of productivity factors in software development*. 2nd International Workshop on Software Productivity Analysis and Cost Estimation (SPACE'08), State Key Laboratory of Computer Science, Institute of *Software*, Chinese Academy of Sciences.

Wegner, D. M. (1987). Transactive memory: a contemporary analysis of the group mind. In B. Mullen & G. Goethals (eds.), *Theories of Group Behavior* (pp. 185-208). Springer-Verlag.

Wray, B. y Mathieu, R. (2008). Evaluating the performance of open source software projects using data envelopment analysis. *Inf Manag Comput Secur*, *16*(5), 449-462.

Yang, Z. y Paradi, J. C. (2004). DEA evaluation of a Y2K software retrofit program. *IEEE Trans Eng Manag*, *51*(3), 279-287.

Yanosky, M. M. (2005). *Modelo funcional de la Factoría de Software de la UCI para la línea Carrefour*.

Yilmaz, M. y O'Connor, R. V. (2011). *An empirical investigation into social productivity of a software process: An approach by using the structural equation modeling. European Conference on Software* Process Improvement

2011: Communications in Computer and Information Science.

Zhan, J., Zhou, X. y Zhao, J. (2012). Impact of software complexity on development productivity. *International Journal of Software Engineering and Knowledge Engineering, 22*(8), 1103-1122.



EDITORIAL
NAVEGANTE