ESCUELA DE POSGRADO
UNIVERSIDAD CÉSAR VALLEJO

Transformación de modelos para mejorar la calidad del modelo software en estudiantes de ingeniería Universidad César Vallejo Lima 2016

TESIS PARA OPTAR EL GRADO ACADÉMICO DE:

Maestro en Gestión de Tecnologías de Información

AUTOR:

Br. Chunga Huatay, Edwin José

ASESOR:

Mg. Aramburu García Freddy Edgard

SECCIÓN

Ingeniería de Sistemas

LÍNEA DE INVESTIGACIÓN

Proyectos de tecnologías de información

PERÚ – 2018

Página de Jurado

_____

Dr. Carhuancho Mendoza Irma Milagros

Presidente

_____

Mg. Torres Cabanillas Luis

Secretario

_____

Mg. Aramburu García Freddy Edgard

Vocal

Dedicatoria

A Dios por el don de la vida y habernos permitido llegar hasta esta etapa de nuestra vida, por el cuidado de nuestra salud para el cumplimiento de nuestros objetivos. A mi familia, gracias por su apoyo incondicional.

Agradecimiento

A la Universidad César Vallejo mi alma mater a mi asesor con su enseñanza siempre incentivándome a seguir adelante a la Escuela de Ingeniería de Sistemas por todo lo aprendido y a mi familia por entenderme en aquellos momentos de ausencia para cumplir las metas.

Declaratoria de autenticidad

Yo, Chunga Huatay, Edwin José con DNI 16594298 , estudiante del Programa de Maestría en Gestión de Tecnologías de Información de la Escuela de Posgrado de la Universidad César Vallejo, con la tesis titulada “Transformación de modelos para mejorar la calidad del modelo software en estudiantes de ingeniería Universidad César Vallejo Lima 2016” declaro bajo juramento que:

1) La tesis es de mi autoría
2) He respetado las normas internacionales de citas y referencias para las fuentes consultadas. Por tanto, la tesis no ha sido plagiada ni total ni parcialmente.
3) La tesis no ha sido auto plagiado; es decir, no ha sido publicada ni presentada anteriormente para obtener algún grado académico previo o título profesional.
4) Los datos presentados en los resultados son reales, no han sido falseados, ni duplicados, ni copiados y por tanto los resultados que se presenten en la tesis se constituirán en aportes a la realidad investigada.

De identificarse la falta de fraude (datos falsos), plagio (información sin citar a autores), auto plagio (presentar como nuevo algún trabajo de investigación propio que ya ha sido publicado), piratería (uso ilegal de información ajena) o falsificación (representar falsamente las ideas de otros), asumo las consecuencias y sanciones que de mi acción se deriven, sometiéndome a la normatividad vigente de la Universidad César Vallejo.

Lima, Marzo del 2017

Chunga Huatay, Edwin José

DNI 16594298

Presentación

Señores miembros del jurado calificador:

Dando cumplimiento a las normas del Reglamento de Grados y Títulos para la elaboración y la sustentación de la tesis de la sección de Postgrado de la Universidad “Cesar Vallejo”, para optar el grado de Magister en Gestión de Tecnologías de Información, presento la tesis titulada: “Transformación de modelos para mejorar la calidad del modelo software en estudiantes de ingeniería Universidad César Vallejo Lima 2016”. La investigación tiene por finalidad, demostrar que la transformación de modelos mejora la calidad del modelo software en estudiantes de ingeniería.

El documento consta de ocho capítulos: el primer capítulo denominado introducción, en la cual se describen los antecedentes, el marco teórico de las variables, la justificación, la realidad problemática, la formulación de problemas, la determinación de los objetivos y las hipótesis. El segundo capítulo denominado marco metodológico, el cual comprende la operacionalización de las variables, la metodología, tipos de estudio, diseño de investigación, la población, muestra y muestreo, las técnicas e instrumentos de recolección de datos y los métodos de análisis de datos. En el tercer capítulo se encuentran los resultados, el cuarto capítulo la discusión, en el quinto capítulo las conclusiones, en el sexto capítulo las recomendaciones, en el séptimo capítulo las referencias bibliográficas y por último, en el octavo capítulo, los anexos.

Los resultado según la prueba de distribución presentan un aumento de 13.5% en el nivel alto de la calidad del software, una disminución del 9.8% en el nivel regular de la calidad del software, un aumento del 5.3% en el nivel bajo de la calidad del software y según la prueba de Wilcoxon que con un error de 0.009 (0.09%) la calidad del modelo software mejora al aplicar la transformación de modelos

Espero señores miembros del jurado que esta investigación se ajuste a las exigencias establecidas por la Universidad y merezca su aprobación.

Índice

Lista de tablas

Página

Lista de figuras

Resumen

Esta investigación se titula “Transformación de modelos para mejorar la calidad del modelo software en estudiantes de ingeniería Universidad Cesar Vallejo Lima 2016”, ha tenido como objetivo demostrar que la transformación de modelos mejora la calidad del modelo software en estudiantes de ingeniería.

Es de método hipotético deductivo, tipo experimental – longitudinal, diseño cuasi experimental, la muestra fue constituida por 67 estudiantes de ingeniería, tipo muestreo no probabilístico, estrategia de selección aleatorio estratificado, técnica de recopilación de datos “Encuesta”, tipo de instrumentos “Test”, confiabilidad prueba piloto y coeficiente de “Alfa de Cronbach”, la validación mediante el “Juicio de Expertos”.

Los resultado según la prueba de distribución presentan un aumento de 13.5% en el nivel alto de la calidad del software, una disminución del 9.8% en el nivel regular de la calidad del software, un aumento del 5.3% en el nivel bajo de la calidad del software y según la prueba de Wilcoxon que con un error de 0.009 (0.09%) la calidad del modelo software mejora al aplicar la transformación de modelos.

Palabras claves: Transformación de modelos, Calidad del modelo software.

Abstract

This research is titled "Transformation of models to improve the quality of software model in engineering students University Cesar Vallejo Lima 2016", has aimed to show that the transformation of models improves the quality of software model in engineering students.

It is a hypothetical deductive method, experimental type - longitudinal, quasi experimental design, the sample consisted of 67 engineering students, non - probabilistic sampling type, stratified random selection strategy, data collection technique "Survey", type of instruments "Test ", Reliability pilot test and coefficient of" Cronbach's Alpha ", validation through the" Expert Judgment ".

The results according to the distribution test show a 13.5% increase in the high level of software quality, a decrease of 9.8% in the regular level of software quality, a 5.3% increase in the low level of quality Of the software and according to the Wilcoxon test that with an error of 0.009 (0.09%) the quality of the software model improves when applying the transformation of models.

Key words: Model transformation, Software model quality.

I. Introducción

1.1 Antecedentes

1.1.1 Antecedentes internacionales

A nivel internacional, Escalone (2006) en su tesis: “Estudio comparativo de los Modelos y Estándares de Calidad del Software”. De tipo experimental, llegó a las siguientes conclusiones: (a) Es crucial evaluar la calidad del proceso y producto software, (b) El software se evalúa aplicando métricas, las cuales permiten cuantificar los resultados observados del tratamiento del software, (c) Para una mejor administración de la calidad del software deben usarse modelos y estándares de calidad del software.

El autor, Meliá (2012) en su tesis: “Un Método de Desarrollo Dirigido por Modelos de Arquitectura Web”. De tipo experimental, llegó a las siguientes conclusiones: (a) El paradigma de ingeniería dirigido por modelos, donde los modelos y las transformaciones son computables, proporciona una trazabilidad precisa desde el análisis hasta la implementación y permite acelerar el proceso de desarrollo, (b) Un metamodelo permite establecer restricciones a la hora de especificar modelos, vincular los diferentes modelos y obtienen la información necesaria cuando se definan las transformaciones.

El autor, Díaz (2008) en su tesis: “Plan de calidad para la mejora del desarrollo de software”. De tipo experimental, llegó a las siguientes conclusiones: (a) La disminución de errores posterior a la entrega del producto, se logra como consecuencia de la detección temprana de dichos errores, (b) El modelo de proceso software permite lograr que la organización presente un nivel de madurez alto, cuyo éxito no depende del personal de forma individualizada sino de la propia capacidad de producir software, (c) La productividad media de los proyectos se incrementa como consecuencias del uso de herramientas de productividad y reutilización de componentes.

1.1.2 Antecedentes nacionales

A nivel nacional, Llacctahuaman (2015), en su tesis: “Sistema Integral para mejorar la Calidad de Información en la Recaudación Tributaria de la Municipalidad distrital de El Tambo”. De tipo experimental, llegó a las siguientes conclusiones: (a) El uso de la conjunción de las metodologías de desarrollo de software RUP y XP aseguraron que el Sistema implementado cuente con los modelos necesarios, pruebas completas sobre diversos puntos, la aplicación de buenas prácticas en la fase de construcción y sobretodo la ejecución del ciclo completo de vida desde la concepción hasta el despliegue y mantenimiento de dicho software, (b) Con la implementación del Sistema Integral se ha incrementado la calidad de la información que se maneja y el nivel de servicio a los contribuyentes debido al nivel de confiabilidad, exactitud y seguridad de la información, haciéndolo más accesible, completo y confiable.

El autor, Castro (2013), en su tesis: “Aplicación del Modelo CMMI nivel 2, SCRUM y PSP para el desarrollo de software”. De tipo experimental, llegó a las siguientes conclusiones: (a) El modelo desarrollado proporciona técnicas para la estimación de tiempos y costos más exactos, la técnica que se propone (Estimación de Puntos de Casos de Uso) arroja resultados acertados en cuanto al esfuerzo requerido para desarrollar el producto. También el modelo proporciona herramientas y técnicas más eficaces para la estimación de tamaños del producto, tiempos de desarrollo y esfuerzo para la construcción del producto, (b) Al aplicar el modelo, los productos obtenidos son de mayor calidad ya que se siguen procesos para captar la necesidad del usuario y procesos para desarrollar los requerimientos, (c) El modelo cuenta con un proceso de aseguramiento de la calidad, que permite verificar que los integrantes del equipo utilicen los procesos definidos. Esta revisión permite identificar oportunidades de mejora de los procesos y encontrar aquellos inconvenientes que no permiten seguir los procesos definidos lo que podría conllevar a disminuir la calidad del producto final.

El autor, Dioses (2012), en su tesis: “Desarrollo de la metodología para la implementación de un sistema de gestión de calidad aplicado al software de computadora”. De tipo experimental, llegó a las siguientes conclusiones: (a) Se ha

demostrado que emplear un marco de trabajo propio, guía a las usuarios u organizaciones a llevar a cabo lo recomendado por los estándares internacionales, (b) Se acredita que un modelo muestra más que mil palabras y permite entender lo que sucede, (c) Se ha contrastado que la calidad del software es aplicado a todo el proceso de construcción de software.

Contribuyo indicando que el desarrollo de software requiere de evaluación constante y medidas adecuadas para su logro, permitiendo así disminuir sus errores, mediante el uso de nuevos paradigmas de desarrollo en especial la del desarrollo guiada por los modelos.

1.2 Bases teóricas y fundamentación científica, técnica o humanística

1.2.1 Bases teóricas de la variable transformación de modelos

Teoría de sistemas.

Según, Von (1986) al respecto afirma que:

> Es ante todo un campo matemático que proporciona técnicas, en parte modernas y muy especificadas. Fuertemente asociada a la ciencia de la informática, y orientado más que nada por el imperativo de verlo como un nuevo tipo de problema (p.43).
>
> Contribuyo indicando que la teoría de sistemas es una ciencia moderna vinculada al desarrollo de software, enfocada en un paradigma moderno de solución de problemas.

Por otro lado, Johansen (1993) complementa al respecto:

> “Es una potente herramienta que permite la explicación de los fenómenos que se presentan en la realidad igualmente hace posible la predicción de la conducta futura de esa realidad” (p.1).
>
> Contribuyo indicando que la teoría de sistemas permite dar entendimiento de la realidad actual y predecir los sucesos de una realidad futura.

Finalmente, Saravia (1995) complementa al respecto:

> Es la historia de una filosofía y un método para analizar y estudiar el mundo real y desarrollar modelos, a partir de los cuales puedo

> mostrar una aproximación paulatina a la percepción de una parte de esa realidad que es el Universo, configurando un modelo de la misma no aislado del resto al que llamaremos sistema (p.9).

Contribuyo indicando que la teoría general de sistemas ayuda a entender la realidad a partir de modelos denominados sistemas.

Desarrollo de software dirigido por modelos.

Según, Higuera (2012) al respecto afirma lo sostiene:

> Es un paradigma de ingeniería de software que propone el uso de modelos para conducir las distintas fases del desarrollo. Además promete elevar los niveles de abstracción y automatización. Propone el uso de modelos como elementos de primera clase (que pueden ser procesados por un computador o herramienta) para el desarrollo de software. De acuerdo a lo anterior, estos modelos se pueden utilizar para preparar los conceptos de negocio y la solución del problema en sus diferentes niveles (p.100).

Contribuyo indicando que el desarrollo de software moderno requiere de nuevos paradigmas que usen herramientas que remplacen el esfuerzo humano y fortalezcan su creatividad

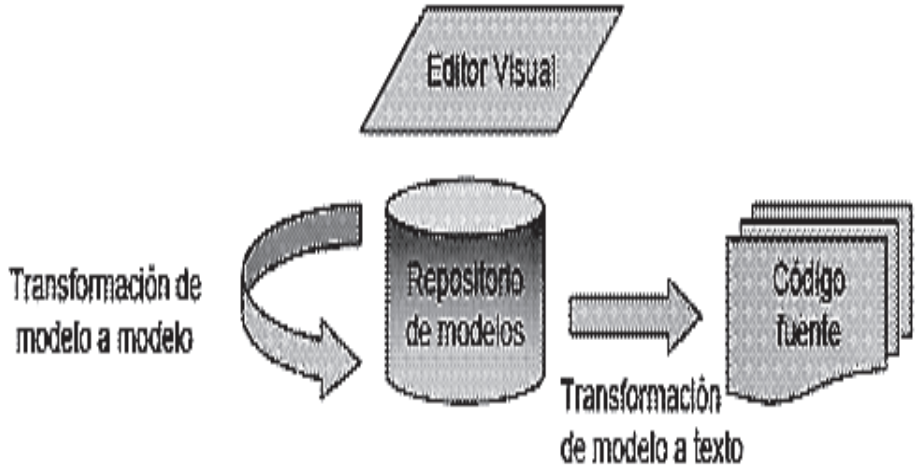


Figura 1: Herramientas de soporte para MDD
Nota: Adaptado del “Desarrollo de Software Dirigido por Modelos” por Pons, Giandini, Perez, 2010. Buenos Aires, Argentina:Edulp

Por otro lado, Pons (2010) complementa al respecto:

> “El proceso de software guiado por modelos MDD se ha transformado en un nuevo estándar de desarrollo de software. Promete perfeccionar el proceso de construcción de software apoyándose en un proceso gobernado por modelos y apoyado por poderosas herramientas” (p.28).
>
> Contribuyo indicando que el desarrollo de software requiere de estándares, modelos y herramientas de desarrollo sofisticadas.

Transformación de modelos.

Según, Pons (2010) al respecto afirma lo siguiente:

> Es la obtención automática de un modelo objetivo desde un modelo básico, respetando a una definición de transformación. Esta definición es un grupo de reglas de transformación que juntas explican como un modelo en el lenguaje básico puede ser transformado en un modelo en el lenguaje objetivo. Una regla de transformación es una explicación de cómo una o más construcciones en el lenguaje básico puede ser transformadas en una o más construcciones en el lenguaje objetivo (p.53).
>
> Contribuyo indicando que la transformación de modelos es obtener un modelo solución a partir de un modelo problema mediante el empleo de reglas de transformación.

Por otro lado, Calero, Moraga y Piattini (2012) complementan al respecto:

> “Establece una agrupación de normas que detallan como un modelo mencionado en un dialecto origen puede ser convertido en un modelo en un dialecto destino” (p.227).
>
> Contribuyo indicando que la transformación de modelos es la obtención de un modelo final a partir del tratamiento de un modelo inicial respetando reglas y lenguajes de transformación.

Figura 2: Definición de transformación entre lenguajes
Nota: Adaptado del “Desarrollo de Software Dirigido por Modelos” por Pons, Giandini, Perez, 2010. Buenos Aires, Argentina:Edulp

Modelo de software.

Según, Pons (2010) al respecto afirma lo siguiente:

> En el contexto científico un modelo puede ser una entidad, un esquema o una entidad concreta. Es una estructuración abstracta o concreta a escala de un proceso o sistema, con el fin de evaluar su naturaleza, construir o demostrar hipótesis o supuestos y facilitar un mejor entendimiento del portento real al cual el modelo representa y permitir así refinar los diseños, antes de comenzar la construcción de las obras u objetos reales (pp.39-40).
>
> Contribuyo indicando que un modelo es una abstracción de una cierta realidad y permite su representación para su estudio.

Por otro lado,Bennet, McRobb y Farmer (2007) complementan al respecto:

> “En cualquier proyecto de desarrollo que aspire a producir artefactos útiles, el objetivo principal del análisis y del diseño se centra en los modelos (aunque el objetivo final sea tener un sistema” (p.104).
>
> Contribuyo indicando que un modelo es necesario en el desarrollo de software.

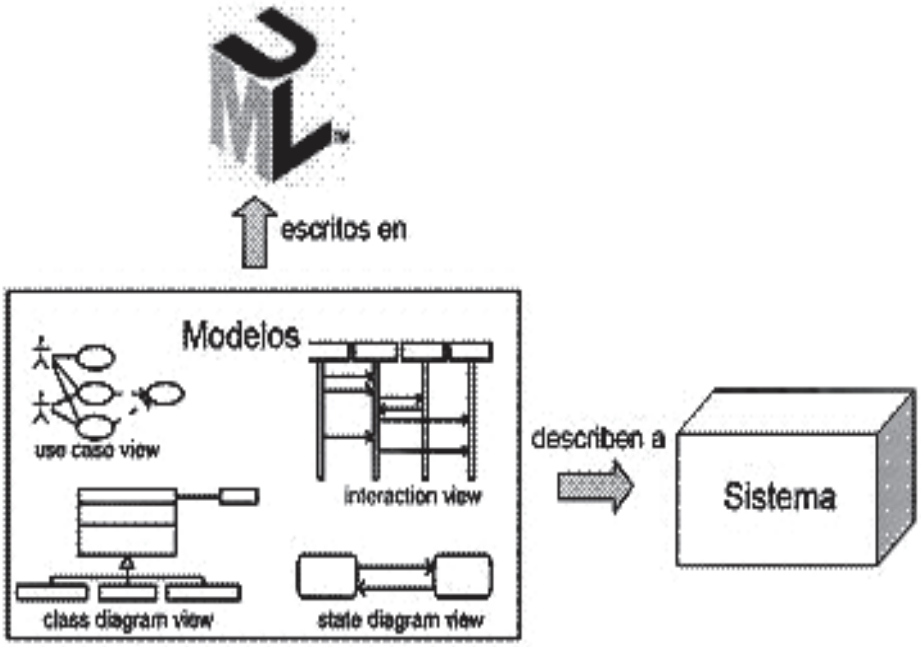


Figura 3: Diferentes modelos que conforman un sistema
Nota: Adaptado del “Desarrollo de Software Dirigido por Modelos” por Pons, Giandini, Perez, 2010. Buenos Aires, Argentina:Edulp

Ciclo de desarrollo guiado por modelos

Según, Pons (2012) al respecto afirma lo siguiente:

> Los sistemas de software involucran varios modelos independientes a diferentes niveles de abstracción (análisis, diseño, implementación), representando diferentes partes del sistema, diferentes requisitos, o diferentes tareas. En muchos casos, es posible generar un modelo a partir de otro, por ejemplo pasando del modelado de requisitos al modelado de análisis, o del modelo de la implementación al modelo de validación (p.30).

Contribuyo indicando que el desarrollo dirigido por modelos sigue los mismos pasos del desarrollo clásico con la diferencia que en cada etapa se transforman los modelos en distintos niveles de abstracción.

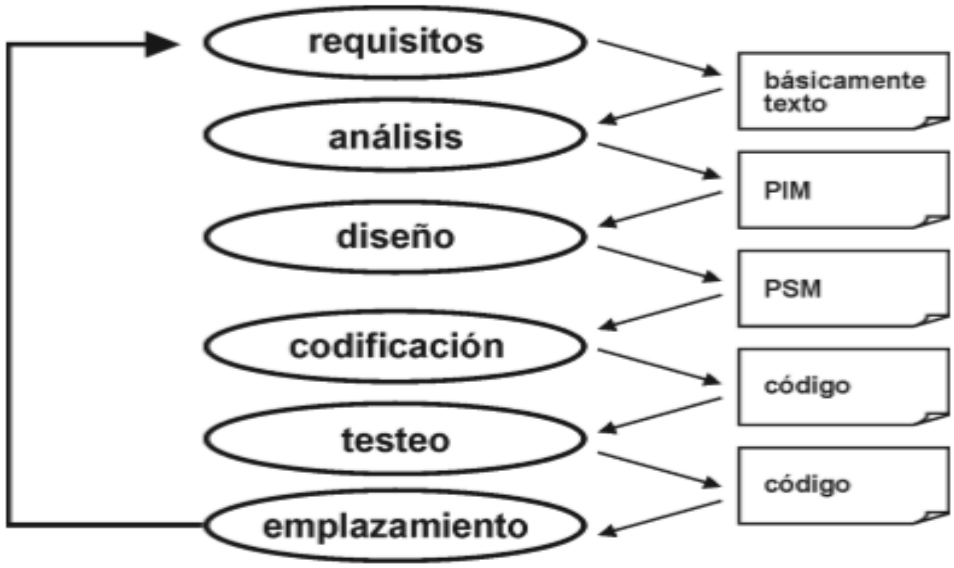


Figura 4: Fases del desarrollo de software guiado por modelos.
Nota: Adaptado del “Desarrollo de Software Dirigido por Modelos” por Pons, Giandini, Perez, 2010. Buenos Aires, Argentina:Edulp

Empleo de modelos en la construcción de software.

Según, Pons (2012) al respecto plantea lo siguiente:

> Durante la construcción de software guiado por modelos MDD se crean diferentes modelos. Los modelos de análisis identifican solo los requerimientos críticos del sistema software, especificando lo que el software hará independiente de cómo se programe. Por otro lado, los modelos de diseño representan decisiones sobre el plano de desarrollo (tratado a objetos, aplicado en componentes, basado en aspectos, etc.), los elementos relacionados del sistema (distintos modelos de arquitectura). Finalmente, los modelos de programación muestran como el software será elaborado en el ámbito de un ambiente de codificación seleccionado (plataforma, sistema operativo, base de datos, lenguajes de programación etc.). Si bien algunos modelos pueden clasificarse claramente como un modelo de análisis, o de diseño o de implementación por ejemplo, un diagrama de casos de uso es un esquema de análisis, mientras

que un modelo de interacción entre objetos es un esquema de diseño y un diagrama de despliegue es un esquema de implementación. En general, esta clasificación no depende del modelo en sí mismo sino de la interpretación que se de en un cierto proyecto a las etapas de análisis, diseño e implementación (p.26).

Contribuyo indicando que en el desarrollo administrado por modelos se crean modelos para el análisis, diseño e implementación, en cada etapa se transforman dichos modelos desde el más abstracto o externo hasta el concreto o interno.

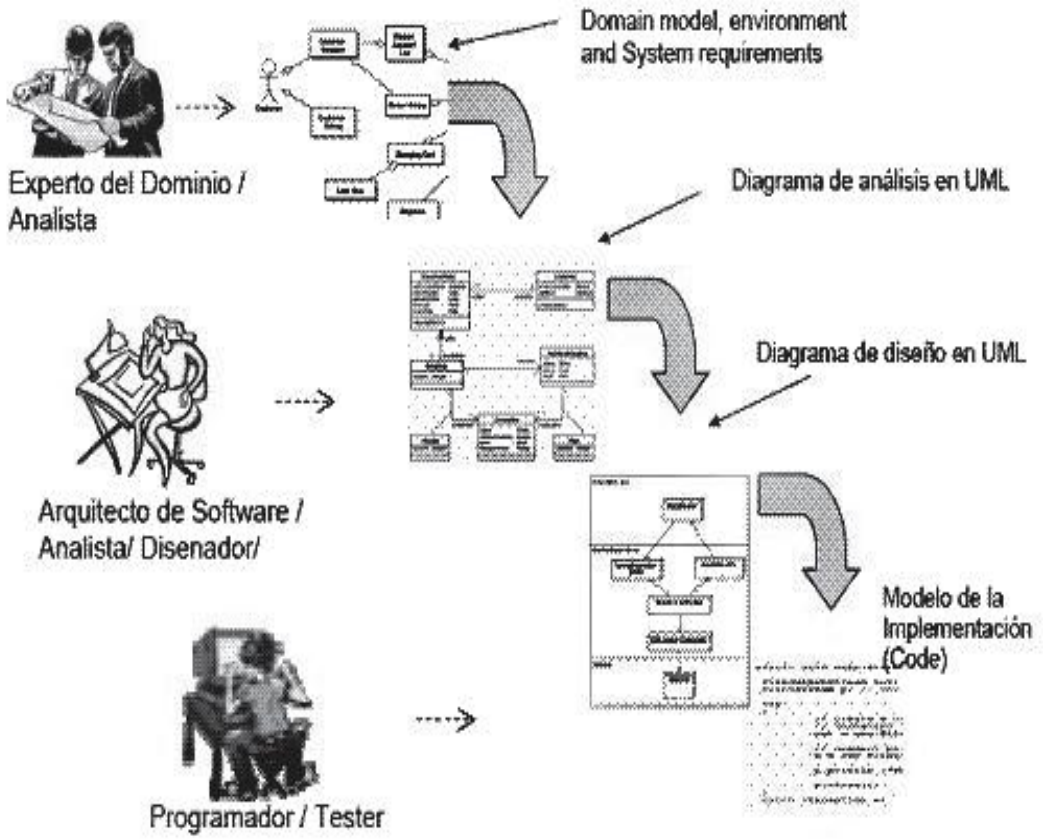


Figura 5: Diferentes modelos a lo largo del proceso de desarrollo
Nota: Adaptado del “Desarrollo de Software Dirigido por Modelos” por Pons, Giandini, Perez, 2010. Buenos Aires, Argentina:Edulp.

Modelos producidos en el proceso de desarrollo dirigido por modelos.

Según, Pons (2010) al respecto sostiene lo siguiente:

> "En el MDD los diagramas se van generando desde los más imaginarios a los más reales a través de procesos de conversión y/o simplificación, hasta obtener el programa fuente empleando una última conversión" (p.40).
>
> Contribuyo indicando que los modelos producidos en el MDD se inician de sistemas abstractos y concluyen en sistemas concretos.

Por otro lado, Pons (2010) complementa al respecto:

> "Algunos esquemas representan al software de manera autónoma de los fundamentos técnicos que involucran su codificación sobre una tecnología de software, mientras que otros esquemas tienen como objetivo primario representar tales fundamentos técnicos teniendo en cuenta esta diferencia" (p.45).
>
> Contribuyo indicando que los modelos producidos en el MDD se inician de sistemas reales y concluyen en sistemas virtuales.

Modelo independiente de la computación.

Según, Pons (2102) al respecto sostiene lo siguiente:

> El modelo independiente de la computación (CIM) usualmente se le nombra esquema del dominio y en su elaboración se emplea una sintaxis que resulte comprensible para los expertos del dominio en cuestión. Se toma en cuenta que los usuarios a quienes está destinado el CIM, los expertos de dominio, no tienen fundamentación técnica acerca de los diagramas que se usarán para construir el software (p.46).
>
> Contribuyo indicando que el modelo CIM son representaciones del sistema real.

Modelo independiente de la plataforma.

Según, Pons (2102) al respecto sostiene lo siguiente:

> El modelo independiente de la plataforma (PIM) es un esquema con un alto grado de abstracción que es exento de cualquier tecnología o lenguaje de codificación. En el PIM el software se estructura desde la visión de cómo estudiar mejor al negocio, sin tomar en consideración cómo va a ser codificado, se ignora los sistemas operativos, los lenguajes de programación, el hardware, la topología de red, etc. (p.46).

Contribuyo indicando que el modelo PIM son representaciones del sistema real independientes de la tecnología informática.

Modelo especifico de la plataforma.

Según, Pons (2102) al respecto sostiene lo siguiente:

> El modelo orientado a la plataforma (PSM) es una estructuración de la proyección del PIM en una plataforma tecnológica específica. Un PIM puede obtener muchos PSMs, cada uno considera una plataforma tecnología particular. Frecuentemente, los PSMs deben interactuar entre sí para una solución correcta y consistente (p.46).

Contribuyo indicando que el modelo PSM son representaciones del sistema informático considerando cierta plataforma tecnológica.

Modelo de la implementación.

Según, Pons (2102) al respecto sostiene lo siguiente:

> "El modelo de la implementación (Código) es la conversión de cada PSM en código de computadora. Ya que el PSM está dirigido al contexto tecnológico específico, esta conversión es bastante directa" (p.46).

Contribuyo indicando que el modelo de la implementación son representaciones del sistema informático considerando su codificación.

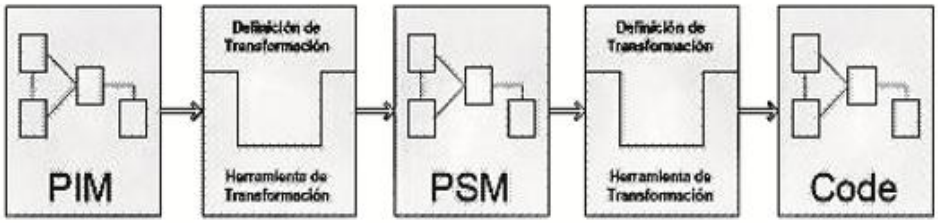


Figura 6: Los modelos empleados en el MDD
Nota: Adaptado del "Desarrollo de Software Dirigido por Modelos" por Pons, Giandini, Perez, 2010. Buenos Aires, Argentina:Edulp.

Proceso de transformación de modelos.

Según, Pons (2010) al respecto: sostiene lo siguiente:

> "Una conversión entre modelos puede visualizarse como una aplicación de computadora que toma un diagrama como ingreso y produce un diagrama como respuesta. Por lo tanto las conversiones podrían describirse utilizando cualquier plataforma de codificación, por ejemplo Java" (p.53)
>
> Contribuyo indicando que los modelos se transforman siguiendo las reglas de un lenguaje

Figura 7: Transformación entre lenguajes
Nota: Adaptado del "Desarrollo de Software Dirigido por Modelos" por Pons, Giandini, Perez, 2010. Buenos Aires, Argentina:Edulp

Ejemplo del proceso de transformación de modelos.

Según, Pons (2010) al respecto muestra el siguiente ejemplo:

> Conversión de un esquema PIM desarrollado en UML a un esquema de codificación en Java. Transformaremos el diagrama de clases del software de venta de libros en las clases implementadas en

Java correspondientes a ese esquema. Transformaremos el PIM en un PSM y luego transformaremos el PSM resultante a código Java (p.54).

Contribuyo con un ejemplo que ilustra el proceso de transformación de modelos en un lenguaje de modelado y programación.

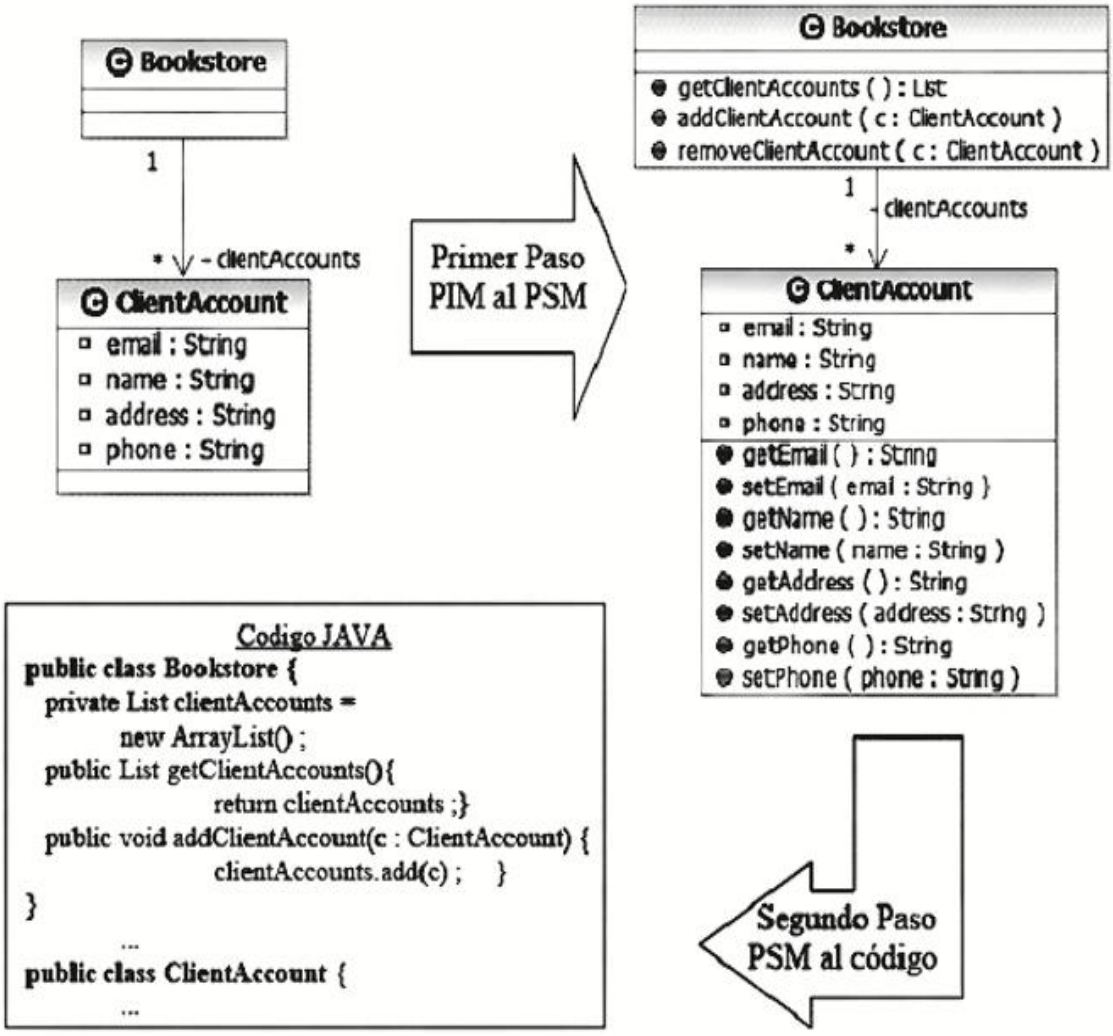


Figura 8: Ejemplo de conversión del PIM al PSM y al CODIGO
Nota: Adaptado del “Desarrollo de Software Dirigido por Modelos” por Pons, Giandini, Perez, 2010. Buenos Aires, Argentina:Edulp

Dimensiones de la transformación de modelos.

Según, Pons (2012) al respecto sostiene lo siguiente:

La conversión entre modelos forman el motor del MDD estos son: El empleo de un alto nivel de abstracción en el detalle tanto del

> problema a resolver como de la solución respectiva. El incremento de la garantía en la automatización guiada por computadora para soportar el análisis, diseño e implementación. El empleo de estándares industriales como mecanismo para garantizar el dialogo, la comunicación entre diferentes aplicaciones y productos, y la especialización tecnológica (p.29).

Contribuyo identificando tres dimensiones en la transformación de modelos: la abstracción, la automatización y los estándares.

Abstracción.

Según, Pons (2010) al respecto define lo siguiente:

> “La abstracción en el enfoque MDD definen los lenguajes de modelado básicos de dominio cuyos fundamentos muestran apropiadamente los conceptos del dominio del problema, mientras se ocultan o minimizan los elementos asociados con las tecnologías de programación” (p.30).

Contribuyo indicando que la abstracción es la muestra de los elementos del sistema real sin tener en cuenta los elementos tecnológicos.

Automatización.

Según, Pons (2010) al respecto define lo siguiente:

> La automatización es la técnica más eficaz para incrementar la productividad y la calidad. En MDD la esencia es utilizar a los equipos de cómputo para automatizar tareas repetitivas que se pueden automatizar, tareas que los seres humanos no elaboran con peculiar eficacia (p.30).

Contribuyo indicando que la automatización es el empleo de herramientas tecnológicas en remplazo al esfuerzo humano.

Estándares.

Según, Pons (2010) al respecto define lo siguiente:

> “Los estándares en MDD deben ser empleados por medio de normas que

originan beneficios, como por ejemplo capacidad para intercambiar detalles entre herramientas complementarias, o entre herramientas similares de diferentes proveedores" (p.30).

Contribuyo indicando que los estándares es el empleo de normas y principios en el uso del proceso de desarrollo de software.

Nivel de abstracción.

Según, De Miguel, Piattini y Marcos (2000) sostienen al respecto:

Es la causa y el efecto de traer, "identificar por medio de un mecanismo intelectual las características de un objeto para tratarlas aisladamente o para estudiar el mismo objeto en su pura esencia o noción". Por tanto la abstracción, como mecanismo mental es capaz de ocultar detalles particulares y fijarse en lo necesario" (p.4).

Contribuyo indicando que la abstracción es traer de una realidad los elementos necesarios para usarlos en la nueva realidad que se quiere modelar.

1.2.2 Bases teóricas de la variable: Calidad del modelo software

Teoría de la calidad.

Según, Zavala (2000) al respecto menciona lo siguiente:

La teoría de la calidad es lo más relevante de la condición del ser humano; en eso es lo que debe estar enfocado todo programa de calidad y así todo lo demás sería una natural y adecuada consecuencia y definitivamente no al revés (p.2).

Contribuyo indicando que la calidad es inherente a la persona humana, y que ayuda a que cada actividad muestre un resultado positivo.

Por otro lado, Deming (1986) al respecto plantea lo siguiente:

Los principios de calidad que sirven en cualquier parte, tanto en las pequeñas organizaciones como en las más grandes, en las empresas de servicios y en las dedicadas a la fabricación.

> Sirven para una división de una compañía estos son: 1) Crear constancia de propósito, 2) Adoptar la nueva filosofía, 3)Terminar con la dependencia de la inspección, 4) Terminar con la práctica de decidir negocios con base en los precios, 5) Mejorar el sistema de producción y de servicios, 6) Entrenamiento del trabajo, 7) Adoptar e instituir el liderazgo, 8) Eliminar temores, 9) Romper las barreras entre los departamentos, 10) Eliminar slogan, 11) Eliminar estándares, 12)Eliminar barreras que impidan alcanzar el orgullo al trabajador, 13) Instituir un activo programa de educación, 14) Implicar a todo el personal en la transformación (p.19).

Contribuyo indicando que los 14 principios permiten el empleo apropiado de la calidad en cualquier proceso de construcción de un producto.

Calidad del modelo software.

Según, Calero, Moraga y Piattini (2012) al respecto sostienen lo siguiente:

> Es un conjunto de atributos de calidad y sus asociaciones, acompañado de las técnicas para evaluar tales características (medidas, listas de control, convenciones de modelado, técnicas de inspección). La calidad de los modelos tiene una gran importancia, ya que determinará la calidad de los productos software finalmente implementado (p.206).

Contribuyo indicando que la calidad del modelo software es el uso de un conjunto de atributos medibles a los modelos que permiten obtener un producto de calidad.

Calidad del software.

Según, Sánchez, Sicilia y Rodríguez (2012) al respecto afirman:

> “Se denomina calidad del software al grado en el que un software posee una combinación de atributos deseables” (p.384).

Por otro lado Weitzenfeld (2005) complementa:

La calidad del software está fuertemente asociado con su proceso de construcción. Se toma en cuenta que un proceso bien conocido, utilizado y sustentado en métricas y predicción de resultados, permite gestionar en buena medida la calidad del desarrollo de software (p.56).

Finalmente Sánchez, Sicilia y Rodríguez (2012) sostienen:

> "El proceso de producción influye decisivamente en la calidad del producto resultante" (p.59).

Contribuyo indicando que la calidad en el proceso del desarrollo de software es fundamental su aplicación porque permitirá obtener un producto software de calidad.

Aseguramiento de la calidad

Según, Pressman (2010) al respecto sostiene lo siguiente:

Garantizar la calidad del software incluye un rango alto de preocupaciones y actividades que se centran en la administración de su calidad, estas son: Reglas, seguimientos, auditorias, pruebas, análisis de errores, gestión del cambio, capacitación del personal, gestión de proveedores, gestión de la seguridad, gestión de riesgos (p.370).

Contribuyo indicando que el aseguramiento de la calidad es la preocupación de todos los involucrados en el desarrollo y debe cumplir un conjunto de tareas para su consecución.

Procesos de aseguramiento de la calidad.

Según, Sánchez, Sicilia y Rodríguez (2012) al respecto sostienen lo siguiente:

Es la actividad de proporcionar las evidencias necesarias para garantizar que la función de calidad se lleva a cabo adecuadamente. Es un patrón sistemático y planificado de todas las acciones necesarias para afirmar con certeza que un producto es conforme con los requisitos técnicos establecidos (p.393).

Contribuyo indicando que los procesos del aseguramiento de la

calidad es aplicar correctamente los fundamentos y principios que guían el desarrollo de un producto software.

Dimensiones de la calidad del modelo software.

Según, Calero, Moraga y Piattini (2012) al respecto sostienen lo siguiente:

"Podemos decir que si bien no existe un modelo de calidad para modelos consensuado y valido, hay cierta tendencia a considerar tres tipos de calidad: calidad semántica, sintáctica y pragmática" (p.210)

Contribuyo identificando tres dimensiones para la calidad de un modelo: la calidad semántica, calidad sintáctica y calidad pragmática.

Calidad semántica.

Según, Calero, Moraga y Piattini (2012) al respecto sostienen lo siguiente:

Es el grado con el que el modelo representa correctamente el dominio del problema. Existen dos objetivos en esta dimensión: 1) Validez: significa que todo lo que refleja el modelo es correcto y relevante para el problema modelado, y 2) Compleción: significa que el modelo refleja todo lo que es relevante y correcto para el problema objeto del modelado (p.211).

Contribuyo indicando que la calidad semántica debe reflejar correctamente los elementos que forman el dominio del problema a estudiar.

Calidad sintáctica.

Segun, Calero, Moraga y Piattini (2012) al respecto sostienen lo siguiente:

"Es el grado con el que el modelo no contiene errores sintácticos. La corrección sintáctica de un modelo implica que todos los elementos que se incluyen en el modelo estén de acuerdo con la sintaxis del lenguaje de modelado" (p.211).

Contribuyo indicando que la calidad sintáctica debe reflejar correctamente la sintaxis del lenguaje de modelado que se esté empleando.

Calidad pragmática.

Según, Calero, Moraga y Piattini (2012) sal respecto sostienen lo siguiente:

> “Es la comprensión del modelo desde la perspectiva de los implicados en el proceso de modelado (modeladores, diseñadores, desarrolladores, etc.) La calidad pragmática captura el grado con el que el modelo es correctamente entendido por los implicados” (p.211).
>
> Contribuyo indicando que la calidad pragmática debe reflejar correctamente el grado de entendimiento del modelo por todo el equipo de desarrollo.

Indicadores de la calidad del modelo software.

Según, Calero, Moraga y Piattini (2012) al respecto sostienen lo siguiente:

> “Para medir un modelo es fundamental contar con técnicas para evaluar al menos cada característica de calidad puede medir la complejidad, la corrección y la entendibilidad de un modelo” (p.210).
>
> Contribuyo identificando tres indicadores para medir la calidad de un modelo: la complejidad, la corrección y la entendibilidad.

La complejidad.

Según, Calero, Moraga y Piattini (2012) al respecto sostienen lo siguiente:

> “El esfuerzo requerido para comprender un modelo y su importancia para la comunicación, comprensión y modificación” (p.207).
>
> Contribuyo indicando que la complejidad es la capacidad para comprender una realidad y reflejarla sus elementos en un modelo.

La corrección.

Segun, Calero, Moraga y Piattini (2012) al respecto sostienen lo siguiente:

> “La inclusión de elementos correctos y relaciones correctas entre ellos, e inclusión de afirmaciones correctas sobre el dominio. La no violación de normas y convenciones” (p.211).
>
> Contribuyo indicando que la corrección es la habilidad para representar de manera correcta en un modelo los elementos de una realidad.

La entendibilidad.

Según, Calero, Moraga y Piattini (2012) al respecto sostienen lo siguiente:

> Se define como el ser entendible por los usuarios; tanto usuarios humanos como herramientas. Para usuarios humanos, varios aspectos impactan en la comprensibilidad, como la estética de los diagramas, la organización de un modelo, la simplicidad o complejidad de un modelo, la utilización de conceptos familiares para los usuarios o seleccionados de la ontología del dominio (p.211).
>
> Contribuyo indicando que la entendibilidad de un modelo implica su comprensión, sus elementos y relaciones entre ellos y el uso de un lenguaje común para todos.

Proceso de software

Según, Sánchez, Sicilia y Rodríguez (2012) al respecto sostienen lo siguiente:

> "Es un grupo consistente de políticas, estructuras organizativas, tecnologías, procesos y artefactos que se requieren para conceptualizar, construir, utilizar y gestionar un producto software" (p.34).
>
> Contribuyo indicando que el proceso de software es el uso de un plan que respete principios y convenciones bien definidas para la construcción de un producto software.

Ciclo de desarrollo de software.

Según, Sánchez, Sicilia y Rodríguez (2012) al respecto sostienen lo siguiente:

> "Es el espacio de tiempo que se inicia cuando se toma la decisión de construir un producto de software y que termina cuando se entrega el software" (p.35).
>
> Contribuyo indicando que el ciclo de desarrollo de software es elaborar un plan respetando tiempos, esfuerzos y resultados.

Proyecto de ingeniería de software.

Según, Sánchez, Sicilia y Rodríguez (2012) al respecto sostienen lo siguiente:

> “Es un proyecto cuyo fin es generar un producto de software que satisfaga ciertas necesidades, en el tiempo previsto y dentro del costo establecido” (p.35).
>
> Contribuyo indicando que un proyecto de ingeniería de software es el uso de principios de ingeniería que guían el desarrollo del producto software.

Metodología de software.

Según, Pantaleo y Rinaudo (2015) al respecto sostienen lo siguiente:

> Una metodología es un marco de trabajo que puede ser utilizado como guía de las actividades a llevar a cabo. Es una forma de trabajo donde se especifica las tareas a llevar a cabo, los artefactos a generar y las relaciones entre ambos (p.54).
>
> Contribuyo indicando que una metodología de software es el empleo de un conjunto de buenas prácticas en la construcción de un producto software.

Propuesta del desarrollo de software conducido por modelos.

Según, Calero, Moraga y Piattini (2012) al respecto sostienen lo siguiente:

> Es mejorar la productividad y calidad del proceso de desarrollo, aplicando un enfoque centrado en modelos que utilizan modelos a distinto grado de abstracción y conversión entre dichos esquemas. El sistema es modelado a un alto grado de abstracción de forma independiente a la plataforma y a continuación es transformado a un modelo especifico de la plataforma y finalmente al código (p.206).
>
> Contribuyo indicando que el desarrollo dirigido por modelos concentra todo su esfuerzo en el modelado y el uso de herramientas sofisticadas que permitan optimizar el tiempo y generar calidad en el desarrollo.

Por otro lado, Cervantes, Velasco y Castro (2015) sostienen lo siguiente:

> El desarrollo de un sistema de software puede verse como una transformación hacia la solución técnica de determinada problemática u oportunidad con el fin de resolverla. Durante la transformación, que inicia en el conocimiento del problema y culmina en el desarrollo de la propuesta, se llevan a cabo distintas actividades técnicas: Requerimientos, diseño, construcción, pruebas e implantación (p.2).

Contribuyo indicando que el desarrollo de software es la creación de soluciones tecnológicas a partir de problemas reales.

Atributo.

Según, Sánchez, Sicilia y Rodríguez (2012) al respecto sostienen lo siguiente:

> "Un atributo es una propiedad medible de un objeto" (p.69).
>
> Contribuyo indicando que la cualidad del atributo de un objeto puede medirse.

Medición.

Según, Sánchez, Sicilia y Rodríguez (2012) al respecto sostienen lo siguiente:

> "Medición es el procedimiento por el que se asigna números o símbolos a propiedades de entidades de la realidad para describirlos de acuerdo a unas reglas definidas de antemano" (p.69).
>
> Contribuyo indicando que la medición es asignar un cierto valor a la medida de la cualidad del atributo de un objeto.

Medida.

Según, Sánchez, Sicilia y Rodríguez (2012) al respecto sostienen lo siguiente:

> "Medida es conceder un símbolo o número originado de una medición a una entidad para caracterizar sus cualidades" (p.69).
>
> Contribuyo indicando que una medida es cuantificar medida de la cualidad del atributo de un objeto.

Escala.

Según, Sánchez, Sicilia y Rodríguez (2012) al respecto sostienen lo siguiente:

> "Una escala de medición es un colección de valores que permite definir relaciones entre medidas. Con frecuencia dicha colección es continua, esta ordenado y viene encerrado por un punto inicial y otro final" (p.70).
>
> Contribuyo indicando que una escala es comparar las medidas de las cualidades de un conjunto de atributos de objetos diferentes.

Indicador.

Según, Pantaleo y Rinaudo (2015) al respecto sostienen lo siguiente:

> "Entendemos por indicador un número medido o calculado a partir de una medida directa que represente y ponga de manifiesto una característica o un comportamiento determinado del proceso observado" (p.245).
>
> Contribuyo indicando que un indicador es tener un patrón de medida establecido para las cualidades de los atributos de un objeto.

Métricas de complejidad de modelos.

Según, Calero, Moraga y Piattini (2012) al respecto proponen lo siguiente:

> "Medida de Genero et al. (2000) Plantea un conjunto de medidas para medir la complejidad estructural de los modelos de clases debido al uso de relaciones UML" (p.213).

Tabla 1:

Métricas para medir la complejidad estructural de diagrama de clases

Medida	**Definición**
Número de Clases (NC)	Número total de clases
Número de Atributos (NA)	Número total de atributos
Número de Métodos (NM)	Número total de métodos
Número de Asociaciones (NAssoc)	Número total de relaciones de asociación
Número de Agregaciones (NAgg)	Número total de relaciones de agregación
Número de Dependencias (NDep)	Número total de relaciones de dependencia

Número de Generalizaciones (NGen)	Número total de relaciones de generalización

Nota: Adaptado del “Calidad del producto y proceso software” por Calero, Angeles, Piattini, 2012. España:Ra-Ma Editorial

Métricas de corrección de modelos.

Según, Calero, Moraga y Piattini (2012) al respecto proponen lo siguiente:

> “Las lista de control son técnicas de lecturas orientadas a objetos (OORT) se usan en la inspección para detectar defectos en los artefactos software” (p.223).

Tabla 2:

Métricas para medir la corrección sintáctica del diagrama de clases

Medida	**Referencia**
Las clases deben tener nombres únicos	(Lange, 2007)
El nombre de cada clase debe estar en singular	(Unhelkar, 2005)
Todos los nombres de clases deben comenzar con mayúsculas	(Unhelkar, 2005)
Todos los atributos deben tener especificado su tipo	(Unhelkar, 2005)
Todos los atributos deben tener especificado la visibilidad	(Unhelkar, 2005)
Los nombres de los métodos deben ser verbos	(Ambler, 2005)
Los nombres de los métodos deben empezar en minúsculas	(Wust, 2005)
Las clases deben tener retornos	(Wust, 2005)
Toda clase debe asociarse con al menos una clase	(Berenbach, 2004)
Todas las asociaciones y agregaciones deben tener la multiplicidad especificada	(Berenbach, 2004)

Nota: Adaptado del “Calidad del producto y proceso software” por Calero, Angeles, Piattini, 2012. España:Ra-Ma Editorial

1.3 Justificación

1.3.1 Justificación teórica

Valor teórico, al aportar con nuevos conocimientos en el campo de la Teoría de Sistemas en particular en la Ingeniería del Software permitiendo plantear soluciones a problemas que afectan directa e indirectamente la construcción del producto software. Aportamos una nueva técnica llamada Transformación de Modelos que permite mejorar la calidad en la construcción de modelos de software que aplican los estudiantes y personas involucradas en la industria del software, del mismo modo planteamos sugerencias de nuevas técnicas de construcción de aplicaciones informáticas.

1.3.2 Justificación práctica

Implicancias prácticas, que se generan del logro que se obtienen en la presente investigación que permitirá mejorar las buenas prácticas en el desarrollo de software para estudiantes de ingeniería y personas involucradas; permitiendo paliar los problemas del aprendizaje en el campo del desarrollo de software. Sobre todo que los resultados de la investigación servirán de influencia para otros especialistas que estén interesados en indagar el tema tratado.

1.3.3 Justificación metodológica

Utilidad metodológica, porque tomamos en contemplación los formalismos de la investigación científica, dado que se tomara el test como artefacto de obtención de información, validándola y determinando su viabilidad, así mismo se usa la estadística descriptiva e inferencial para la evaluación de los datos y empleo de los métodos, tipos y diseño de investigación. Ello permitirá a otros investigadores e incluso a los estudiantes y docentes a poner en usanza el método científico para disipar las dificultades que se presentan en la organización.

1.3.4 Justificación epistemológica

Justificación epistemológica, al tener escasos estudios en nuestro país respecto a la transformación de modelos y su implicancia en la calidad del modelo software; los resultados que se consigan en la presente, servirá como ayuda a la rama de la

educación de la ingeniería de software y sentará base para futuras investigaciones, asimismo se permitirá aplicar a través del método científico nuevos paradigmas de desarrollo software en diferentes proyectos de ingeniería de sistemas.

1.4 Problema

1.4.1 Planteamiento del problema

Pons (2010) manifiesta lo siguiente:

> Históricamente, el proceso de construcción de software ha sido caro, complejo, incierto y muy lento para las condiciones de negocios actuales. Estas dificultades dieron nacimiento al concepto de “crisis del software” que en la práctica surgió en conjunto con la elaboración del software (p.23).

Pons (2010) manifiesta lo siguiente:

> Hoy en día, la magnitud de este problema sigue vigente, ya que se incrementan las necesidades de funcionalidades más sofisticadas y de software más eficientes. Por lo tanto, es necesario conocer donde se encuentran los orígenes de esta complejidad y lo que podemos hacer con ella (p.23).

Pantaleo y Rinaudo (2015) manifiestan lo siguiente:

> Los problemas presentes que afectan a los proyectos de elaboración de software son la carencia de predictibilidad en su realización dentro del tiempo y presupuesto planificado. Gran parte de estos casos son generados por errores en las estimaciones de esfuerzo de dichos proyectos o la utilización de algunas metodologías de desarrollo y gestión con criterios equivocados (p.7).

Bennet, McRobb y Farmer (2007) manifiestan lo siguiente:

> Una serie de informes sobre proyectos de sistemas de información desarrollados en EE.UU. estiman que los proyectos fallidos contabilizan un importe de 81,000 millones de dólares en 1994. Aunque esta situación ha mejorado sustancialmente en el año 2001, la tasa de fallo todavía

> seguía de un proyecto cada cuatro, mientras que un 49% de una muestra formada por casi 300,000 proyectos fueron descritos como “desafiantes”, en otras palabras “sufrieron retrasos y/o superaron el presupuesto inicial (p.27).

Bennet, McRobb y Farmer (2007) manifiestan lo siguiente:

> En cualquier equipo de desarrollo de sistemas de información que desee tener éxito, sus miembros deben gozar de una armoniosa mezcla de habilidades que resulte apropiado para las necesidades del proyecto. Estas pueden incluir el empleo de técnicas (tales como análisis orientado a objetos), conocimientos de metodologías (tales como el marco de trabajo unificado de software), experiencia en herramientas de codificación (tales como C# o Java) o un conocimiento detallado de las prestaciones del hardware (tales como los dispositivos de red). También debe existir un conjunto complementario de habilidades dentro del equipo para que un proyecto tenga éxito (p.34).

Pantaleo y Rinaudo (2015) manifiestan lo siguiente:

> Las revisiones de artefactos asociada a un proyecto de elaboración de software han mostrado ser el medio más eficiente para la detección y prevención de errores. Algunas estadísticas muestran que del total de errores de un proyecto, el 60% es resuelto con revisiones contra el 40% mediante test (p.9).

Serna (2013) manifiesta lo siguiente:

> América latina es una región que se ha distinguido por ser comprador de los desarrollos que provienen de los países industrializados, y eso le ha originado un consumismo que muchas veces supera lo racional. Pero, aunque no se construye tecnología, esta región puede lidiar con talento humano preparado. Por tanto, contar con ingenieros de software bien preparados, profesional idóneo, ético y involucrado, le brinda una oportunidad única frente a las otras regiones del mundo. Pero esto debe

> ser un plan de amplia envergadura, en el que se deben comprometer al país, la universidad y la industria, de tal forma que esos expertos sean apetecidos en todo el mundo por sus conocimientos y habilidades, y porque desarrollan productos software de alta calidad (p.72).

Serna (2013) manifiesta lo siguiente:

> La probabilidad de obtener un producto software de mucha calidad es una pregunta que depende de diversos factores, pero uno de los que mucha influencia tiene es la habilidad de los profesionales encargados de su construcción. En tal sentido, su preparación técnico-académica debe ser lo suficientemente apropiada como para poder realizar los proyectos, y enfrentar a las exigencias, dificultades y necesidades que plantean los problemas sociales (p.78).

Serna (2013) manifiesta lo siguiente:

> Estos magros índices de calidad descritos están asociados con dos tipos de cuestiones, por un lado con dificultades asociados al proceso de construcción de software y a su vez, con dificultades en la formación de las habilidades y competencias académicas de los profesionales integrantes de los equipos de construcción de software (p.80).

Pons (2010) manifiesta lo siguiente:

> La industria del software tiene algo en particular que la diferencia de las otras industrias. Cada año surgen nuevas tecnologías que fácilmente llegan a ser populares y muchas empresas requieren aplicar estas nuevas tecnologías. Las nuevas tecnologías ofrecen beneficios para las compañías y muchas de ellas no quieren quedarse atrás. Por lo tanto, tienen que pasar a utilizarlas dichas tecnologías cuanto antes. Como consecuencia del cambio, las inversiones en tecnologías anteriores pierden valor. Por ende, el software existente debe pasar a una nueva tecnología, o a una versión nueva y diferente de la usada en su elaboración (p.28).

Estos bajos índices en el desarrollo de software están asociados con dos problemas: dificultades relacionadas al campo del desarrollo de software por la complejidad que este tiene y, dificultades relacionadas al campo de la formación académico de los equipos de desarrollo por los bajos niveles de competencias aprendidas en los centros de enseñanza esto nos lleva a mejorar la calidad del desarrollador así como la calidad del software en especial la calidad del modelo.

Sobre la base de esta problemática, el presente estudio propone emplear el paradigma del desarrollo de software guiado por modelos y sus transformaciones de modelos en el campo de la construcción de productos software que permitirá que se logre mejorar la calidad del software en especial sus modelos de esta manera se podrá mejorar las habilidades de los equipos de desarrollo.

1.4.2 Problema general

¿En qué medida aplicar transformación de modelos mejora la calidad del modelo software en estudiantes de Ingeniería Universidad Cesar Vallejo Lima 2016?

1.4.3 Problemas específicos

Problema específico 1

¿En qué medida aplicar transformación de modelos mejora la calidad semántica del modelo software en estudiantes de Ingeniería Universidad César Vallejo Lima 2016?

Problema específico 2

¿En qué medida aplicar transformación de modelos mejora la calidad sintáctica del modelo software en estudiantes de Ingeniería Universidad César Vallejo Lima 2016?

Problema específico 3

¿En qué medida aplicar transformación de modelos mejora la calidad pragmática del modelo software en estudiantes de Ingeniería Universidad César Vallejo Lima 2016?

1.5 Hipótesis

1.5.1 Hipótesis general

La aplicación de transformación de modelos mejora la calidad del modelo software

en estudiantes de Ingeniería Universidad César Vallejo Lima 2016.

1.5.2 Hipótesis específicas

Hipótesis especifica 1

La aplicación de transformación de modelos mejora la calidad semántica del modelo software en estudiantes de Ingeniería Universidad César Vallejo Lima 2016.

Hipótesis especifica 2

La aplicación de transformación de modelos mejora la calidad sintáctica del modelo software en estudiantes de Ingeniería Universidad César Vallejo Lima 2016.

Hipótesis especifica 3

La aplicación de transformación de modelos mejora la calidad pragmática del modelo software en estudiantes de Ingeniería Universidad César Vallejo Lima 2016.

1.6 Objetivos

1.6.1 Objetivo general

Demostrar que aplicar transformación de modelos mejora la calidad del modelo software en estudiantes de Ingeniería Universidad César Vallejo Lima 2016.

1.6.2 Objetivos específicos

Objetivo específico 1

Demostrar que aplicar transformación de modelos mejora la calidad semántica del modelo software en estudiantes de Ingeniería Universidad César Vallejo Lima 2016.

Objetivo específico 1

Demostrar que aplicar transformación de modelos mejora la calidad sintáctica del modelo software en estudiantes de Ingeniería Universidad César Vallejo Lima 2016.

Objetivo específico 1

Demostrar que aplicar transformación de modelos mejora la calidad pragmática del modelo software en estudiantes de Ingeniería Universidad César Vallejo Lima 2016.

II. Marco metodológico

2.1 Variables

2.1.1 Transformación de modelos

Definición conceptual

Pons (2010) define la transformación de modelos:

> Es la obtención automática de un modelo objetivo desde un modelo básico, respetando a una definición de transformación. Esta definición es un grupo de reglas de transformación que juntas explican como un modelo en el lenguaje básico puede ser transformado en un modelo en el lenguaje objetivo. Una regla de transformación es una explicación de cómo una o más construcciones en el lenguaje básico puede ser transformadas en una o más construcciones en el lenguaje objetivo (p.53).

Definición operacional

Aplicación de la transformación de modelos.

1. Formación de los alumnos en los fundamentos de modelos conceptuales
2. Formación de los alumnos en los fundamentos de modelos lógicos
3. Entrenamiento en habilidades para la construcción de modelos conceptuales
4. Formación de los alumnos en los fundamentos de reglas de transformación de modelos
5. Entrenamiento en la aplicación de las reglas de transformación para la construcción de modelos lógicos
6. Puesta en práctica de la técnica de transformación de modelos

2.1.2 Calidad del modelo software

Definición conceptual

Calero, Moraga y Piattini (2012) definen la calidad del modelo software:

> El un grupo de características de calidad y sus relaciones, acompañado de las técnicas para evaluar tales características (medidas, listas de control, convenciones de modelado, técnicas de inspección). La calidad de los modelos tiene una gran importancia, ya que determinará la calidad de los productos software finalmente implementado (p.206).

Definición operacional

Test tipo escala de Guttman con 20 ítems en tres dimensiones: (a) Calidad semántica, (b) Calidad sintáctica y (c) Calidad pragmática.

2.2 Operacionalización de variables

Tabla 3:

Operacionalización de la variable calidad del modelo software

Dimensiones	Indicadores	Ítems	Escalas	Niveles
Calidad semántica	-Complejidad	1-7	0:Falso 1:Verdadero	0-6 Bajo 7-13 Regular 14-20 Alto
Calidad sintáctica	-Corrección	8-17		
Calidad pragmática	-Entendibilidad	18-20		

2.3 Metodología

Empleamos el método “Hipotético-Deductivo” porque obtendremos resultados a partir de supuestos que pueden ser demostradas a través de experimentos

2.4 Tipo de estudio

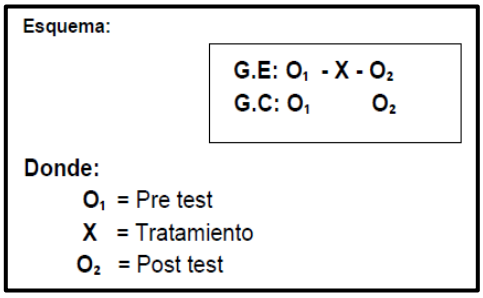
Aplicamos el tipo de estudio “Experimental – Longitudinal” porque experimentaremos el comportamiento de una variable con dos observaciones, antes y después del tratamiento.

Para definir el tipo experimental, Supo (2012) sostiene al respecto: “Son prospectivos, longitudinales, analíticos y de nivel investigativo; además de ser controlados” (p.1).

Para definir el tipo longitudinal, Supo (2012) sostiene al respecto: “La variable de estudio es evaluada en dos o más observaciones; por ello, de realizar contrastaciones (antes – después) son entre muestras vinculadas” (p.1).

2.5 Diseño de investigación

Aplicamos el diseño “Cuasi experimental”, porque estudiamos dos grupos, control y experimental, ejecutando dos observaciones en cada uno de los grupos.



Para definir el diseño cuasi-experimental, Supo (2012) sostiene al respecto: “Sí no hay grupo control, no se debe realizar la asignación aleatoria, se realiza dos evaluaciones en el mismo grupo” (p.3).

2.6 Población, muestra y muestreo

2.6.1 Población

La población de estudio está constituida por 67 unidades de estudio constituidos por estudiantes de V ciclo de la Escuela Profesional de Ingeniería de Sistemas de la Universidad César Vallejo, Lima Norte, periodo académico 2016-II, asignatura Ingeniería de Software.

Tabla 4:

Distribución de la población

Sección	Turno	Población
A	Mañana	30
B	Tarde	37
Total		67

Nota: Escuela de Ingeniería de Sistemas – Universidad César Vallejo

Para definir la población, Supo (2012) sostiene al respecto: “Es la reunión de todas las sujetos de estudio, cuya característica observable o reacción que pueden expresar nos incumbe analizar” (p.16).

2.6.2 Muestra

La muestra es no probabilística porque permite seleccionar por conveniencia los grupos de estudio, calculamos el tamaño de muestra usando la fórmula de proporción con marco muestral:

$$n = \frac{N * Z_{1-\alpha/2}{}^{2} * p * q}{d^{2} * (N-1) + Z_{1-\alpha/2}{}^{2} * p * q}$$

Z=1.960 (para el nivel de confianza del 95%)

e=0.05 (error de estimación)

N=67 (tamaño de población)

p=0.500

q=0.500

d=0.050

Remplazando valores:

$$\mathrm{n} = \frac{67(1.960)^{2}(0.500)(0.500)}{(0.050)^{2}(100-1) + (1.960)^{2(}0.500)(0.500)} = 57$$

El tamaño de la muestra calculada fue de 57 estudiantes.

Para definir el tamaño de la muestra, Supo (2012) sostiene al respecto: “El tamaño de la muestra se relaciona con el tamaño del marco muestral y la posibilidad de ser incluido en la muestra debe calcularse” (p.16).

2.6.3 Muestreo

Aplicamos el procedimiento de muestreo no probabilístico, porque se escogieron dos grupos por conveniencia ya que debían contar con conocimientos parejos del tema a tratar en la experimentación.

Para definir el muestreo no probabilístico, Supo (2012) sostiene al respecto: “Muestreo por conveniencia, de juicio o criterio, muestreo por cuotas o accidental”

(p.16).

Aplicamos el muestreo aleatorio estratificado, para obtener los estratos que conformaran los grupos de estudio. Los estratos obtenidos son: “Control” con 26 cuotas y “Experimental” con 31 cuotas.

Tabla 5

Distribución por estratos de la muestra

ESTRATO	Ni	Ni*F	CUOTA
Grupo Control	30	30*0.851	26
Grupo Experimental	37	37*0.851	31
TOTAL	67		57

Nota: Elaboración propia

Para definir el muestreo aleatorio estratificado, Supo (2012) sostiene al respecto: “Es generar grupos heterogéneos entre sí de acuerdo a la variable de estudio pero homogéneos dentro de cada grupo, así garantizamos la representación de cada estrato en la muestra” (p.18).

2.7 Técnicas e instrumentos de recolección de datos

2.7.1 Técnica

Usamos el procedimiento de recopilación de datos “Encuesta” conformada por un formulario basados en un caso de estudio y preguntas para obtener el comportamiento de la variable.

Para definir la técnica encuesta, Supo (2012) sostiene al respecto:

> La encuesta pretende conocer la reacción o la respuesta de un conjunto de individuos que pueden corresponder a una muestra o a una población, por lo tanto se puede contar, requiere de un instrumento que provoque las reacciones en el encuestado (p.19).

2.7.2 Instrumento

Aplicamos el instrumento de tipo TEST con un caso de estudio y proponemos el modelo origen para que se obtenga el modelo destino y preguntas cerradas, para medir el efecto que causa el tratamiento aplicado.

Para definir un TEST, Supo (2012) sostiene al respecto: “Este concepto hace referencia a las pruebas destinadas a evaluar conocimientos, aptitudes o funciones” (p.16).

Ficha técnica del instrumento: Calidad del modelo software

Nombre del instrumento	Test de calidad del modelo software
Autor	Br. Chunga Huatay Edwin -
Año	2016
Universo de estudio	Estudiantes de ingeniería
Nivel de confianza	95.0%
Margen de error	5.0%
Tamaño muestral	57 estudiantes
Tipo de técnica	Encuesta
Tipo de instrumento	Test
Fecha trabajo de campo	Periodo Académico 2016-II
Escala de medición	Guttman
Tiempo utilizado	20 minutos

Validez

La validación se obtuvo mediante el procedimiento juicio de expertos.

Confiabilidad

La confiabilidad se dio a través de los resultados de una prueba piloto aplicada a diez (10) estudiantes seleccionados adecuadamente.

Tabla 6:

Resultado de la confiabilidad del instrumento

Calidad del modelo software	
Alfa de Cronbach	Nº de elementos
0.809	10

Interpretación.

En la tabla 6 se observa el resultado de confiabilidad, de acuerdo al Alfa de Cronbach es igual a 0.809; por lo que el instrumento se considera “fuertemente confiable”.

2.8 Métodos de análisis de datos

2.8.2 Matriz de datos

Construcción de la tabla de datos para almacenar evidencias obtenidas de las observaciones.

2.8.3 Estadística descriptiva

Elaboramos tablas y gráficos estadísticas para el análisis de resultados de la información obtenida aplicando la distribución de frecuencias.

Para Nolberto (2008), la Estadística Descriptiva es: “Tata sobre la descripción y análisis estadístico de una población, que compendia y presenta datos obtenidos de la población o de una muestra, mediante técnicas adecuadas” (p 14).

2.8.4 Estadística inferencial

Evaluamos dos grupos obtenidos de una muestra aleatoria de 57 alumnos.

Par Supo (2012), comparar grupos es:” el análisis bivariado más elemental con una variable fija y una variable aleatoria. Su objetivo es identificar las diferencias entre los grupos participantes; se puede contrastar dos o más grupos” (p 15).

Aplicamos pruebas no paramétricas a partir del siguiente análisis la investigación es de tipo experimental, nivel investigativo relacional, diseño cuasi-experimental, objetivo estadístico comparar grupos en muestras relacionadas, escala de medición

de la variable de estudio ordinal, el comportamiento de los datos no tiene distribución normal ni varianzas homogéneas en conclusión para la prueba de hipótesis aplicamos la prueba de Wilcoxon.

Para Supo (2012), la elección del estadístico de prueba es:

> El estadístico de prueba se selecciona en función a 6 conceptos: tipo de estudio, nivel investigativo, diseño de la investigación, objetivo estadístico, escalas de medición de las variables y comportamiento de los datos, es este último punto donde debemos considerar la distribución de los datos en las variables de numéricas y las frecuencias esperadas para las variables categóricas (p-24).

2.8.5 Software aplicativo

Software aplicativo MS WORD, MS EXCEL y de estadística SPSS V 22.

2.9 Aspectos éticos

Se consideran los siguientes principios éticos: la confidencialidad de la información obtenida no será revelada ni divulgada para cualquier otro fin, el consentimiento informado se solicitara autorización a la Universidad César Vallejo para la realización del estudio y lograr su participación de manera voluntaria y la libre participación de los estudiantes es sin presión alguna, pero si motivándolos sobre la importancia de la investigación.

III. Resultados

3.1 Resultados descriptivos

3.1.1 Calidad del modelo software

Tabla 7

Distribución de frecuencias de la variable: Calidad del modelo software antes del tratamiento

		ANTES			
		Frecuencia	**Porcentaje**	**Porcentaje válido**	**Porcentaje acumulado**
Válidos	BAJO	5	8,8	8,8	8,8
	REGULAR	49	86,0	86,0	94,7
	ALTO	3	5,3	5,3	100,0
	Total	57	100,0	100,0	

Nota: La fuente se obtuvo de los test

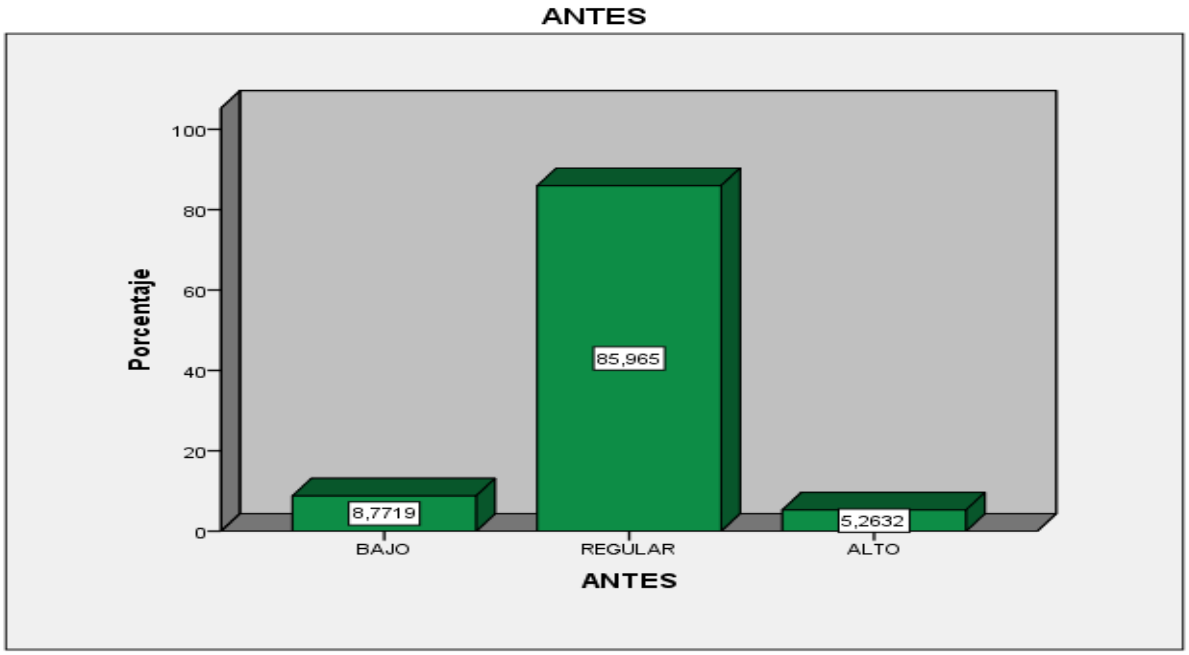


Figura 9: Nivel de la variable: Calidad del modelo software antes del tratamiento

Interpretación:

La calidad de modelo software antes del experimento según la tabla 7 y figura 9 se, observa que de 57 encuestados 3 que equivale el 5.8% manifiestan un nivel alto, 49 que equivale al 86.0% manifiestan un nivel regular y 5 que equivale al 8.8% manifiesta un nivel bajo de conocimiento de la calidad del modelo software.

Tabla 8

Distribución de frecuencias de la variable: Calidad del modelo software después del tratamiento

		DESPUÉS			
		Frecuencia	**Porcentaje**	**Porcentaje válido**	**Porcentaje acumulado**
Válidos	BAJO	2	3,5	3,5	3,5
	REGULAR	44	77,2	77,2	80,7
	ALTO	11	19,3	19,3	100,0
	Total	57	100,0	100,0	

Nota: La fuente se obtuvo de los test

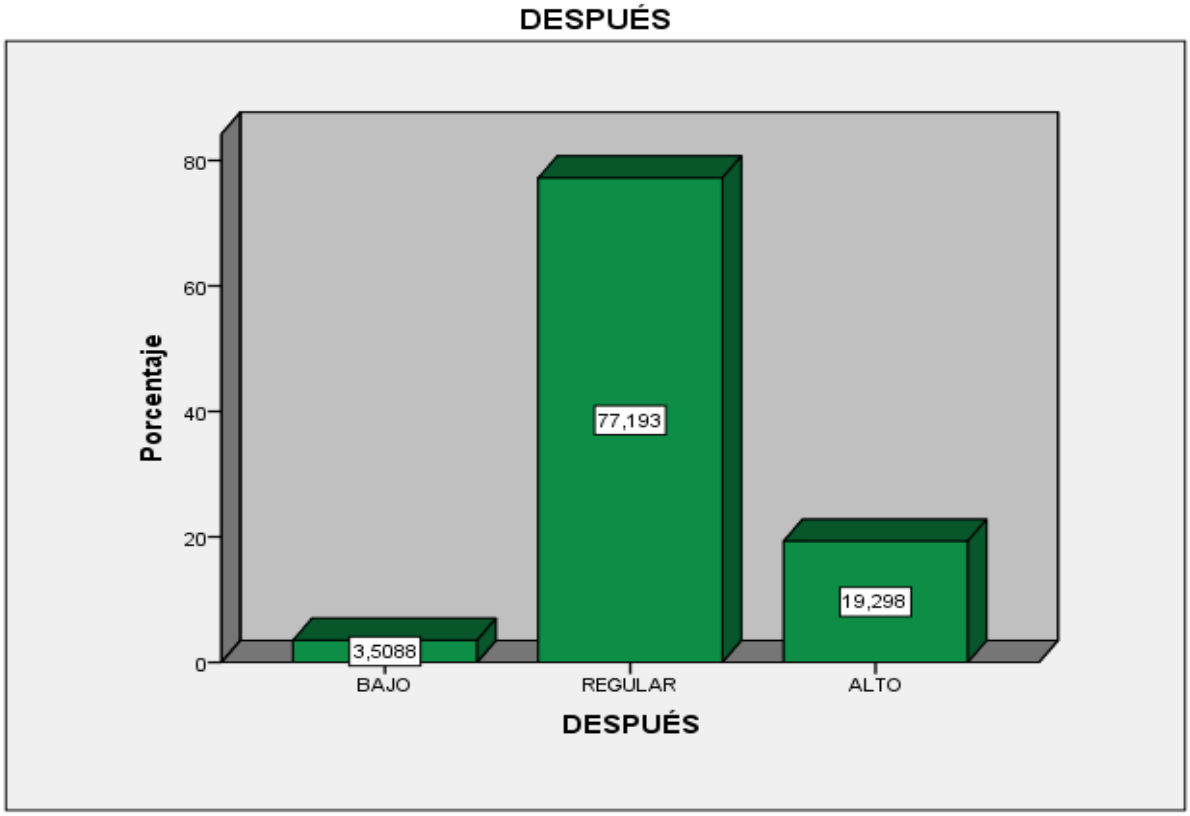


Figura 10: Nivel de la variable: Calidad del modelo software después del tratamiento

Interpretación:

La calidad de modelo software después del experimento según tabla 8 y figura 10 se, observa que de 57 encuestados 11 que equivale el 19.3% manifiestan un nivel alto, 44 que equivale al 77.2% manifiestan un nivel regular y 2 que equivale al 3.5% manifiesta un nivel bajo de conocimiento de la calidad en el modelo software.

3.1.2 Calidad semántica

Tabla 9

Distribución de frecuencias de la dimensión: Calidad semántica del modelo software antes del tratamiento

Calidad Semántica Antes					
		Frecuencia	Porcentaje	Porcentaje válido	Porcentaje acumulado
Válidos	Bajo	10	17,5	17,5	17,5
	Regular	42	73,7	73,7	91,2
	Alto	5	8,8	8,8	100,0
	Total	57	100,0	100,0	

Nota: La fuente se obtuvo de los test

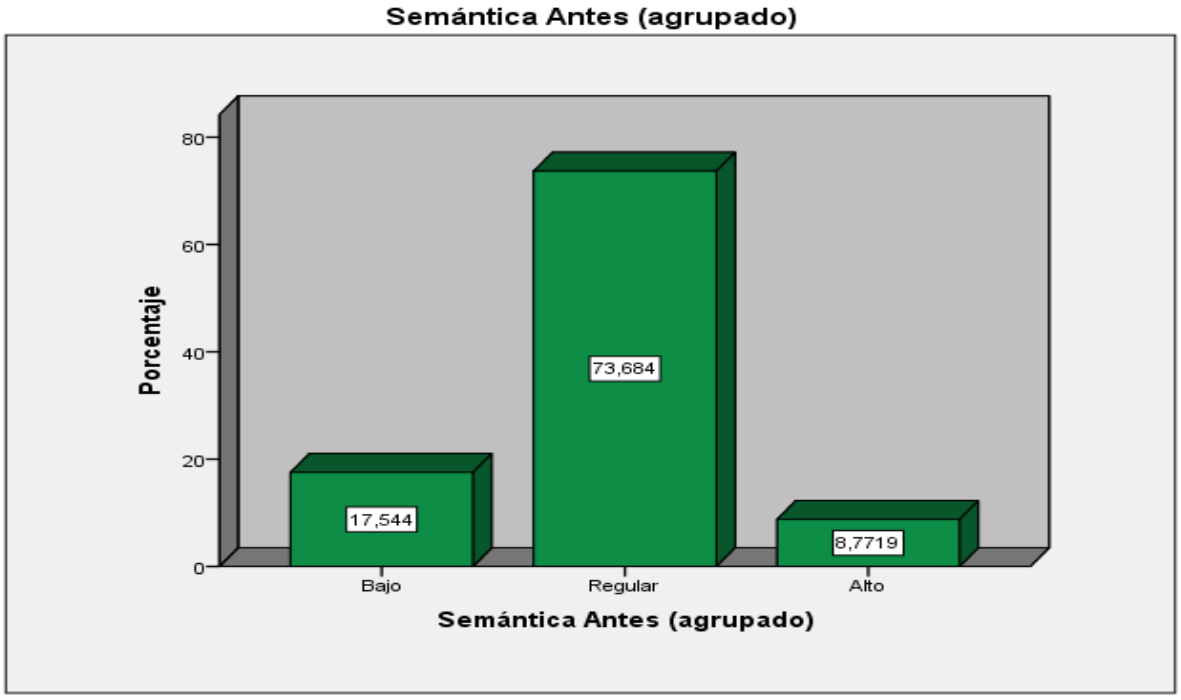


Figura 11: Nivel de la dimensión: Calidad semántica del modelo software antes

Interpretación:

La calidad semántica de modelo software antes del experimento según la tabla 9 y figura 11 se, observa que de 57 encuestados 5 que equivale el 8.8% manifiestan un nivel alto, 42 que equivale al 73.7% manifiestan un nivel regular y 10 que equivale al 17.5% manifiesta un nivel bajo de conocimiento de la calidad semántica del modelo software.

Tabla 10

Distribución de frecuencias de la dimensión: Calidad semántica del modelo software después del tratamiento

		Calidad Semántica Después			
		Frecuencia	Porcentaje	Porcentaje válido	Porcentaje acumulado
Válidos	Bajo	5	8,8	8,8	8,8
	Regular	38	66,7	66,7	75,4
	Alto	14	24,6	24,6	100,0
	Total	57	100,0	100,0	

Nota: La fuente se obtuvo de los test

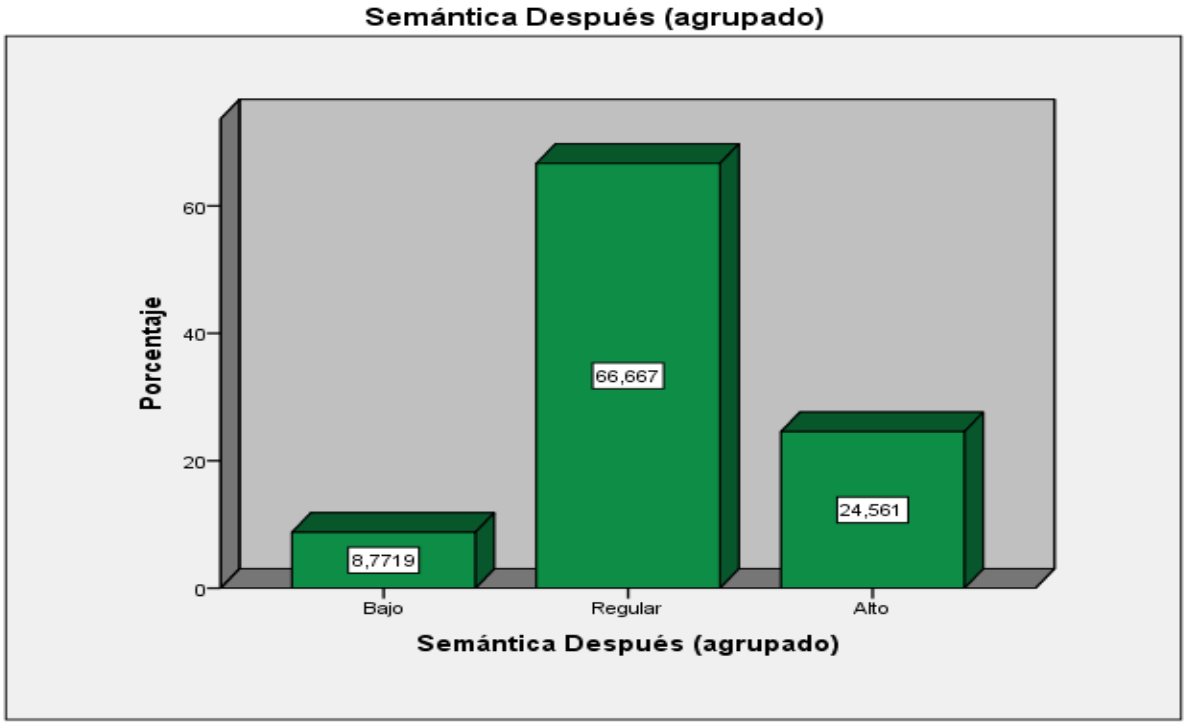


Figura 12: Nivel de la dimensión: Calidad semántica del modelo software después

Interpretación:

La calidad semántica de modelo software después del experimento según la tabla 10 y figura 12 se, observa que de 57 encuestados 14 que equivale el 24.6% manifiestan un nivel alto, 38 que equivale al 66.7% manifiestan un nivel regular y 5 que equivale al 8.8% manifiesta un nivel bajo de conocimiento de la calidad semántica del modelo software.

3.1.3 Calidad sintáctica

Tabla 11

Distribución de frecuencias de la dimensión: Calidad sintáctica del modelo software antes del tratamiento

Calidad Sintáctica Antes					
		Frecuencia	Porcentaje	Porcentaje válido	Porcentaje acumulado
Válidos	Bajo	3	5,3	5,3	5,3
	Regular	46	80,7	80,7	86,0
	Alto	8	14,0	14,0	100,0
	Total	57	100,0	100,0	

Nota: La fuente se obtuvo de los test

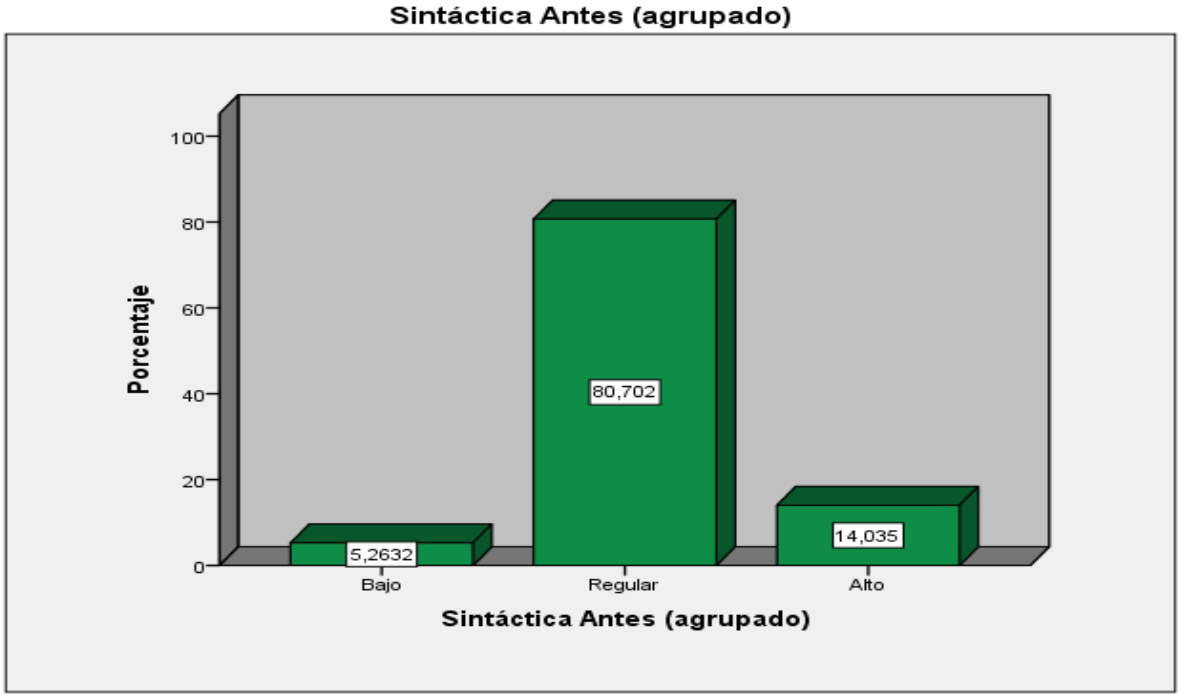


Figura 13: Nivel de la dimensión: Calidad sintáctica del modelo software antes.

Interpretación:

La calidad sintáctica de modelo software antes del experimento según la tabla 11 y figura 13 se, observa que de 57 encuestados 8 que equivale el 14.0% manifiestan un nivel alto, 46 que equivale al 80.7% manifiestan un nivel regular y 3 que equivale al 5.3% manifiesta un nivel bajo de conocimiento de la calidad sintáctica del modelo software.

Tabla 12

Distribución de frecuencias de la dimensión: Calidad sintáctica del modelo software después del tratamiento

Calidad Sintáctica Después

		Frecuencia	Porcentaje	Porcentaje válido	Porcentaje acumulado
Válidos	Bajo	3	5,3	5,3	5,3
	Regular	43	75,4	75,4	80,7
	Alto	11	19,3	19,3	100,0
	Total	57	100,0	100,0	

Nota: La fuente se obtuvo de los test

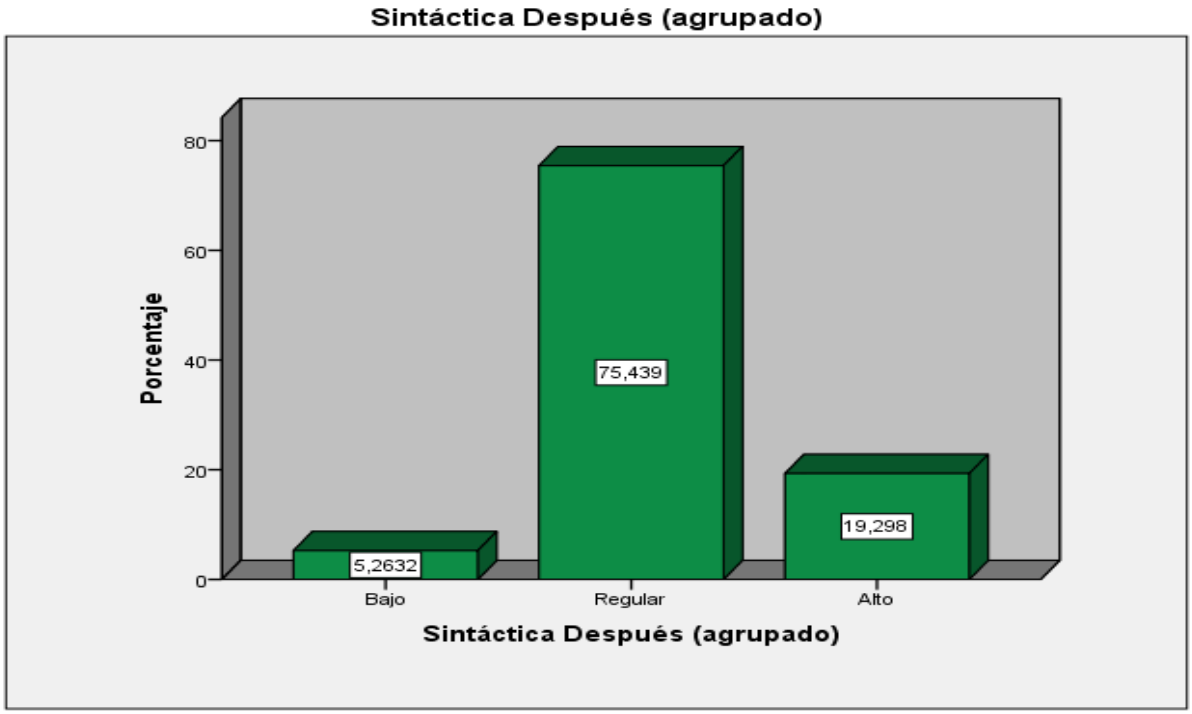


Figura 14: Nivel de la dimensión: Calidad sintáctica del modelo software después

Interpretación:

La calidad sintáctica de modelo software después del experimento según la tabla 12 y figura 14 se, observa que de 57 encuestados 11 que equivale el 19.3% manifiestan un nivel alto, 43 que equivale al 75.4% manifiestan un nivel regular y 3 que equivale al 5.3% manifiesta un nivel bajo de conocimiento de la calidad sintáctica del modelo software.

3.1.4 Calidad pragmática

Tabla 13

Distribución de frecuencias de la dimensión: Calidad pragmática del modelo software antes del tratamiento

Pragmática Antes (agrupado)					
		Frecuencia	Porcentaje	Porcentaje válido	Porcentaje acumulado
Válidos	Regular	10	17,5	17,5	17,5
	Alto	47	82,5	82,5	100,0
	Total	57	100,0	100,0	

Nota: La fuente se obtuvo de los test

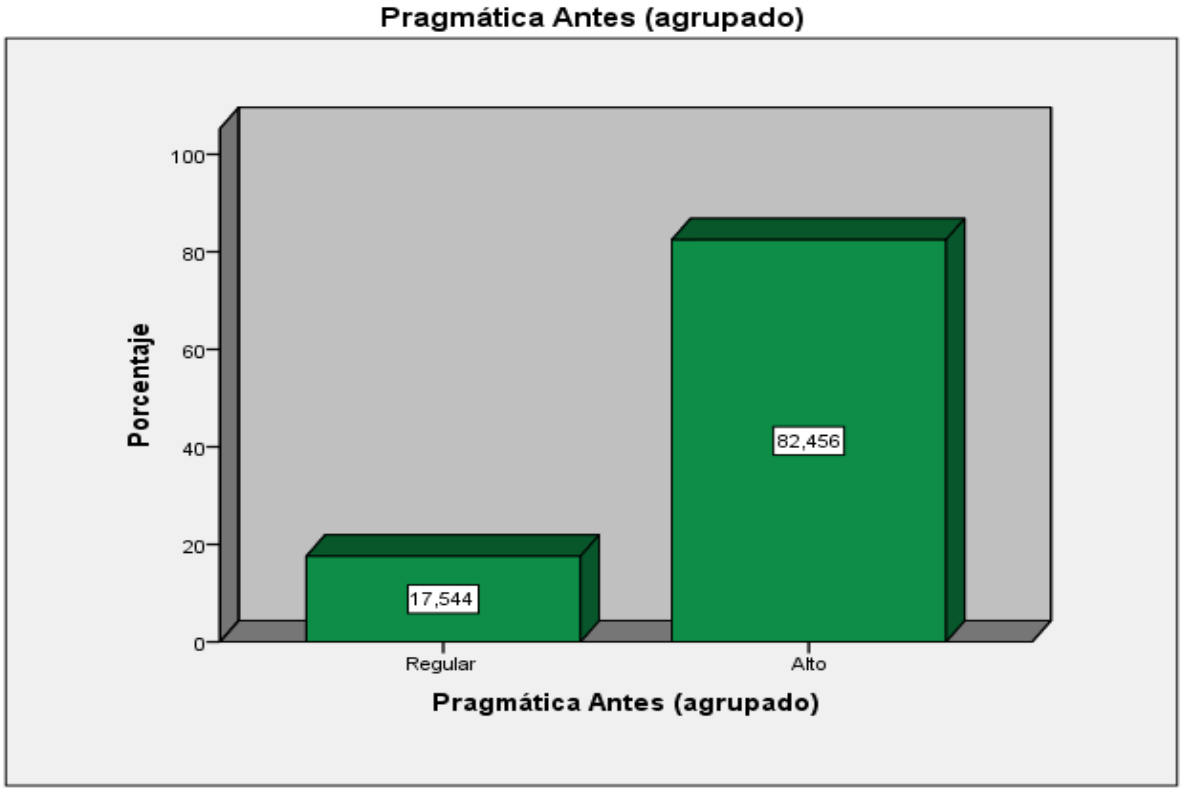


Figura 15: Nivel de la dimensión: Calidad pragmática del modelo software antes

Interpretación:

La calidad pragmática de modelo software antes del experimento según la tabla 13 y figura 15 se, observa que de 57 encuestados 47 que equivale el 82.5% manifiestan un nivel alto, 10 que equivale al 17.5% manifiestan un nivel regular de conocimiento de la calidad pragmática del modelo software.

Tabla 14

Distribución de frecuencias de la dimensión: Calidad pragmática del modelo software después del tratamiento

Calidad Pragmática Después

		Frecuencia	Porcentaje	Porcentaje válido	Porcentaje acumulado
Válidos	Regular	3	5,3	5,3	5,3
	Alto	54	94,7	94,7	100,0
	Total	57	100,0	100,0	

Nota: La fuente se obtuvo de los test

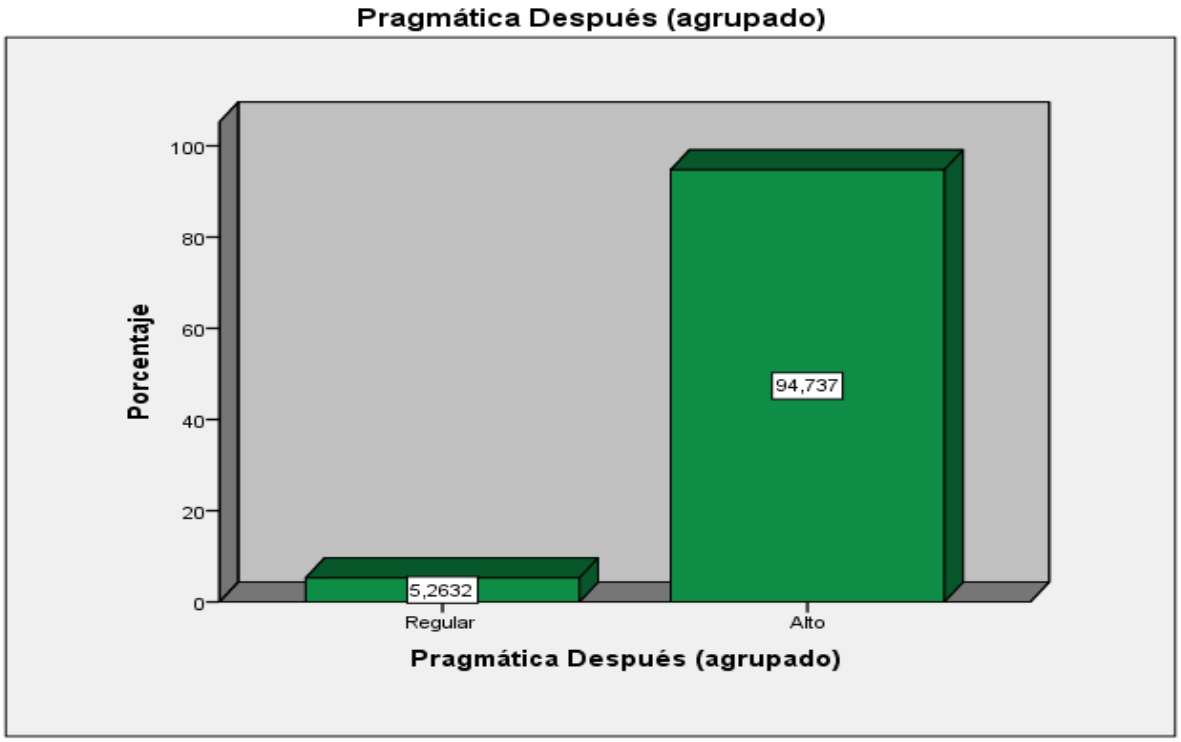


Figura 16: Nivel de la dimensión: Calidad pragmática del modelo software después

Interpretación:

La calidad pragmática de modelo software antes del experimento según la tabla 14 y figura 16 se, observa que de 57 encuestados 54 que equivale el 94.7% manifiestan un nivel alto, 3 que equivale al 5.3% manifiestan un nivel regular de conocimiento de la calidad pragmática del modelo software.

3.2 Resultados inferenciales

3.2.1 La aplicación de transformación de modelos mejora la calidad del modelo software

$H_{0:}$ No mejora la calidad del modelo software después de aplicar la transformación de modelos en el grupo control

$H_{a:}$ Mejora la calidad del modelo software después de aplicar la transformación de modelos en el grupo control

Prueba de Hipótesis general

Nivel de confianza: 95% (α = 0.05)

Regla de decisión:

- Si p < α; se rechaza la hipótesis nula.
- Si p > α; se acepta la hipótesis nula.

Prueba Estadística: Prueba de Wilcoxon.

Tabla 15

Prueba de Wilcoxon de la calidad del modelo software del grupo control

		N	Rango promedio	Suma de rangos
DESPUÉS - ANTES	Rangos negativos	7[a]	14,71	103,00
	Rangos positivos	17[b]	11,59	197,00
	Empates	2[c]		
	Total	26		

a. DESPUÉS < ANTES b. DESPUÉS > ANTES c. DESPUÉS = ANTES

Estadísticos de contraste[b]	
	DESPUÉS - ANTES
Z	-1,353[a]
Sig. asintót. (bilateral)	,176

a. Basado en los rangos negativos.

b. Prueba de los rangos con signo de Wilcoxon

Interpretación:

La significancia estadística obtenida es p (0.176) muestra que p es mayor a 0.05, entonces se acepta la hipótesis nula, por lo tanto, se rechaza la hipótesis alterna y se concluye que no mejora la calidad del modelo software después de aplicar la transformación de modelos en el grupo control.

$H_{0:}$ No mejora la calidad del modelo software después de aplicar la transformación de modelos en el grupo experimental

$H_{a:}$ Mejora la calidad del modelo software después de aplicar la transformación de modelos en el grupo experimental

Tabla 16

Prueba de Wilcoxon de la calidad del modelo software del grupo experimental

		N	Rango promedio	Suma de rangos
DESPUÉS - ANTES	Rangos negativos	8[a]	9,19	73,50
	Rangos positivos	18[b]	15,42	277,50
	Empates	5[c]		
	Total	31		

a. DESPUÉS < ANTES b. DESPUÉS > ANTES c. DESPUÉS = ANTES

Estadísticos de contraste[b]	
	DESPUÉS - ANTES
Z	-2,606[a]
Sig. asintót. (bilateral)	,009

a. Basado en los rangos negativos.

b. Prueba de los rangos con signo de Wilcoxon

Interpretación:

La significancia estadística obtenida es p (0.009) muestra que p es menor a 0.05, entonces se rechaza la hipótesis nula, por lo tanto, se acepta la hipótesis alterna y se concluye que mejora la calidad del modelo software después de aplicar la transformación de modelos en el grupo experimental.

3.2.2 La aplicación de transformación de modelos mejora la calidad semántica del modelo software

$H_{0:}$ No mejora la calidad semántica del modelo software después de aplicar la transformación de modelos en el grupo experimental

$H_{a:}$ Mejora la calidad semántica del modelo software después de aplicar la transformación de modelos en el grupo experimental

Tabla 17

Prueba de Wilcoxon de la calidad semántica del modelo software del grupo experimental

		N	Rango promedio	Suma de rangos
DESPUÉS - ANTES	Rangos negativos	8[a]	11,13	89,00
	Rangos positivos	18[b]	14,56	262,00
	Empates	5[c]		
	Total	31		

a. DESPUÉS < ANTES b. DESPUÉS > ANTES c. DESPUÉS = ANTES

Estadísticos de contraste[b]	DESPUÉS - ANTES
Z	-2,262[a]
Sig. asintót. (bilateral)	,024

a. Basado en los rangos negativos.

b. Prueba de los rangos con signo de Wilcoxon

Interpretación:

La significancia estadística obtenida es p (0.024), muestra que p es menor a 0.05, entonces se rechaza la hipótesis nula, por lo tanto, se acepta la hipótesis alterna y se concluye que mejora la calidad semántica del modelo software después de aplicar la transformación de modelos en el grupo experimental.

3.2.3 La aplicación de transformación de modelos mejora la calidad sintáctica del modelo software

$H_{0:}$ No mejora la calidad sintáctica del modelo software después de aplicar la transformación de modelos en el grupo experimental

$H_{a:}$ Mejora la calidad sintáctica del modelo software después de aplicar la transformación de modelos en el grupo experimental

Tabla 18

Prueba de Wilcoxon de la calidad sintáctica del modelo software del grupo experimental

		N	Rango promedio	Suma de rangos
DESPUÉS - ANTES	Rangos negativos	7[a]	11,64	81,50
	Rangos positivos	14[b]	10,68	149,50
	Empates	10[c]		
	Total	31		

a. DESPUÉS < ANTES b. DESPUÉS > ANTES c. DESPUÉS = ANTES

Estadísticos de contraste[b]	
	DESPUÉS - ANTES
Z	-1,196[a]
Sig. asintót. (bilateral)	,232

a. Basado en los rangos negativos.

b. Prueba de los rangos con signo de Wilcoxon

Interpretación:

La significancia estadística obtenida es p (0.232), muestra que p es mayor a 0.05, entonces se acepta la hipótesis nula, por lo tanto, se rechaza la hipótesis alterna y se concluye que no mejora la calidad sintáctica del modelo software después de aplicar la transformación de modelos en el grupo experimental.

3.2.4 La aplicación de transformación de modelos mejora la calidad pragmática del modelo software.

$H_{0:}$ No mejora la calidad pragmática del modelo software después de aplicar la transformación de modelos en el grupo experimental

$H_{a:}$ Mejora la calidad pragmática del modelo software después de aplicar la transformación de modelos en el grupo experimental

Tabla 19

Prueba de Wilcoxon de la calidad pragmática del modelo software del grupo experimental

		N	Rango promedio	Suma de rangos
DESPUÉS - ANTES	Rangos negativos	2[a]	6,00	12,00
	Rangos positivos	9[b]	6,00	54,00
	Empates	20[c]		
	Total	31		

a. DESPUÉS < ANTES b. DESPUÉS > ANTES c. DESPUÉS = ANTES

Estadísticos de contraste[b]	DESPUÉS - ANTES
Z	-2,111[a]
Sig. asintót. (bilateral)	,035

a. Basado en los rangos negativos.

b. Prueba de los rangos con signo de Wilcoxon

Interpretación:

La significancia estadística obtenida es p (0.035),), muestra que p es menor a 0.05, entonces se rechaza la hipótesis nula, por lo tanto, se acepta la hipótesis alterna y se concluye que mejora la calidad pragmática del modelo software después de aplicar la transformación de modelos en el grupo experimental.

IV. Discusión

Discusión

Pensamos que la tarea de la educación universitaria, particularmente en la Escuela de Ingeniería de Sistemas, debe facilitar un especial interés para que los estudiantes puedan desarrollar de manera sistémica el conocimiento y la práctica del complejo proceso de desarrollo de software. También consideramos que es tarea de las Instituciones educativas de nivel universitario, crear las condiciones que permitan a sus estudiantes adquirir las competencias necesarias que permitan afrontar el exigente mundo de la industria del software. Finalmente concientizar a docentes y alumnos el uso de modernas herramientas tecnológicas en el desarrollo de software.

Es por ello, según Serna (2013) los bajos índices de calidad mencionados están relacionados con dos tipos de problemas, por un lado con inconvenientes asociados al proceso de desarrollo de software. Por otro lado con dificultades en el desarrollo de las habilidades y competencias académicas de los profesionales que integran los equipos de desarrollo de software. De acuerdo a estos planteamientos se formuló la siguiente pregunta ¿En qué medida aplicar transformación de modelos mejora la calidad del modelo software en estudiantes de Ingeniería Universidad Cesar Vallejo Lima 2016?

Los antecedentes encontrados están en relación con el objeto de estudio y estos según Escalone (2006) en su tesis titulada: Estudio comparativo de los modelos y estándares de calidad del software. Buenos Aires Argentina; Llegó a la siguiente conclusión: Resulta fundamental evaluar la calidad del software, usando métricas que permitan cuantificar los datos obtenidos y modelos de calidad. Del mismo modo Meliá (2012) en su tesis titulada: Un método de desarrollo dirigido por modelos de arquitectura web. Alicante, España: Llegó a la siguiente conclusión: Los modelos y transformaciones proporcionan una trazabilidad precisa desde el análisis hasta la implementación y permiten acelerar el proceso de desarrollo. Finalmente Dioses (2012) en su tesis titulada: desarrollo de la metodología para la implementación de un sistema de gestión de calidad aplicado al software de computadora. Lima Perú: Llegó a la siguiente conclusión: Se ha demostrado que la calidad del software es aplicable a todo proceso, incluyendo el desarrollo de software

En función a estos enunciados los resultados de la investigación, demuestran:

La transformación de modelos proporciona un conjunto de facilidades que hace aconsejable utilizar en la construcción de software, la calidad de un modelo se puede medir a través de métricas y herramientas modernas. La calidad del modelo debe aplicarse en cada etapa del proceso, permitiendo garantizar un producto de calidad. La transformación de modelos es una estrategia que permite mejorar el proceso de desarrollo de software reflejado en la calidad de los modelos que se producen.

En concordancia a la variable calidad del modelo software: Antes del tratamiento se, observa que de 57 encuestados el 5.8% muestran un alto nivel, 86.0% muestran un regular nivel y un 8.8% muestran un bajo nivel en la calidad del modelo software. De igual forma después del tratamiento se, observa que de 57 encuestados el 19.3% muestran un alto nivel, 77.2% muestran un regular nivel y un 3.5% muestran un bajo nivel en la calidad del modelo software. Se puede demostrar un aumento en la calidad del modelo software después de aplicar el tratamiento.

En relación a la hipótesis general, la prueba de Wilcoxon para el pre test y post test, obtuvo la significancia estadística de 0.176, entonces se concluye que no mejora la calidad del modelo software después de aplicar la transformación de modelos en el grupo control. De igual forma, la prueba de Wilcoxon para el pre test y post test, obtuvo la significancia estadística de 0.009, entonces se concluye que mejora la calidad del modelo software después de aplicar la transformación de modelos en el grupo experimental. Se puede demostrar que mejora la calidad del modelo software después de aplicar el tratamiento.

En relación a las hipótesis específicas, la prueba de Wilcoxon para el pre test y post test, obtuvo la significancia estadística de 0.024, entonces se concluye que mejora la calidad semántica del modelo software después de aplicar la transformación de modelos en el grupo experimental. De igual forma la prueba de Wilcoxon para el pre test y post test, obtuvo la significancia estadística de 0.232, entonces se concluye que no mejora la calidad sintáctica del modelo software después de aplicar la transformación de modelos en el grupo experimental.

Finalmente, la prueba de Wilcoxon para el pre test y post test, obtuvo la significancia estadística de 0.035, entonces se concluye que mejora la calidad pragmática del modelo software después de aplicar la transformación de modelos en el grupo experimental.

Cabe señalar, que estos resultados nos permiten afirmar con criterio objetivo, que aplicar transformación de modelos en el desarrollo de software mejora la calidad del modelo software. Que usar modernas herramientas tecnológicas permite una mejora en el desarrollo de software. Finalmente, se considera que esta investigación es un aporte que permitirá contribuir a futuras investigaciones en nuevas técnicas en el campo del desarrollo del software tanto para estudiantes como investigadores.

V. Conclusiones

Conclusiones

Primero. Según la prueba de distribución de frecuencias aplicada a la investigación, se observó un aumento en el nivel alto de la calidad en un 13.5%, también se observó una disminución en el nivel regular de la calidad en un 9.8%, por último se observó un aumento en el nivel bajo en un 5.3%, por lo que concluimos que existe un aumento en la calidad del modelo software al aplicar la transformación de modelos.

Segundo. Según la prueba de Wilcoxon aplicada a la investigación, se observó que con un error de 0.176 (17.6%), la calidad del modelo software no mejora al aplicar la transformación de modelos en el grupo control. Del mismo modo se observó que con un error de 0.009 (0.09%) la calidad del modelo software mejora al aplicar la transformación de modelos en el grupo experimental.

Tercero. Según la prueba de Wilcoxon aplicada a la investigación, se observó que con un error de 0.024 (2.4%), la calidad semántica del modelo software mejora al aplicar transformación de modelos en el grupo experimental, de igual forma se observó que con un error de 0.232 (23.2%), la calidad sintáctica del modelo software no mejora al aplicar transformación de modelos en el grupo experimental (Este hecho debería ser estudiado en detalle a través de más experimentación), finalmente se observó que con un error de 0.035 (3.5%), la calidad pragmática del modelo software mejora al aplicar transformación de modelos en el grupo experimental. Por lo tanto concluimos que la transformación de modelos mejora la calidad semántica, sintáctica y pragmática del modelo software en el grupo experimental.

Cuarto. En el campo de la Ingeniería de Software, la aplicación del desarrollo dirigido por modelos a través de la transformación de modelos permite mejorar la calidad del modelo software como consecuencia de esto mejora la calidad del proceso y producto software.

VI. Recomendaciones

Recomendaciones

Primero. La universidad César Vallejo a través del Director de la Escuela de Ingeniería de Sistemas, debe promover capacitaciones sobre desarrollo de software dirigido por modelos para los docentes de Ingeniería de Software con la finalidad que los estudiantes mejoren el desarrollo de competencias técnico educativos en el área del desarrollo de software.

Segundo. Los docentes de la Escuela de Ingeniería de sistemas deben comprometerse con la investigación de modernos paradigmas y herramientas tecnológicas que permitan mejorar las competencias profesionales de sus estudiantes.

Tercero. Las autoridades educativas deben asumir la responsabilidad de liderar, proponer, incentivar y facilitar la investigación en el área de la Ingeniería del Software contribuyendo así con el logro de la calidad educativa.

Cuarto. Creemos que el desarrollo de software dirigido por modelos cambiará el futuro del desarrollo de software significativamente. Un argumento para sustentar esta opinión radica en que ya podemos lograr ventajas significativas en productividad y esfuerzo de mantenimiento, a través de la aplicación de MDD y sus herramientas de transformación, aunque MDD aún este inmaduro.

VII. Referencias

Referencias

Bennet, S., McRobb, S. y Farmer, R. (2007). *Análisis y diseño orientado a objetos de sistemas (3.a ed.).* Madrid: McGraw-Hill.

Calero, C., Moraga, A. y Piattini, M. (2012). *Calidad del producto y proceso software.* España: RA-MA Editorial.

Castro, R. (2013). *Aplicación del modelo CMMI nivel 2, SCRUM y PSP para el desarrollo de software.* Título Profesional de Ingeniero de Sistemas. Universidad Nacional de Ingeniería. Perú.

Cervantes, H., Velasco-Elizondo, P. y Castro, L. (2015). *Arquitectura de software Conceptos y ciclo de desarrollo.* México: Cengage Learning.

De Miguel, A., Piattini, M. y Marcos, E. (2000). *Diseño de base de datos relacionales.* Madrid: AlfaOmega Grupo Editor S.A. de C.V.

Deming, E. (1986). *Calidad, Productividad y Competitividad.* Madrid: Diaz de Santos, S.A.

Diaz, C. (2008). *Plan de calidad para la mejora del desarrollo de software.* Postgrado en sistemas de calida. Universidad Católica Andrés Bello. Colombia.

Dioses, S. (2012). *Desarrollo de la metodología de la implementación de un sistema de gestión de calidad aplicado al software de computadora.* Título de Ingeniero de Sistemas. Pontificia Universidad Cátolica del Perú. Perú.

Escalone, F. (2006). *Estudio comparativo de los Modelos y Estándares de Calidad del Software.* Maestría en Ingeniería de Calidad. Universidad Tecnologica Nacional. Argentina.

Ghezzi, C. (1991). *Fundamentals of Software Engineering.* Madrid: Prentice-Hall.

Guillen, O. (2016). *Guía de SPSS 22 para desarrollo de trabajos de investigación.* Málaga: Ando educando.

Higuera, M. G. (2012). Una introducción al desarrollo de software dirigido por modelos. *Seri Cientifica de la Universidad de las Ciencias Informáticas*, 11.

Johansen, O. (1993). *Introducción a la Teoría General de Sistemas.* Mexico: Limusa.

Larman, C. (2003). *UML y Patrones Una introducción al análisis y diseño orientado a objetos y al proceso unificado.* Madrid: Pearson Educación S.A.

Lawrence, S. (2002). *Ingenieria de software: Teoría y práctica.* Madrid: Pearson Educación S.A.

Llacctahuaman, J. (2015). *Sistema integral para mejorar la calidad de información en la recaudación tributaria de la Municipalidad Distrital de El Tambo.* Título profesional de Ingeniero de Sistemas. Universidad Nacional del Centro del Perú. Perú.

Meliá, S. (2012). *Un método de Desarrollo Dirigido por Modelos de Arquitectura Web.* Tesis Doctoral. Universidad de Alicante. Madrid.

Nolberto, V. P. (2008). *Estadística Inferencial Aplicada.* Lima: Unidad de Post Grado de la Facultad de Educación.

Pantaleo, G. y Rinaudo, L. (2015). *Ingeniería de Software.* Buenos Aires: Alfaomega Grupo Editor Argentino.

Pons, C. G. (2010). *Desarrollo de software dirigido por modelos. Cnceptos teóricos y su aplicación práctica.* Buenos Aires: Edulp.

Pressman, R. (2010). *Ingeniería del software Un enfoque práctico (7a. ed.).* Mexico: McGraw-Hill Interamericana Editores.

Sanchez, S., Sicilia, M. y Rodriguez, D. (2012). *Ingeniería del Software. Un enfoque desde la guía SWEBOK.* México: Alfaomega Grupo Editor S.A. de C.V.

Sarabia, A. (1995). *Toria General de Sistemas.* Madrid: Isdefe.

Serna, E. (2013). *Libro Blanco de la Ingeniería de Software en América Latina.* Medellin: Editorial Instituto Antioqueño de Investigación.

Somerville, L. (2005). *Ingeniería de software (7a. ed.).* Madrid: Rivera de Loira.

Supo, J. (2012). *Investigación Científica.* Arequipa: Bioestadistico EIRL.

Von Bertalanffy, L. (1986). *Teoría General de Sistemas.* Nueva York. : George Br.

Weitzenfeld, A. (2005). *Ingeniería de Software Orientada a Objetos con UML, Java e Internet.* Mexico: Thomson Editores.

Zavala, A. (2000). *Teoría de la Calidad.* Lima: San Marcos.

Anexos

ANEXO A: INSTRUMENTO VARIABLE I. TRANSFORMACIÓN DE MODELOS

1. Formación de los alumnos en los fundamentos de modelos conceptuales
2. Formación de los alumnos en los fundamentos de modelos lógicos
3. Entrenamiento en habilidades para la construcción de modelos conceptuales
4. Formación de los alumnos en los fundamentos de reglas de transformación de modelos
5. Entrenamiento en la aplicación de las reglas de transformación para la construcción de modelos lógicos
6. Puesta en práctica de la técnica de transformación de modelos

Experimento estudiado:

Dado el siguiente caso de estudio:

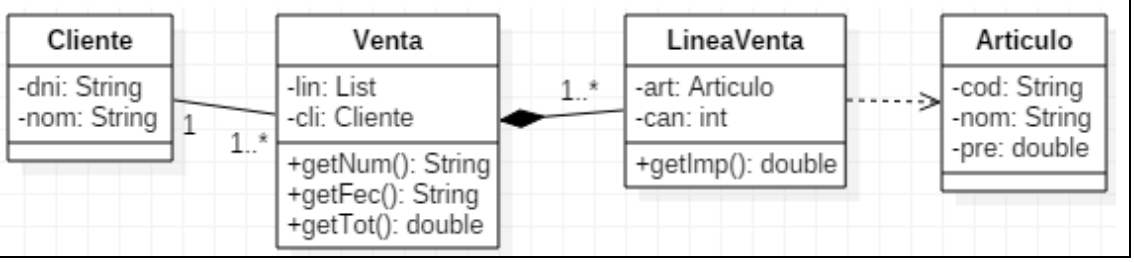
Gestión de ventas: La aplicación debe permitir al cliente consultar el artículo, luego agregar la línea de compra a la cesta y mostrar el importe y total de la venta; finalmente se graba la venta.

Modelo independiente de la plataforma (PIM): Modelo de dominio (Modelos conceptuales)

Reglas de transformación de modelos

Transformación:	Regla:
Objeto	Todo objeto se convierte en clase
Atributos	Todo atributo se convierte en propiedad y se define su visibilidad y tipo
Relación de creación interna	Toda relación de creación interna se convierte relación de asociación
Relación parte todo	Toda relación parten todo se convierte en relación de agregación o composición
Relación de creación externa	Toda relación de creación externa se convierte relación de dependencia
Relación de herencia	Toda relación de herencia se convierte en relación de generalización
Operación	Toda operación se convierte en método y se define sus parámetros y retorno

Modelo específico de la plataforma (PSM): Modelo de negocio (Modelos lógicos)



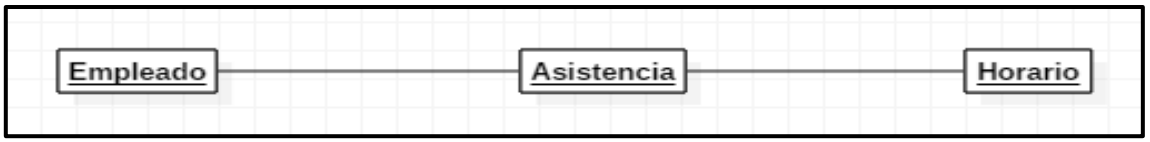
ANEXO B: INSTRUMENTO VARIABLE II. CALIDAD DEL MODELO SOFTWARE

Estimado alumno, el presente test es parte de un proyecto de investigación. Su finalidad es la obtención de información, acerca de la calidad de modelos.

Dado el siguiente caso de estudio:

Registro de asistencia: La aplicación debe permitir al empleado ingresar su DNI, luego almacenar y mostrar por unos segundos sus datos (nombre, cargo) la fecha actual, hora actual, los minutos de tardanza (si marco después de las 8:00) o minutos de adelanto (si marco antes de las 8:00); luego se muestra la página de registro de asistencia lista para nueva marcación.

Proponemos el modelo independiente de la plataforma (PIM): Modelo de dominio



Elabora el modelo específico de la plataforma (PSM): Modelo de negocio

Estimado alumno para concluir el experimento le pedimos que, por favor, complete la siguiente información en la que podrá exponer una visión subjetiva acerca de la calidad del modelo. Marque con una “X” la opción que considere más apropiada:

1	0
Verdadero	Falso

N°		1	0
1	El número total de clases es 4		
2	El número total de atributos es 12		
3	El número total de métodos es 6		
4	El número total de relaciones de asociación es 1		
5	El número total de relaciones de agregación es 1		
6	El número total de relaciones de dependencia es 1		
7	El número total de relaciones de generalización es 0		
8	Las clases tienen nombres únicos		
9	El nombre de cada clase esta en singular		
10	Todos los nombres de clases comienzan con mayúsculas		
11	Todos los atributos tienen especificado su tipo		
12	Todos los atributos tienen especificado la visibilidad		
13	Los nombres de los métodos son verbos		

14	Los nombres de los métodos empiezan en minúsculas
15	Los métodos tienen retorno
16	Toda clase están asociadas con al menos una clase
17	Todas las asociaciones y agregaciones tienen la multiplicidad especificada
18	El tiempo empleado en la solución fue menor a 10 minutos
19	El tiempo empleado en la solución fue entre 11 y 20 minutos
20	El tiempo empleado en la solución fue mayor a 20 minutos

ANEXO C: MATRIZ DE CONSISTENCIA

PROBLEMAS	OBJETIVOS	HIPOTESIS	VARIABLES E INDICADORES
Problema principal: ¿En qué medida aplicar transformación de modelos mejora la calidad del modelo software en estudiantes de Ingeniería Universidad César Vallejo Lima 2016?	**Objetivo general:** Demostrar que aplicar transformación de modelos mejora la calidad del modelo software en estudiantes de Ingeniería Universidad César Vallejo Lima 2016.	**Hipótesis general:** La aplicación de transformación de modelos mejora la calidad del modelo software en estudiantes de Ingeniería Universidad César Vallejo Lima 2016	**Variable 1: Transformación de modelos** **Dimensiones** / **Indicadores** Automatización / Manejo de herramientas Abstracción / Nivel de abstracción Estándares / Reglas de transformación
Problemas secundarios: ¿En qué medida aplicar transformación de modelos mejora la calidad semántica del modelo software en estudiantes de Ingeniería Universidad César Vallejo Lima 2016?	**Objetivos específicos:** Demostrar que aplicar transformación de modelos mejora la calidad semántica del modelo software en estudiantes de Ingeniería Universidad César Vallejo Lima 2016.	**Hipótesis específicas:** La aplicación de transformación de modelos mejora la calidad semántica del modelo software en estudiantes de Ingeniería Universidad César Vallejo Lima 2016.	**Variable 2: Calidad del modelo software** **Dimensiones** / **Indicadores** / **Ítems** / **Escala** / **Rangos** Calidad sintáctica / Complejidad / 1-7 / 0: Falso 1: Verdadero / Bajo 0 – 6 Calidad semántica / Corrección / 8-17 / / Regular 7-13 Calidad pragmática / Entendibilidad / 18-20 / / Alto 14-20
¿En qué medida la aplicación aplicar transformación de modelos mejora la calidad sintáctica del modelo software en estudiantes de Ingeniería Universidad César Vallejo Lima 2016?	Demostrar que aplicar transformación de modelos mejora la calidad sintáctica del modelo software en estudiantes de Ingeniería Universidad César Vallejo Lima 2016.	La aplicación de transformación de modelos mejora la calidad sintáctica del modelo software en estudiantes de Ingeniería Universidad César Vallejo Lima 2016.	
¿En qué medida aplicar transformación de modelos mejora la calidad pragmática del modelo software en estudiantes de Ingeniería Universidad César Vallejo Lima 2016?	Demostrar que aplicar transformación de modelos mejora la calidad pragmática del modelo software en estudiantes de Ingeniería Universidad César Vallejo Lima 2016.	La aplicación de transformación de modelos mejora la calidad pragmática del modelo software en estudiantes de Ingeniería Universidad César Vallejo Lima 2016.	
TIPO Y DISEÑO DE	**POBLACIÓN Y**	**TÉCNICAS E**	**ESTADISTICAS A UTILIZAR**

INVESTIGACIÓN	**MUESTRA**	**INSTRUMENTOS**	
TIPO:	**POBLACIÓN:**	**VARIABLE** : Calidad del	**DESCRIPTIVA:**
Experimental	67 estudiantes	modelo software	Prueba de distribución de frecuencias.
DISEÑO:	**TIPO MUESTRA:**	**TÉCNICA**:	**INFERENCIAL:**
Cuasi experimental	Muestreo estratificado	Experimentación	Prueba de Wilcoxon
MÉTODO:	**TAMAÑO MUESTRA:**	**INSTRUMENTO**:	
Estadístico	57 estudiantes	Test	
		AUTOR:	
		Chunga Huatay Edwin	
		AÑO: 2016	
		MONITOREO:	
		AMBITO APLICA.:	
		U. Cesar Vallejo	
		FORMA ADM.	

ANEXO D: CONSENTIMIENTO POR LA INSTITUCIÓN

UNIVERSIDAD CÉSAR VALLEJO

Los Olivos, 21 de agosto de 2017

Oficio N° 064-2017-UCV/VA

Señor
Edwin Chunga Huatay

Presente.-

Asunto: Consentimiento de Desarrollo de Investigación

De mi consideración

Es grato saludarlo y, en atención a su solicitud de fecha 18 de agosto, indicarle que el Vicerrectorado de Investigación, para evaluar la pertinencia de su pedido, necesita contar con la Resolución de aprobación de su Proyecto de Investigación, un resumen del proyecto, así como precisar su importancia y la metodología a emplear, ya que se trata de un trabajo experimental.

La información requerida debe remitirse a la dirección electrónica: vicerrectoradoacademico@ucv.edu.pe

Aprovecho la oportunidad para expresarle las muestras de mi especial consideración y estima.

Atentamente,

UCV

Heraclio Campana Añasco
Vicerrector Académico

ANEXO E: BASE DE DATOS DE LA VARIABLE CALIDAD DEL MODELO SOFTWARE ANTES Y DESPUES DEL GRUPO CONTROL

Antes	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11	p12	p13	p14	p15	p16	p17	p18	p19	p20
Alumno 1	0	0	0	0	0	0	1	1	1	1	0	1	0	0	0	0	0	0	0	1
Alumno 2	0	0	0	1	1	0	1	1	1	1	1	0	1	0	1	1	1	0	1	1
Alumno 3	0	0	0	0	0	0	1	1	0	1	0	1	0	0	0	0	0	0	1	1
Alumno 4	0	0	0	1	0	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1
Alumno 5	0	0	0	1	0	0	1	1	1	1	0	1	0	0	0	1	0	0	1	1
Alumno 6	0	0	0	1	1	0	1	1	1	1	1	1	1	1	1	1	0	0	1	1
Alumno 7	1	0	1	1	1	0	1	0	1	1	0	1	1	1	1	1	0	0	1	1
Alumno 8	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1
Alumno 9	1	1	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	1
Alumno 10	1	0	0	1	1	0	1	1	1	1	1	1	0	0	0	1	0	0	1	1
Alumno 11	1	0	1	1	1	0	1	1	1	1	1	1	1	1	1	1	0	0	0	1
Alumno 12	0	0	0	0	0	0	1	1	0	1	1	1	1	1	1	1	0	0	0	1
Alumno 13	1	1	0	1	0	0	1	1	1	0	1	1	0	0	0	1	0	1	1	1
Alumno14	1	0	0	1	1	0	1	1	0	1	1	1	1	0	0	1	0	0	0	1
Alumno 15	1	1	1	1	1	0	1	0	1	1	0	0	1	1	0	1	0	0	0	1
Alumno 16	1	1	0	1	0	0	1	1	1	1	0	1	0	0	0	1	0	0	0	1
Alumno 17	0	0	0	1	0	0	1	1	1	1	1	0	0	0	0	1	0	0	1	1
Alumno 18	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	1	1	1
Alumno 19	0	0	0	1	0	0	1	1	1	1	1	0	0	0	0	1	1	1	1	1
Alumno 20	0	0	0	1	0	0	1	1	1	1	0	0	0	0	0	1	0	1	1	1
Alumno 21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
Alumno 22	1	0	0	1	0	0	1	1	1	0	0	1	0	0	0	1	0	0	0	1
Alumno 23	0	0	0	1	0	0	1	1	1	1	1	0	1	0	1	1	0	0	1	1
Alumno 24	1	1	0	1	0	0	1	1	1	1	1	0	0	0	0	1	0	0	0	1
Alumno 25	1	0	0	1	1	1	1	1	0	1	1	0	1	0	0	1	0	0	0	1
Alumno 26	0	0	0	1	0	0	1	1	1	1	0	1	0	0	0	1	0	1	0	1

Después	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11	p12	p13	p14	p15	p16	p17	p18	p19	p20
Alumno 1	1	1	0	1	1	1	1	1	1	1	1	0	1	1	0	1	0	1	1	1
Alumno 2	0	0	0	0	0	0	1	1	0	1	1	0	0	0	0	1	0	1	1	1
Alumno 3	1	1	0	1	0	1	1	1	1	1	1	0	0	0	0	1	1	1	1	1
Alumno 4	1	1	0	0	1	0	1	1	1	1	0	0	0	0	0	0	0	1	1	1
Alumno 5	1	1	0	0	0	0	1	1	1	1	1	0	1	1	1	1	0	0	1	1
Alumno 6	1	1	0	1	0	0	1	1	1	0	1	0	0	0	0	1	0	0	1	1
Alumno 7	1	1	0	1	1	1	1	1	0	1	1	0	1	0	1	1	0	0	1	1
Alumno 8	0	1	0	1	1	0	1	1	1	1	0	1	0	0	0	1	0	0	1	1
Alumno 9	1	0	0	1	0	1	1	1	1	0	1	0	1	0	0	1	0	0	1	1
Alumno 10	1	1	1	1	1	0	1	1	1	1	1	0	0	0	0	1	0	0	1	1
Alumno 11	1	1	0	1	1	0	1	1	0	1	0	0	0	0	0	1	0	0	1	1
Alumno 12	1	1	0	1	1	0	1	1	1	1	0	0	0	0	0	1	0	0	1	1
Alumno 13	0	1	0	1	0	0	1	1	1	1	0	0	0	0	0	0	0	1	1	1
Alumno14	0	0	0	0	0	0	1	1	1	1	1	0	1	0	1	1	1	1	1	1
Alumno 15	0	0	0	1	1	0	1	0	1	1	0	0	0	0	0	0	0	1	1	1
Alumno 16	1	0	0	0	0	0	1	1	1	1	1	0	1	0	0	1	0	1	1	1
Alumno 17	0	0	0	1	1	0	1	1	0	1	1	0	0	0	0	1	0	1	1	1
Alumno 18	0	0	0	1	0	0	1	1	1	1	1	0	1	0	1	1	0	0	1	1
Alumno 19	1	1	1	1	0	0	1	1	1	0	1	1	0	1	1	0	0	1	1	1
Alumno 20	0	0	1	1	0	0	1	1	1	0	1	1	1	0	0	1	0	0	1	1
Alumno 21	0	0	0	1	1	0	0	1	1	1	0	0	0	0	0	0	0	0	1	1
Alumno 22	1	1	0	0	0	0	1	0	0	1	0	1	0	0	0	1	0	1	0	1
Alumno 23	0	0	0	1	0	0	1	1	1	1	1	0	0	0	0	1	0	0	1	1
Alumno 24	1	0	0	1	1	0	1	1	0	1	1	0	1	0	1	1	0	0	0	1
Alumno 25	1	1	0	1	1	1	1	1	1	1	1	0	1	0	1	1	0	0	0	1
Alumno 26	0	0	0	1	0	0	0	1	1	1	0	0	1	0	1	1	0	0	0	1

BASE DE DATOS DE LA VARIABLE CALIDAD DEL MODELO SOFTWARE ANTES Y DESPUES DEL GRUPO EXPERIMENTAL

Antes	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11	p12	p13	p14	p15	p16	p17	p18	p19	p20
Alumno 1	0	1	0	1	0	0	1	1	1	0	0	0	1	1	1	1	0	1	1	1
Alumno 2	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	1	1
Alumno 3	0	0	0	1	0	0	1	1	1	0	0	0	1	1	1	1	0	1	1	1
Alumno 4	0	0	0	1	1	0	1	1	1	1	1	0	1	1	1	1	0	0	1	1
Alumno 5	0	0	0	1	0	0	1	1	1	1	1	0	0	0	0	1	0	0	1	1
Alumno 6	0	0	0	1	0	0	1	1	1	1	1	0	0	0	0	1	0	0	1	1
Alumno 7	0	0	0	1	0	0	1	1	1	1	1	0	0	0	0	1	0	1	1	1
Alumno 8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
Alumno 9	0	0	0	1	0	0	1	1	1	1	0	0	0	0	0	1	0	0	1	1
Alumno 10	0	1	0	1	0	0	1	1	1	1	0	1	0	0	0	1	0	0	1	1
Alumno 11	0	1	0	1	0	0	1	1	1	1	0	0	0	0	0	1	0	1	1	1
Alumno 12	0	0	0	0	0	0	1	1	1	1	0	1	0	0	0	1	0	1	1	1
Alumno 13	0	0	0	1	0	0	1	1	1	1	0	0	0	0	0	1	0	0	1	1
Alumno 14	0	0	0	1	0	0	1	1	1	1	0	1	0	0	0	1	0	0	1	1
Alumno 15	0	0	0	1	0	0	1	1	1	1	0	1	0	0	0	1	0	0	1	1
Alumno 16	0	0	0	1	0	0	1	1	1	1	1	0	1	0	0	1	0	1	1	1
Alumno 17	0	1	0	1	0	0	1	1	1	1	0	0	0	0	0	0	0	0	1	1
Alumno 18	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	1	0	1	1	1
Alumno 19	0	1	0	1	0	0	1	1	1	1	0	1	0	0	0	1	0	1	1	1
Alumno 20	0	0	0	1	0	0	1	1	1	1	0	0	0	0	0	1	1	1	1	1
Alumno 21	0	0	0	1	0	0	1	1	1	1	1	0	0	0	0	0	0	1	1	1
Alumno 22	0	0	0	0	1	0	1	1	1	1	1	0	1	0	0	0	0	0	1	1
Alumno 23	0	1	0	0	1	0	1	1	1	1	1	0	0	0	0	1	0	0	1	1
Alumno 24	0	0	0	1	1	0	1	1	1	1	1	0	0	0	0	0	0	0	1	1
Alumno 25	0	1	0	0	1	0	1	1	1	1	1	0	0	0	0	1	0	1	1	1
Alumno 26	0	0	0	1	1	0	1	1	1	1	1	0	0	0	0	0	0	0	1	1
Alumno 27	0	0	0	0	0	0	1	1	1	1	1	0	1	0	1	1	0	1	1	1
Alumno 28	0	1	0	1	0	0	1	1	1	1	1	0	0	0	0	1	0	1	1	1
Alumno 29	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	1	1	1
Alumno 30	0	0	0	1	0	0	1	1	1	1	0	0	0	0	0	1	0	1	1	1
Alumno 31	0	0	0	1	0	0	1	1	1	1	0	0	0	0	0	1	0	1	1	1

Después	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11	p12	p13	p14	p15	p16	p17	p18	p19	p20
Alumno 1	1	1	0	1	1	0	1	1	1	1	1	0	1	0	1	1	0	0	1	1
Alumno 2	0	1	0	1	0	0	1	1	1	1	0	0	0	0	0	1	0	1	1	1
Alumno 3	0	1	0	1	0	0	1	1	1	1	0	0	0	0	0	1	0	1	1	1
Alumno 4	1	1	0	1	1	0	1	1	1	1	1	0	1	1	1	1	0	1	1	1
Alumno 5	0	1	0	1	0	0	1	1	1	1	1	0	1	0	0	1	0	0	1	1
Alumno 6	0	1	0	1	0	0	1	1	1	1	1	0	1	1	1	1	0	1	1	1
Alumno 7	1	1	1	1	0	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1
Alumno 8	0	1	0	1	0	0	1	1	1	1	0	1	0	0	0	1	0	0	1	1
Alumno 9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1
Alumno 10	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	1	1
Alumno 11	0	1	0	1	0	0	1	1	1	1	0	0	0	0	0	1	0	1	1	1
Alumno 12	1	1	0	1	1	1	1	1	1	1	1	0	1	0	0	1	0	1	1	1
Alumno 13	0	1	0	1	0	0	1	1	1	1	0	1	0	0	0	1	0	0	1	1
Alumno 14	1	1	0	1	1	0	1	1	1	1	1	0	0	0	0	1	0	1	1	1
Alumno 15	0	1	0	1	0	0	1	1	1	1	0	1	0	0	0	1	0	1	1	1
Alumno 16	1	0	0	1	0	0	1	1	1	1	0	0	0	0	0	1	0	1	1	1
Alumno 17	1	0	0	1	0	0	1	1	1	1	0	1	0	0	0	1	0	1	1	1
Alumno 18	0	1	0	1	0	0	1	1	1	1	1	0	1	0	0	1	0	1	1	1
Alumno 19	0	1	0	0	0	0	1	1	1	1	1	0	1	0	0	1	0	1	1	1
Alumno 20	0	1	0	0	0	0	1	1	1	1	1	0	1	1	1	0	0	1	1	1
Alumno 21	0	0	0	1	0	0	1	1	1	1	0	0	1	0	0	0	0	1	1	1
Alumno 22	0	1	0	1	0	0	1	1	1	1	0	0	1	0	0	1	0	1	1	1
Alumno 23	1	0	0	0	0	0	1	1	1	1	1	0	0	0	0	1	0	0	1	1
Alumno 24	1	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	1	1	1
Alumno 25	0	0	0	1	0	0	1	1	1	1	1	0	0	0	0	0	0	1	1	1

Alumno 26	0	1	0	0	0	0	1	1	1	1	0	0	1	0	0	1	1	1	1	1
Alumno 27	0	0	0	1	0	0	1	1	1	1	1	0	1	0	0	0	0	1	1	1
Alumno 28	0	1	0	1	0	0	1	1	1	1	0	0	0	0	0	1	0	1	1	1
Alumno 29	0	1	0	0	0	0	1	1	1	1	0	0	0	0	0	1	0	1	1	1
Alumno 30	0	1	0	1	0	0	1	1	1	1	0	0	1	0	0	0	0	1	1	1
Alumno 31	1	1	1	1	1	0	1	1	1	1	0	0	1	1	1	1	1	1	1	1

ANEXO F: VALIDACION DEL INSTRUMENTOS DE MEDICIÓN POR JUICIO DE EXPERTOS

DEFINICIÓN CONCEPTUAL DE LAS VARIABLES

Variable: Calidad del modelo software

Calero, Moraga y Piattini (2012) definen la calidad del modelo software: El un grupo de características de calidad y sus relaciones, acompañado de las técnicas para evaluar tales características (medidas, listas de control, convenciones de modelado, técnicas de inspección). La calidad de los modelos tiene una gran importancia, ya que determinará la calidad de los productos software finalmente implementado (p.206).

Dimensiones de la variable:

Dimensión 1: Calidad semántica

Según, Calero, Moraga y Piattini (2012) sostienen al respecto: Es el grado con el que el modelo representa correctamente el dominio del problema. Existen dos objetivos en esta dimensión: 1) Validez: significa que todo lo que refleja el modelo es correcto y relevante para el problema modelado, y 2) Compleción: significa que el modelo refleja todo lo que es relevante y correcto para el problema objeto del modelado (p.211).

Dimensión 2: Calidad sintáctica

Según, Calero, Moraga y Piattini (2012) sostienen al respecto: “Es el grado con el que el modelo no contiene errores sintácticos. La corrección sintáctica de un modelo implica que todos los elementos que se incluyen en el modelo estén de acuerdo con la sintaxis del lenguaje de modelado” (p.211).

Dimensión 3: Calidad pragmática

Según, Calero, Moraga y Piattini (2012) sostienen al respecto: “Es la comprensión del modelo desde la perspectiva de los implicados en el proceso de modelado (modeladores, diseñadores, desarrolladores, etc.) La calidad pragmática captura el grado con el que el modelo es correctamente entendido por los implicados” (p.211)

.

Indicadores de la variable:

Indicador 1: Complejidad

Según, Calero, Moraga y Piattini (2012) sostienen al respecto: “El esfuerzo requerido para comprender un modelo y su importancia para la comunicación, comprensión y modificación” (p.207).

Indicador 2: Corrección

Según, Calero, Moraga y Piattini (2012) sostienen al respecto: “La inclusión de elementos correctos y relaciones correctas entre ellos, e inclusión de afirmaciones correctas sobre el dominio. La no violación de normas y convenciones” (p.211).

Indicador 3: Entendibilidad

Según, Calero, Moraga y Piattini (2012) sostienen al respecto: Se define como el ser entendible por los usuarios; tanto usuarios humanos como herramientas. Para usuarios humanos, varios aspectos impactan en la comprensibilidad, como la estética de los diagramas, la organización de un modelo, la simplicidad o complejidad de un modelo, la utilización de conceptos familiares para los usuarios o seleccionados de la ontología del dominio (p.211).

MATRIZ DE OPERACIONALIZACION DE LA VARIABLE: CALIDAD DEL MODELO SOFTWARE

Dimensiones	Indicadores	Ítems	Escalas	Niveles
Calidad semántica	- Complejidad.	El número total de clases es 4		
		El número total de atributos es 12		
		El número total de métodos es 6		
		El número total de relaciones de asociación es 1		
		El número total de relaciones de agregación es 1		
		El número total de relaciones de dependencia es 1		
		El número total de relaciones de generalización es 0		
Calidad sintáctica	- Corrección	Las clases tienen nombres únicos		
		El nombre de cada clase esta en singular	0:	Bajo
		Todos los nombres de clases comienzan con mayúsculas	Falso	Regular
		Todos los atributos tienen especificado su tipo	1:	
		Todos los atributos tienen especificado la visibilidad	Verdadero	Alto
		Los nombres de los métodos son verbos		
		Los nombres de los métodos empiezan en minúsculas		
		Los métodos tienen retorno		
		Toda clase están asociadas con al menos una clase		
		Todas las asociaciones y agregaciones tienen la multiplicidad especificada		
Calidad pragmática	- Entendibilidad	El tiempo empleado en la solución fue menor a 10 minutos		
		El tiempo empleado en la solución fue entre 11 y 20 minutos		
		El tiempo empleado en la solución fue mayor a 20 minutos		

CERTIFICADO DE VALIDEZ DEL CONTENIDO DEL INSTRUMENTO QUE MIDE LA CALIDAD DEL MODELO

UCV UNIVERSIDAD CÉSAR VALLEJO
ESCUELA DE POSTGRADO

CERTIFICADO DE VALIDEZ DE CONTENDO DEL INSTRUMENTO QUE MIDE .....

Nº	DIMENSIONES / ítems	Pertinencia[1]		Relevancia[2]		Claridad[3]		Sugerencias
	DIMENSIÓN 1: Vías	Si	No	Si	No	Si	No	
1	¿Han realizado anteriormente el asfaltado de la vía?	X		X		X		
2	¿Le toma mucho tiempo movilizarse por esta vía?	X		X		X		
3	¿Es alta la frecuencia de tránsito en la vía?	X		X		X		
4	¿Cree Ud. que los vehículos pesados no deberían circular por esta vía?	X		X		X		
	DIMENSIÓN 2: Patologías	Si	No	Si	No	Si	No	
5	¿Realizan mantenimiento continuo en la vía?		X	X		X		
6	¿Realizan mantenimiento al sistema de drenaje pluvial?	X		X		X		
7	¿Se siente Ud. seguro y cómodo al transitar por la vía?	X		X		X		
8	¿La vía se deteriora con facilidad?	X			X	X		
	DIMENSIÓN 3: Productividad	Si	No	Si	No	Si	No	
9	¿Ud. se ha beneficiado con la construcción de la vía?	X		X			X	
10	¿La construcción de la vía a elevado su nivel socio económico?	X		X		X		
11	¿Se ha inundado la vía por las lluvias?	X		X		X		
12	¿Existen fallas de deterioro de la vía?	X		X		X		
	DIMENSIÓN 4: Seguridad	Si	No	Si	No	Si	No	
13	¿Hay accidentes frecuentes en la vía?		X	X		X		
14	¿Existen señales de seguridad de tránsito en la vía?	X		X		X		
15	¿Ud. ha sufrido asalto o robo en la vía?	X		X		X		
16	¿Existen medidas de seguridad en la infraestructura vial?	X		X		X		

Observaciones (precisar si hay suficiencia): _____

Opinión de aplicabilidad: **Aplicable [ ]** **Aplicable después de corregir [ ]** **No aplicable [ ]**

Apellidos y nombres del juez validador. Dr/ Mg: Sinche Rosillo, Fredy M. **DNI:** 40327565

Especialidad del validador: Gestión de Proyectos

[1]**Pertinencia:** El ítem corresponde al concepto teórico formulado.
[2]**Relevancia:** El ítem es apropiado para representar al componente o dimensión específica del constructo
[3]**Claridad:** Se entiende sin dificultad alguna el enunciado del ítem, es conciso, exacto y directo

Nota: Suficiencia, se dice suficiencia cuando los ítems planteados son suficientes para medir la dimensión

03 de Diciembre del 20 16

Firma del Experto Informante.

UCV UNIVERSIDAD CÉSAR VALLEJO

ESCUELA DE POSTGRADO

CERTIFICADO DE VALIDEZ DE CONTENDO DEL INSTRUMENTO QUE MIDE .....

Nº	DIMENSIONES / ítems	Pertinencia[1]		Relevancia[2]		Claridad[3]		Sugerencias
	DIMENSIÓN 1: Vías	Si	No	Si	No	Si	No	
1	¿Han realizado anteriormente el asfaltado de la vía?							
2	¿Le toma mucho tiempo movilizarse por esta vía?							
3	¿Es alta la frecuencia de tránsito en la vía?							
4	¿Cree Ud. que los vehículos pesados no deberían circular por esta vía?							
	DIMENSIÓN 2: Patologías	Si	No	Si	No	Si	No	
5	¿Realizan mantenimiento continuo en la vía?							
6	¿Realizan mantenimiento al sistema de drenaje pluvial?							
7	¿Se siente Ud. seguro y cómodo al transitar por la vía?							
8	¿La vía se deteriora con facilidad?							
	DIMENSIÓN 3: Productividad	Si	No	Si	No	Si	No	
9	¿Ud. se ha beneficiado con la construcción de la vía?							
10	¿La construcción de la vía a elevado su nivel socio económico?							
11	¿Se ha inundado la vía por las lluvias?							
12	¿Existen fallas de deterioro de la vía?							
	DIMENSIÓN 4: Seguridad	Si	No	Si	No	Si	No	
13	¿Hay accidentes frecuentes en la vía?							
14	¿Existen señales de seguridad de tránsito en la vía?							
15	¿Ud. ha sufrido asalto o robo en la vía?							
16	¿Existen medidas de seguridad en la infraestructura vial?							

Observaciones (precisar si hay suficiencia): _____

Opinión de aplicabilidad: **Aplicable** [✓] **Aplicable después de corregir** [ ] **No aplicable** [ ]

Apellidos y nombres del juez validador Dr/ Mg: JUAN CAMPOS LUIS. **DNI:** 08076105

Especialidad del validador: ING. DE SISTEMAS

03 de 12 del 2016

[1]**Pertinencia:** El ítem corresponde al concepto teórico formulado.
[2]**Relevancia:** El ítem es apropiado para representar al componente o dimensión específica del constructo
[3]**Claridad:** Se entiende sin dificultad alguna el enunciado del ítem, es conciso, exacto y directo

Nota: Suficiencia, se dice suficiencia cuando los ítems planteados son suficientes para medir la dimensión

---

Firma del Experto Informante.

ANEXO G: ARTICULO CINTÍFICO

ESCUELA DE POSGRADO
UNIVERSIDAD CÉSAR VALLEJO

Transformación de modelos para mejorar la calidad del modelo software

Br. Chunga Huatay, Edwin José

Escuela de Posgrado

Universidad Cesar Vallejo Filial Lima Norte

Resumen

Esta investigación se titula “Transformación de modelos para mejorar la calidad del modelo software” ha tenido como objetivo demostrar que el usar transformación de modelos al modelar diagramas de UML se mejora la calidad del modelo software en estudiantes de ingeniería, es de método hipotético deductivo, tipo experimental – longitudinal, diseño cuasi experimental, la muestra fue constituida por 67 estudiantes, tipo muestreo no probabilístico, estrategia de selección aleatorio estratificado, técnica de recopilación de datos “Encuesta”, tipo de instrumentos “Test”, los resultado según la prueba de distribución presentan un aumento de 13.5% en el nivel alto de la calidad del software, una disminución del 9.8% en el nivel regular de la calidad del software, un aumento del 5.3% en el nivel bajo de la calidad del software y según la prueba de Wilcoxon que con un error de 0.009 (0.09%) se comprobó que la calidad del modelo software mejora al aplicar la transformación de modelos.

Palabras claves: Transformación de modelos, Calidad del modelo software

Abstract

This research is entitled "Transformation of models to improve the quality of software model" has aimed to demonstrate that using model transformation when modeling UML diagrams improves the quality of the software model in engineering students, is hypothetical deductive method, Experimental type - longitudinal, quasi experimental design, the sample consisted of 67 students, type non - probabilistic sampling, strategy stratified random selection, data collection technique "Survey", type of instruments "Test", results according to the test Distribution show a 13.5% increase in the high level of software quality, a decrease of 9.8% in the regular level of software quality, a 5.3% increase in the low level of software quality and according to the test Of Wilcoxon that with an error of 0.009 (0.09%) it was verified that the quality of the software model improves when applying the transfor Mation of models.

Key words: Model transformation, Software model quality

Introducción

Pensamos que la tarea de la educación universitaria, particularmente en la Escuela de Ingeniería de Sistemas, debe facilitar un especial interés para que los estudiantes puedan desarrollar de manera sistémica el conocimiento y la práctica del complejo proceso de desarrollo de software. También consideramos que es tarea de las Instituciones educativas de nivel universitario, crear las condiciones que permitan a sus estudiantes adquirir las competencias necesarias que permitan afrontar el exigente mundo de la industria del software.

Es por ello, según Serna (2013) los bajos índices de calidad están relacionados con dos tipos de problemas, por un lado con inconvenientes asociados al proceso de desarrollo de software y a su vez, con dificultades en el desarrollo de las habilidades y competencias académicas de los profesionales que integran los equipos.

De acuerdo a este planteamiento se formuló la siguiente pregunta ¿En qué medida aplicar transformación de modelos mejora la calidad del modelo software?

Se espera aportar con nuevos conocimientos en el campo del desarrollo de software permitiendo plantear soluciones a problemas que afectan la construcción de software, proponemos una técnica llamada transformación de modelos que permitirá mejorar la calidad en la construcción de modelos de software en estudiantes y personas involucradas en la industria del software.

Antecedentes del problema

Los antecedentes encontrados según Escalone (2006) en su tesis titulada: Estudio comparativo de los modelos y estándares de calidad del software. Buenos Aires Argentina; Llegó a la siguiente conclusión: Resulta fundamental evaluar la calidad del software, usando métricas que permitan cuantificar los datos obtenidos y modelos de calidad.

Del mismo modo Meliá (2012) en su tesis titulada: Un método de desarrollo dirigido por modelos de arquitectura web. Alicante, España: Llegó a la siguiente conclusión: Los modelos y transformaciones proporcionan una trazabilidad precisa desde el análisis hasta la implementación y permiten acelerar el proceso de desarrollo.

Finalmente Dioses (2012) en su tesis titulada: desarrollo de la metodología para la implementación de un sistema de gestión de calidad aplicado al software de computadora. Lima Perú: Llegó a la siguiente conclusión: Se ha demostrado que la calidad del software es aplicable a todo proceso, de desarrollo de software.

En función a estos enunciados los resultados de la investigación, demuestran:

Que la transformación de modelos proporciona un conjunto de facilidades que hace aconsejable utilizar en la construcción de software y que la calidad de un modelo se puede medir a través de métricas y herramientas, finalmente la calidad debe aplicarse en cada etapa del proceso, permitiendo garantizar un producto de calidad.

Revisión de la literatura

Para definir el desarrollo de software guiado por modelos MDD, Pons (2010) sostiene al respecto: “El proceso de software guiado por modelos MDD se ha transformado en un nuevo estándar de desarrollo de software. Promete perfeccionar el proceso de construcción de software apoyándose en un proceso gobernado por modelos y apoyado por poderosas herramientas” (p.28).

Contribuyo que el desarrollo de software moderno requiere de nuevos paradigmas que usen herramientas que remplacen el esfuerzo humano y fortalezcan su creatividad

Para definir un modelo, Pons (2010) sostiene al respecto: “En el contexto científico un modelo puede ser una entidad, un esquema o una entidad concreta. Es una estructuración abstracta o concreta a escala de un proceso o sistema, con el fin de evaluar su naturaleza” (p.39).

Bennet, McRobb y Farmer (2007) indican al respecto: “En cualquier proyecto de desarrollo que aspire a producir artefactos útiles, el objetivo principal del análisis y del diseño se centra en los modelos” (p.104).

Para definir la transformación de modelos, Pons (2010) sostiene al respecto: “Es la obtención automática de un modelo objetivo desde un modelo básico, respetando a una definición de transformación. Esta definición es un grupo de reglas que juntas explican como un modelo en el lenguaje básico puede ser transformado en un

modelo en el lenguaje objetivo” (p.53).

Contribuyo que la transformación de modelos es la obtención de un modelo final a partir del tratamiento de un modelo inicial respetando reglas y lenguajes de transformación

Figura 3: Definición de transformación entre lenguajes

Para ilustrar acerca de los modelos producidos a lo largo del proceso de desarrollo dirigido por modelo, Pons (2010) sostiene al respecto: “En el MDD los diagramas se van generando desde los más imaginarios a los más reales a través de procesos de conversión y/o simplificación, hasta obtener el programa fuente empleando una última conversión” (p.40).

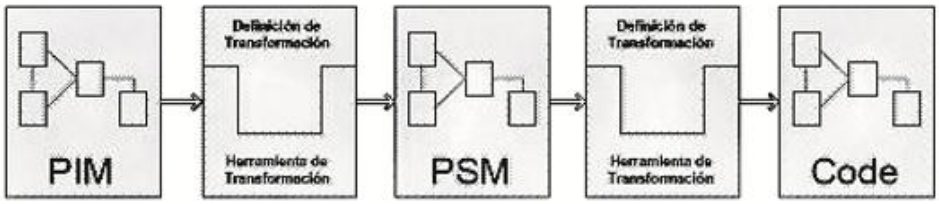


Figura 4: Los modelos empleados en el MDD

Para ilustrar un ejemplo del proceso de transformación de modelos, Pons (2010) sostiene al respecto: “Conversión de un esquema PIM desarrollado en UML a un esquema de codificación en Java. Transformaremos el diagrama de clases del software de venta de libros en las clases implementadas en Java correspondientes a ese esquema” (p.54).

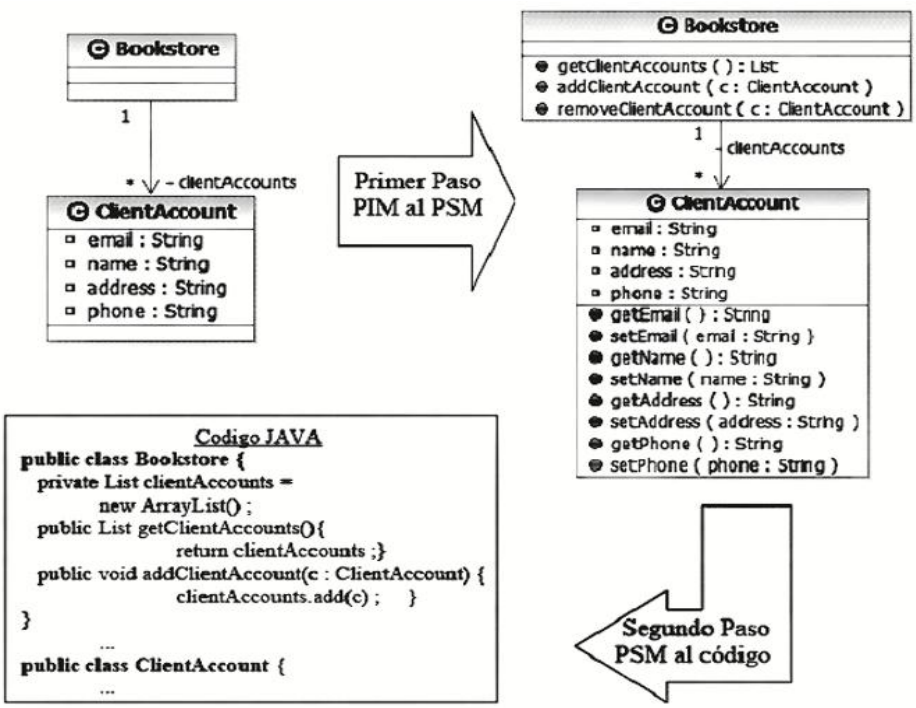


Figura 5: Ejemplo de conversión del PIM al PSM y al CODIGO

Para definir la calidad del modelo software, Calero, Moraga y Piattini (2012) sostienen al respecto: “Es un conjunto de atributos de calidad y sus asociaciones, acompañado de las técnicas para evaluar tales características (medidas, listas de control, convenciones de modelado, técnicas de inspección). La calidad de los modelos tiene una gran importancia, ya que determinará la calidad de los productos software finalmente implementados” (p.206).

Para definir las dimensiones de la calidad de modelos software, Calero, Moraga y Piattini (2012) sostienen al respecto: “Podemos decir que si bien no existe un modelo de calidad para modelos consensuado y valido, hay cierta tendencia a considerar tres tipos de calidad: calidad semántica, sintáctica y pragmática” (p.210)

Para definir la calidad semántica, Calero, Moraga y Piattini (2012) sostienen al respecto: “Es el grado con el que el modelo representa correctamente el dominio del problema” (p.211).

Para definir la calidad sintáctica, Calero, Moraga y Piattini (2012) sostienen al respecto: “Es el grado con el que el modelo no contiene errores sintácticos. La

corrección sintáctica de un modelo implica que todos los elementos que se incluyen en el modelo estén de acuerdo con la sintaxis del lenguaje de modelado" (p.211).

Para definir la calidad pragmática, Calero, Moraga y Piattini (2012) sostienen al respecto: "Es la comprensión del modelo desde la perspectiva de los implicados en el proceso de modelado (modeladores, diseñadores, desarrolladores, etc.) La calidad pragmática captura el grado con el que el modelo es correctamente entendido por los implicados" (p.211).

Problema

Sobre la base de esta problemática, el presente estudio propone emplear el paradigma del desarrollo de software guiado por modelos y sus transformaciones en el campo de la construcción del mismo permitiendo que se logre mejorar la calidad del software en especial los modelos por lo que proponemos como problemática: ¿En qué medida aplicar transformación de modelos mejora la calidad del modelo software en estudiantes de Ingeniería?

Objetivo

El objetivo es demostrar que la transformación de modelos mejora la calidad del modelo software en estudiantes de Ingeniería

Método

La investigación es de tipo experimental, diseño cuasi experimental, la población de estudio está constituida por 67 unidades de estudio formados por estudiantes de Ingeniería de Sistemas, del curso Ingeniería de Software el tamaño de la muestra fue de 57 estudiantes el tipo muestreo no probabilístico y la estrategia de selección aleatorio estratificado.

Usamos el procedimiento de recopilación de datos "Encuesta" conformada por formularios basados en un caso de estudio y preguntas para obtener el comportamiento de las variables, aplicamos el instrumento de tipo TEST, la validación se obtuvo mediante el procedimiento juicio de expertos y la confiabilidad que se aplicó es el coeficiente de Cronbach.

Elaboramos tablas y gráficos estadísticas para el análisis de resultados a partir de la distribución de frecuencias y para la parte inferencial aplicamos pruebas no paramétricas para la prueba de hipótesis mediante la prueba de Wilcoxon.

Resultados

Los resultados en las pruebas descriptivas fueron:

La calidad de modelo software antes del experimento según la tabla 1 se, observa que de 57 encuestados 3 que equivale el 5.8% manifiestan un nivel alto, 49 que equivale al 86.0% manifiestan un nivel regular y 5 que equivale al 8.8% manifiesta un nivel bajo de conocimiento de la calidad del modelo software

Tabla 1

Frecuencia de la variable calidad del modelo software antes del tratamiento.

		ANTES			
		Frecuencia	**Porcentaje**	**Porcentaje válido**	**Porcentaje acumulado**
Válidos	BAJO	5	8,8	8,8	8,8
	REGULAR	49	86,0	86,0	94,7
	ALTO	3	5,3	5,3	100,0
	Total	57	100,0	100,0	

La calidad de modelo software después del experimento según tabla 2 se, observa que de 57 encuestados 11 que equivale el 19.3% manifiestan un nivel alto, 44 que equivale al 77.2% manifiestan un nivel regular y 2 que equivale al 3.5% manifiesta un nivel bajo de conocimiento de la calidad en el modelo software.

Tabla 2

Frecuencia de la variable calidad del modelo software después del tratamiento.

		DESPUÉS			
		Frecuencia	**Porcentaje**	**Porcentaje válido**	**Porcentaje acumulado**
Válidos	BAJO	2	3,5	3,5	3,5
	REGULAR	44	77,2	77,2	80,7
	ALTO	11	19,3	19,3	100,0
	Total	57	100,0	100,0	

Los resultados en las pruebas inferenciales planteadas a partir de las hipótesis:

$H_{0:}$ No mejora la calidad del modelo software después de aplicar la transformación de modelos en el grupo control

$H_{1:}$ Mejora la calidad del modelo software después de aplicar la transformación de modelos en el grupo control

La significancia estadística obtenida es p (0.176), ver anexo 7 entonces se acepta la hipótesis nula y se concluye que no mejora la calidad del modelo software después de aplicar la transformación de modelos en el grupo control

$H_{0:}$ No mejora la calidad del modelo software después de aplicar la transformación de modelos en el grupo experimental

$H_{1:}$ Mejora la calidad del modelo software después de aplicar la transformación de modelos en el grupo experimental

La significancia estadística obtenida es p (0.009), ver anexo 7 entonces se acepta la hipótesis alterna y se concluye que mejora la calidad del modelo software después de aplicar la transformación de modelos en el grupo experimental.

Discusión

Pensamos que la tarea de la educación universitaria, debe facilitar un especial interés para que los estudiantes puedan desarrollar de manera sistémica el conocimiento y la práctica del complejo proceso de desarrollo de software.

Es por eso que Meliá (2012) en su tesis titulada: Un método de desarrollo dirigido por modelos de arquitectura web. Alicante, España: Llegó a la siguiente conclusión: Los modelos y transformaciones proporcionan una trazabilidad precisa desde el análisis hasta la implementación y permiten acelerar el proceso de desarrollo.

De acuerdo a este planteamiento se formuló la siguiente pregunta ¿En qué medida aplicar transformación de modelos mejora la calidad del modelo software? en función a este enunciado los resultados de la investigación, demuestran

Según la prueba de distribución de frecuencias aplicada a la investigación, se

observó un aumento en el nivel alto de la calidad en un 13.5%, también se observó una disminución en el nivel regular de la calidad en un 9.8%, por último se observó un aumento en el nivel bajo en un 5.3%, por lo que concluimos que existe un aumento en la calidad del modelo software al aplicar la transformación de modelos.

Según la prueba de Wilcoxon aplicada a la investigación, se observó que con un error de 0.176 (17.6%), la calidad del modelo software no mejora al aplicar la transformación de modelos en el grupo control. Del mismo modo se observó que con un error de 0.009 (0.09%) la calidad del modelo software mejora al aplicar la transformación de modelos en el grupo experimental.

Referencias

Bennet, S., McRobb, S. y Farmer, R. (2007). *Análisis y diseño orientado a objetos de sistemas (3.a ed.).* Madrid: McGraw-Hill.

Calero, C., Moraga, A. y Piattini, M. (2012). *Calidad del producto y proceso software.* España: RA-MA Editorial.

Dioses, S. (2012). *Desarrollo de la metodología de la implementación de un sistema de gestión de calidad aplicado al software de computadora.* Título de Ingeniero de Sistemas. Pontificia Universidad Católica del Perú. Perú.

Escalone, F. (2006). *Estudio comparativo de los Modelos y Estándares de Calidad del Software.* Maestría en Ingeniería de Calidad. Universidad Tecnologica Nacional. Argentina.

Meliá, S. (2012). *Un método de Desarrollo Dirigido por Modelos de Arquitectura Web.* Tesis Doctoral. Universidad de Alicante. Madrid.

Pantaleo, G. y Rinaudo, L. (2015). *Ingeniería de Software.* Buenos Aires: Alfaomega Grupo Editor Argentino.

Pons, C. G. (2010). *Desarrollo de software dirigido por modelos. Cnceptos teóricos y su aplicación práctica.* Buenos Aires: Edulp.