Pontificia Universidad Católica del Perú
ESCUELA DE POSGRADO

VALIDACIÓN DE TÉCNICAS DE ESTIMACIÓN DE ESFUERZO EN PROYECTOS DE SOFTWARE CON CICLOS DE VIDA INCREMENTAL Y PARADIGMA ORIENTADO A OBJETOS

Tesis para optar por el Grado Académico de
MAGÍSTER EN INFORMÁTICA
CON MENCIÓN EN INGENIERÍA DE SOFTWARE

AUTORA
Daniela Vanessa VILLANUEVA BENDEZÚ

ASESORES
Dr. José Antonio POW-SANG PORTILLO
Dr. Cristian RUSU

JURADO
Mg. Claudia ZAPATA DEL RÍO
Dr. Andrés MELGAR SASIETA
Dr. José Antonio POW-SANG PORTILLO
Dr. Cristian RUSU

LIMA – PERÚ
2013

INDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

El presente trabajo tiene por finalidad realizar la validación de unas técnicas para la estimación de esfuerzo de proyectos informáticos que siguen un modelo de ciclo de vida incremental y a su vez desarrollados bajo un modelo orientado a objetos presentadas en un trabajo previo (Pow-Sang J. A., 2012) y conocer su efectividad y percepción por parte de los participantes. Unos primeros experimentos fueron realizados en la Católica del Perú previamente mostrando alentadores resultados. Para validar dichos resultados, se han realizado réplicas de estos experimentos controlados esta vez con alumnos de Ingeniería Informática de la Pontificia Universidad Católica de Valparaíso. De los resultados obtenidos para ambas técnicas se pudo validar que pueden ser aplicadas en otro contexto, en este caso, de la PUCV obteniendo resultados similares a los obtenidos en experimentaciones previas.

PALABRAS CLAVE. Estimación de esfuerzo, técnicas de estimación, proyectos de software, replicación, UML.

Capítulo 1. PRESENTACIÓN DEL PROYECTO

1.1 Introducción

Una de las tareas más complicadas e importantes al iniciar un proyecto de software es la estimación de esfuerzo y costo necesarios para llevarlo a cabo. Gracias a esta estimación se podrán planificar y definir los recursos que un proyecto necesita, los plazos de entrega, costos involucrados, entre otros.

Existen diferentes técnicas para realizar dicha estimación, sin embargo, éstas no especifican su aplicación para los distintos modelos de ciclo de vida que existen para proyectos de software ni los enfoques a utilizar para su desarrollo. Por tal motivo, el Dr. José Antonio Pow-Sang, propuso en su tesis doctoral a Tupuy (Pow-Sang J. A., 2012), un conjunto de técnicas que apoya a la estimación de esfuerzo en proyectos de software.

Dentro de esta previa investigación, se partió por definir y justificar los parámetros específicos para los proyectos de software donde Tupuy sería aplicable: Modelo de ciclo de vida incremental y enfoque orientado a objetos.

Tupuy está conformado por 3 técnicas: UML_2FP, para el cálculo de puntos de función a partir de un diagrama de clases de análisis; Use Case Precedence Diagram (UCPD) (Pow-Sang, Nakasone, Moreno, & Imbert, 2008), técnica basada en diagramas UML y que se utilizará para definir la secuencia de construcción de los casos de uso; e Incremental-FP, que utilizando los resultados de las dos primeras técnicas, definirá los incrementos y el esfuerzo necesario para cada uno de ellos.

Este conjunto de técnicas han sido evaluadas por medio de experimentos con alumnos de pregrado y posgrado de la Universidad Católica del Perú mostrando resultados alentadores. Sin embargo, cuando hablamos de experimentación en Ingeniería de Software, debemos considerar la importancia de la replicación de dichos experimentos. La replicación consiste en la repetición de un experimento para verificar sus resultados (Juristo & Vegas, 2009) y adquirir más conocimiento acerca de lo estudiado.

El motivo de la investigación tiene por finalidad la validación de estas técnicas mediante la réplica de los experimentos controlados y que contará con la participación de con alumnos de la Pontificia Universidad Católica de Valparaíso (Chile). Los resultados obtenidos serán evaluados y comparados con los obtenidos en la PUCP y así presentar posibles sugerencias de mejora.

1.2 Definición del Problema

Para explicar el problema a solucionar con la presente investigación, detallaremos el contexto sobre el cuál se basa.

1.2.1 Técnicas de Estimación

A medida que avanza la tecnología, observamos que los sistemas de software son cada vez más complejos y particulares.

La estimación del esfuerzo para desarrollar software ha sido siempre un reto en los proyectos, tanto en la industria como a nivel académico. Muchas de estas estimaciones se dejan a manos de la experiencia del jefe de proyectos o miembros del equipo, llevando muchas veces a una situación de incertidumbre debido a los constantes cambios que pueden surgir durante el ciclo de desarrollo.

Esta estimación de un proyecto es un tema bastante crucial pues a partir de ello se podrán definir los recursos, plazos y costos necesarios para su desarrollo.

Existen, sin embargo, técnicas para el cálculo de este esfuerzo. Entre las más conocidas podemos nombrar Puntos de Función (Internacional Function Point User Group, 2004), Puntos de Casos de Uso (Ochodek, Nawrocki, & Kwarciak, 2011) o COCOMO II (Boehm, y otros, 2009). Sin embargo estas técnicas están dirigidas a paradigmas de programación estructurada y bajo el ciclo de desarrollo de software en cascada y, como era de esperarse, no son del todo eficaces y aplicables para otros tipos de modelos de ciclo de vida como los incrementales o iterativos, muy utilizados hoy en día incluso en las metodologías ágiles (Atkinson & Hummel, 2012).

1.2.2 Tupuy

Tupuy es un conjunto de técnicas que apoyan la estimación de esfuerzo conformada a su vez por 3 técnicas: UML$_2$FP, una técnica que permite realizar el cálculo de Puntos de función con modelos orientados a objeto; UCPD, una técnica de modelado que apoya en la priorización o definición de la secuencia de construcción de los casos de uso; e INCREMENTAL-FP (Pow-Sang J. A., 2012), una técnica que, utilizando los resultados de las técnicas previamente mencionadas, permite definir qué casos de uso se van a construir en cada incremento y estimar el esfuerzo que se requiere para desarrollar cada uno de ellos.

Este conjunto de técnicas son aplicables para proyectos de software orientado a objetos y que emplean un modelo de ciclo incremental.

1.2.3 La Experimentación en Ingeniería de Software y su validación

La experimentación ha jugado un rol importante en el avance científico pues Permite la validación de unas hipótesis relacionadas a un fenómeno. Tichy dijo que "La experimentación puede ayudar a construir una confiable base de conocimiento y así reducir la incertidumbre sobre cuáles teorías, métodos y herramientas son adecuadas" (Tichy, 1998).

Un aspecto importante de la experimentación es la replicación (Juristo & Gómez, 2012). Para crear un conocimiento sólido, es necesario que sea extensamente verificado. Esta verificación puede ser llevada a cabo a través de la replicación de un experimento para comprobar si los resultados son reproducibles. Si se obtienen los mismo resultados en diferentes replicaciones se puede llegar a la conclusión que esos resultados son confiables.

La Ingeniería de Software incluye también a la experimentación como parte de su estudio pues es necesaria. Según Basili (Basili, Shull, & Lanubile, 1999) "La Ingeniería de Software experimental es necesaria, la sabiduría común, la intuición, especulación y las pruebas de concepto no son fuentes confiables del conocimiento"

En ninguna disciplina de Ciencias o Ingeniería tiene sentido hablar de experimentos aislados. Los resultados de un solo experimento no pueden considerarse como una realidad o totalmente confiables. Es por eso que el concepto de experimento está estrechamente relacionado con la replicación del mismo. La replicación consiste en la repetición de un experimento para verificar sus resultados (Juristo & Vegas, 2009) y adquirir más conocimiento acerca de lo estudiado.

Los nuevos resultados obtenidos de la replicación son comparados con los resultados obtenidos en el experimento original (u otras replicaciones), lo cual afectará de manera positiva o negativa la credibilidad de los resultados.

Sin importar los resultados obtenidos, estos siempre agregarán mayor conocimiento acerca de un fenómeno lo cual podría tomarse como una oportunidad de nuevos estudios y mejoras.

1.2.4 El problema

La técnica Tupuy nació como una necesidad a cubrir para la estimación de esfuerzo en proyectos de software. Dicha técnica ha sido probada mediante experimentos con estudiantes de la Pontificia Universidad Católica del Perú (Pow-Sang & Jolay-Vázquez, 2006) (PUCP).

Sin embargo, es necesario validar estos resultados mediante la replicación de los experimentos realizados, cuyos nuevos resultados puedan incrementar la confiabilidad de los resultados. Las replicaciones nos permiten construir un historial con cada experimento que vamos realizando de esta manera podremos evaluar las mismas en varios ambientes distintos, para comprobar si nos

acercamos a los mismos resultados. Al final si las técnicas y usadas en esta replica funcionan en varios ambientes de desarrollo podemos estar frente a un estudio que nos permita tener una mejor práctica en la cual podremos invertir tiempo y dinero para que nos de mejores resultados.

Por tal motivo se decidió reproducir estos experimentos utilizando estudiantes de la Pontificia Universidad Católica de Valparaíso, Chile (PUCV) y poder percibir un nuevo enfoque que pueda ser contrastado con los resultados obtenidos previamente en la PUCP.

1.3 Objetivo General

El motivo de la investigación es realizar la validación de las técnicas que conforman a Tupuy: UCPD y UML$_2$FP mediante la replicación de los experimentos previos con estudiantes de la Pontificia Universidad Católica de Valparaíso y cuyos resultados deberán contrastarse con los obtenidos en experimentos previos en la Pontificia Universidad Católica del Perú.

1.4 Objetivos Específicos

- Evaluar la efectividad y percepción de la técnica UCPD (Diagrama de Precedencia de Casos de Uso) como herramienta para poder definir la secuencia de construcción de casos de uso frente a técnicas propias (AD HOC) mediante experimentos controlados.

- Comparar de los resultados obtenidos con los resultados de experimentos previos.

- Definir propuestas de mejora para las técnicas.

1.5 Justificación de Objetivos y Resultados Esperados

- De la evaluación al Diagrama de precedencia (UCPD) se pretende conocer su efectividad frente a las técnicas propias de los participantes (Ad hoc) para la definición de la secuencialidad de construcción de los casos de uso, tarea que se considera importante para el correcto desarrollo de proyectos de software (Pow-Sang, Nakasone, Moreno, & Imbert, 2008).

- De la evaluación cualitativa al Diagrama de precedencia (UCPD) se pretende conocer la percepción por parte de los participantes bajo los constructos de facilidad de uso, utilidad y la intención de uso para futuros proyectos por medio de una encuesta. Es importante conocer, una vez aplicada la técnica, cuáles son las opiniones de los participantes que puedan aportar ideas de mejora.

- De la evaluación de la técnica de conversión a ficheros y cálculo de puntos de función (UML$_2$FP) se pretende conocer su precisión frente a la técnica tradicional

- Con los resultados obtenidos de las evaluaciones y el análisis de los mismos se pretende validar las técnicas comparando con los resultados previos y presentar propuestas de mejora que apoyen a su desarrollo.

1.6 Preguntas de Investigación

A partir de la problemática definida en el punto 1.2.3, plantearemos las siguientes preguntas de investigación como a problema a resolver:

A. ¿Se podrían aplicar las técnicas que conforman a Tupuy: UCPD y UML_2FP bajo el contexto de la PUCV?

B. ¿Qué percepción se tiene sobre las técnicas UCPD y UML_2FP bajo el contexto de la PUCV?

1.7 Límites del proyecto

El proyecto abarca:

- La validación por medio de experimentos controlados de la técnica UCPD para la secuencialidad de la construcción de los casos de uso de un proyecto de software con alumnos de la PUCV de Chile.

- Evaluación cualitativa percibida de la técnica UCPD al finalizar el experimento.

- La validación por medio de experimentos controlados de la técnica UML_2FP para la contabilización de los Puntos de Función Sin Ajustar (PFA) con alumnos de la PUCV de Chile.

- El análisis estadístico de los resultados obtenidos de los experimentos utilizando la herramienta SPSS.

- La comparativa entre los resultados obtenidos en los experimentos previos con los alumnos de la PUCP con respecto a los obtenidos en los experimentos con los alumnos de la PUCV.

- Formulación de Propuestas de mejora y trabajo a futuro a partir del análisis obtenido.

1.8 Métodos y procedimientos

A continuación se detallarán los procedimientos a seguir para realizar la validación de las técnicas.

1.8.1 Evaluación de Efectividad del Diagrama de Precedencia (UCPD)

Por medio de dos casos de estudio, se evaluará la efectividad de la técnica UCPD frente a una técnica propia (Ad hoc) de los participantes para la definición de la secuencia de construcción de los casos de uso en un proyecto de software.

1.8.2 Evaluación Cualitativa del Diagrama de Precedencia (UCPD)

Se tomará una encuesta (cuestionario) para capturar la información sobre la percepción de la técnica, teniendo en cuenta los constructos (Moody, 2001) (Abrahão, Insfran, Carsí, & Genero, 2011) de FACILIDAD DE USO PERCIBIDA, UTILIDAD PERCIBIDA e INTENCIÓN DE USO PERCIBIDA.

1.8.3 Evaluación de Efectividad de la técnica UML_2FP para el cálculo de Ficheros

Por medio de casos de estudio, se evaluará la efectividad de la técnica UML_2FP para el cálculo de ficheros y Puntos de Función sin Ajustar (PFA).

1.8.4 Análisis estadístico de resultados

Con el fin de obtener los análisis respectivos, se realizarán pruebas estadísticas utilizando la herramienta SPSS (IBM Corporation), un software utilizado para el análisis predictivo perteneciente a la compañía IBM.

Capítulo 2. Marco Conceptual

2.1 Modelos de Ciclo de Vida del Software

Cuando hablamos de un Proyecto, podemos definirlo como "un esfuerzo temporal llevado a cabo para crear un producto, servicio o resultado único" (Project Management Institute, 2013).

Para facilitar la gestión de un proyecto de cualquier índole, este se divide en fases que a su vez forman lo que se denomina "Ciclo de Vida del Proyecto" (ISO, 1998).

Existen distintos modelos de ciclo de vida de proyectos que pueden aplicarse. Entre los más conocidos y utilizados podemos mencionar: Modelo en Cascada (Royce, 1987), cuyas fases siguen un orden secuencial y en donde una de ellas empieza cuando su predecesora se completa; El Modelo Iterativo (Pfleeger, 2002), que consiste en entregar una primera versión completa del sistema en la primera iteración y que se irá mejorando en las iteraciones posteriores; y el Modelo Incremental (ISO, 1998), en dónde se inicia con un conjunto de requisitos a cubrir y mediante una secuencia de fases, éstas van añadiendo nuevas funcionalidades al software hasta completar el producto final.

2.2 Planificación y Estimación en Proyectos de Software

Cualquiera que fuera el modelo de ciclo de vida que se elija para la gestión de un proyecto de software, debe contar con actividades de planificación y estimación para poder llevarse a cabo. Entre las más generales podríamos nombrar:

- Definir el alcance del proyecto y los requerimientos
- Planificar y secuenciar las actividades a desarrollar.
- Planificar los recursos necesarios para llevar a cabo las actividades (humanos, tecnológicos, de infraestructura)
- Planificar la duración de las actividades
- Planificar los Costos

Las estimaciones en las etapas iniciales para el desarrollo de un proyecto, son esenciales para la planificación de recursos, tiempo y costos.

Cuando se realiza la gestión y planificación de proyectos lo primero que se realiza es la estimación del tamaño del software para luego proceder a estimar el esfuerzo necesario, y los posibles cronogramas. La estimación de costos es una de las primeras actividades que se realiza en el proyecto, la mayoría de las veces después del establecimiento de los requerimientos, sin embargo se sigue aplicando

con regularidad a lo largo de la ejecución del proyecto con el fin de ajustar la precisión en la estimación previa.

No existe un modelo universal de estimación que se adapte a cualquier tipo de proyecto, por lo cual, actualmente se dispone de varias técnicas para la estimación del esfuerzo necesario. Éstas técnicas pueden clasificarse en 3 categorías (Shepperd, Schofield, & Kitchenham, 1996).

- **Juicio de Expertos.**

  Estas técnicas realizan estimaciones de acuerdo a la opinión de expertos basados en su experiencia en proyectos de similar índole. Muchas empresas consultoras aplican esta técnica debido a la experiencia que tienen en el mercado del desarrollo de software. Sin embargo, como es de esperarse, tiene la desventaja de que cuánto más novedoso sea el proyecto, las estimaciones podrían ser más imprecisas.

- **Modelos algorítmicos**.

  Son los modelos más populares, e intentan representar la relación entre el esfuerzo necesario para llevar a cabo un proyecto y una o más características del mismo. Estos modelos deben partir, principalmente, con el cálculo del tamaño del software y deberán ser ajustados a las circunstancias. Entre las más conocidas existen Puntos de Función (Internacional Function Point User Group, 2004) y COCOMO II.

- **Aprendizaje de Máquina**

  Estas técnicas han sido usadas en años recientes como complemento o alternativa al juicio experto y modelos algorítmicos. Entre estos aparecen las redes neuronales y la lógica difusa.

2.3 Conceptos sobre los que se basa Tupuy

Para poder comprender la técnica Tupuy, debemos definir algunos conceptos sobre los que se basa su desarrollo: la especificación de requisitos de casos de uso, los diagrama de clases de análisis y la estimación del esfuerzo con COCOMO II.

2.3.1 Especificación de Requisitos de Casos de Uso

Los casos de uso, que son una herramienta para comprender el funcionamiento de un sistema y su interacción con quienes lo utilizan.

Fueron introducidos por Jacobson como parte esencial de la Ingeniería Orientada a Objetos (Jacobson, 1992). A pesar de los años transcurridos desde su primera aparición, su empleo sigue vigente, tanto en nivel académico como profesional, sobre todo cuando se utiliza el modelado basado en UML (Unified

Modeling Language), convirtiéndose en, prácticamente un estándar para explicar el comportamiento de un sistema.

Los casos de uso representan una secuencia de interacciones entre un sistema y alguien o algo que usa alguno de sus servicios. Un caso de uso es iniciado por un actor.

2.3.1.1 Actor y los casos de uso.

Representa a una entidad que puede ser una agrupación de personas, sistemas o máquinas que van a interactuar con el sistema.

Según la funcionalidad del sistema, los actores cumplen el papel de un determinado rol más no de una persona en sí. Por ejemplo, en un sistema de ventas, un actor sería el cliente o el vendedor dependiendo de quién interactúa directamente con el sistema. La forma gráfica de representar un actor y un caso de uso se observa en la Figura 2.1.

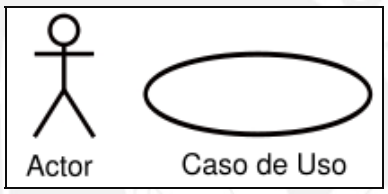


Figura 2.1 Representación gráfica de un Actor y un Caso de Uso

2.3.1.2 Relación entre actor y casos de uso

Un escenario muestra la secuencia de pasos que se produce cuando un actor interactúa con el sistema en una situación específica y un tiempo determinado. Un escenario representaría una "instancia" de un caso de uso, eso quiere decir que todos los escenarios similares se agrupan en un solo caso de uso.

Un ejemplo de escenario para el caso de uso "Registrar Ventas" de un sistema de ventas podría ser: "El vendedor ingresa los datos de los productos que un cliente desea comprar en el sistema. El sistema le muestra la pantalla de 'Detalle de Venta' donde ingresará cada producto de manera individual' y al terminar seleccionará 'Guardar' para salvaguardar la transacción. Otro escenario sería si el vendedor deseara ingresar un descuento sobre la compra realiza. Otro escenario sería si el vendedor quisiera corregir algún detalle ingresado.

En el ejemplo dado, podemos identificar los 2 tipos de escenario existentes: Los escenarios primarios y los secundarios. Según el ejemplo, el escenario primario sería el primero, donde el registro de la venta sigue un flujo básico sin errores ni excepciones. Los escenarios siguientes, representarían escenarios secundarios, donde la secuencia sale del flujo básico a uno excepcional o alternativo.

Para representar gráficamente la relación entre un actor y un caso de uso, según la notación UML, se deberá trazar una línea que los una que se denomina “Asociación de comunicación”. Utilizando el ejemplo anterior, para un sistema de ventas, podemos tener el caso de uso “Registrar venta” con un actor llamado “vendedor”. La línea de asociación representará la relación que tiene el actor con el caso de uso con el cual va a interactuar (Figura 2.2).

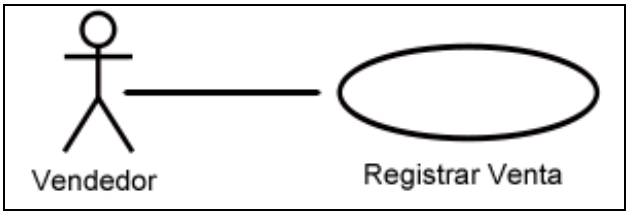


Figura 2.2 Ejemplo de Relación Actor-Caso de Uso

2.3.1.3 Documentación de los casos de uso

Una vez que se han identificado los casos de uso y sus actores, es necesario detallar la interacción que estos actores tienen con cada caso de uso. Esto se realiza mediante la especificación y documentación de caso de uso.

Según RUP (Rational Unified Process), cada especificación debe explicar lo siguiente:

- **Pre-condición:** Consiste en especificar las condiciones en las cuales debe estar el sistema para que el caso de uso pueda empezar. Por ejemplo, utilizando nuevamente el ejemplo anterior, en el caso de uso “Registrar Venta” una pre-condición podría ser que el actor debe haber ingresado al sistema correctamente y otra pre-condición podría ser que existan artículos previamente registrados en el sistema con sus respectivos costos.

- **Flujo Básico:** Muestra una secuencia de pasos donde se describirá el orden en el que se lleva a cabo un caso de uso en un escenario primario.

- **Flujo Alternativo:** Muestra una secuencia de pasos de aquellos escenarios secundarios que se producen por situación excepcionales que se encuentran fuera del flujo básico.

- **Post-condición:** Consiste en detallar el estado en el cual deberá encontrarse el sistema al finalizar el caso de uso. Por ejemplo, en el caso de uso “Registrar Venta”, una post-condición sería que la venta sea registrada satisfactoriamente en la base de datos del establecimiento.

2.3.1.4 Relaciones entre Casos de Uso

Según la notación UML, existen tres tipos de relaciones entre casos de uso:

- **Inclusión (Include):** En muchos casos, sucede que una misma funcionalidad del sistema puede ser necesitada a partir de otros casos de uso. Por ejemplo, en un sistema de ventas, la funcionalidad que permite buscar un producto puede ser utilizada en el caso de uso "Registro de Ventas", "Consultar Stock" o algún otro caso de uso relacionado a reportes. Como no se desea repetir esta funcionalidad dentro de cada caso de uso que la utiliza, entonces se crea un nuevo caso de uso llamado "Consultar Producto" que la contenga y que se relacionará con aquellos casos de uso que la utilicen. Este tipo de relaciones se llama "Inclusión" y se representa por una línea punteada desde el caso que 'usa a' al caso que es 'usado' con la etiqueta "<<include>>" (Figura 2.3)

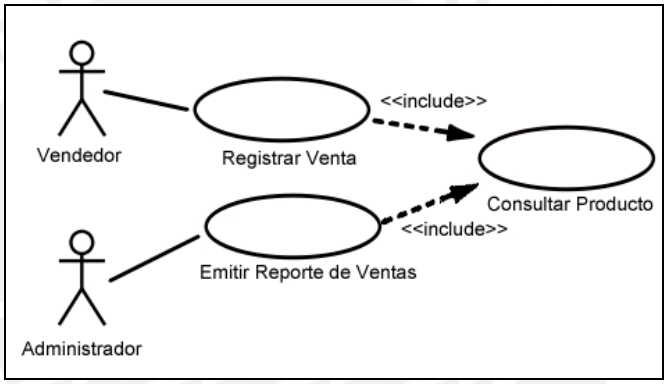


Figura 2.3 Ejemplo de relación "Include".

- **Extensión (Extend):** Una relación de tipo "extendida" ocurre cuando, dentro de un caso de uso, se presenta una funcionalidad excepcional. Entonces, se creará un nuevo caso de uso que extenderá al original. Un ejemplo sería en un caso de uso de "Registrar Venta" el cliente podría solicitarle al vendedor que le diga cuáles fueron los productos que más ha estado adquiriendo en el último mes para evaluar si debe adquirir más de ellos. Entonces podría nacer un caso de uso "Consultar historial de Ventas de cliente" donde el vendedor interrumpirá momentáneamente el caso de uso de "Registrar Venta" para pasar a este nuevo caso de uso, darle la información requerida al cliente y continuar nuevamente con el caso de uso original (Figura 2.4).

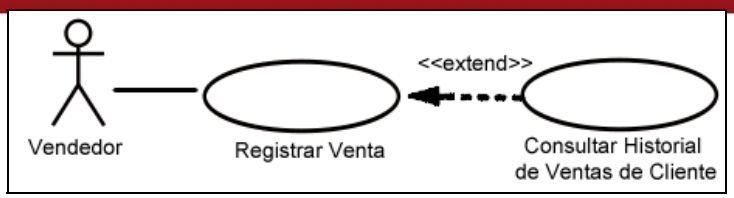


Figura 2.4 Ejemplo de relación “Extend”.

- **Generalización (Generalization):** Se identifica cuando existen casos de uso cuyo propósito es similar y contienen secuencias de acciones parecidas. En ese caso, se crea un caso de uso genérico al cual se le denomina caso de uso padre, del cual heredan dos o más casos de uso. La relación generalization entre casos de uso es análoga a la relación de “herencia” entre clases Figura 2.5.

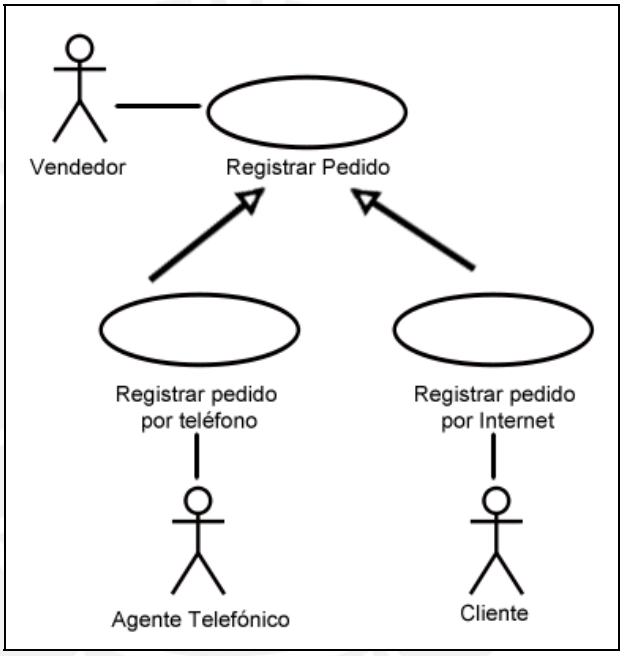


Figura 2.5 Ejemplo de relación "Generalization".

2.3.2 Análisis con Diagramas de Clase

El análisis utilizando diagramas de clase es considerado casi un estándar para entender el funcionamiento de un sistema. A continuación explicaremos sus elementos.

2.3.2.1 Objetos y Clases

Un objeto es la representación de cualquier cosa que existe en la realidad, desde personas, máquinas, vehículos, animales, etc. Cada objeto posee características, que en el Modelo Orientado a Objetos en la Ingeniería de Software (Jacobson, 1992), son denominados “atributos” que pueden ser el nombre, el color, la edad, etc.

Todos los objetos en común se pueden representar bajo un solo elemento, llamado clase, que contendrá sus atributos en común. Por lo tanto, cada objeto sería una instancia de la clase de la que forma parte. Dando un ejemplo podemos tener la clase "Vehículo", cuyos atributos podrían ser: Tipo de vehículo (terrestre, acuático, aéreo), color, marca, número de placa, entre otros. Un objeto de la clase vehículo podría ser un automóvil, cuyo color sería "rojo", de marca "Nissan" y un número de placa "N1N-444".

Para representar una clase gráficamente, se utiliza una caja donde se identificará a cada clase por su respectivo nombre. El grado de especificación de una clase dependerá de la fase en la que nos encontremos, ya sea análisis o diseño. En el caso del análisis, bastará con identificar el nombre de la clase. Sin embargo en términos de diseño, será necesario especificar más a detalle los atributos y los métodos vinculados a cada clase que, posteriormente, se traducirán en funcionalidades del sistema (Figura 2.6).

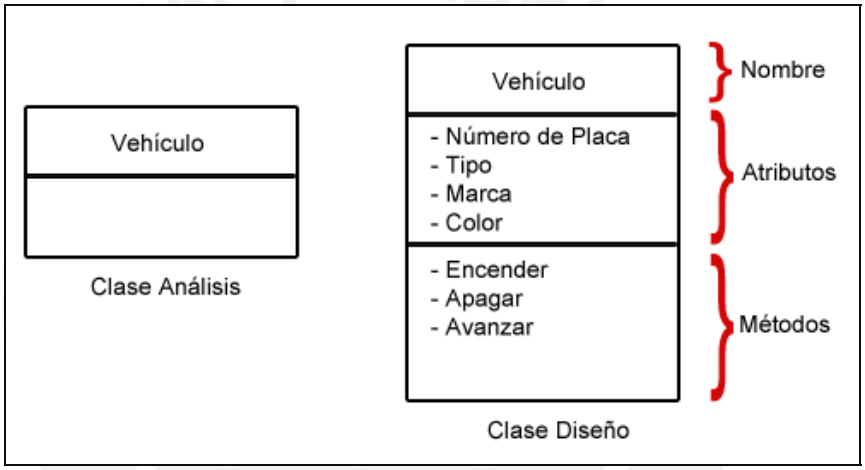


Figura 2.6 Representación gráfica de una clase.

2.3.2.2 Relaciones de asociación, agregación y composición entre clases

Las clases pueden relacionarse entre sí de diferentes formas. En la Figura 2.7 se puede apreciar gráficamente los tipos de relaciones que existen: asociación, agregación y composición.

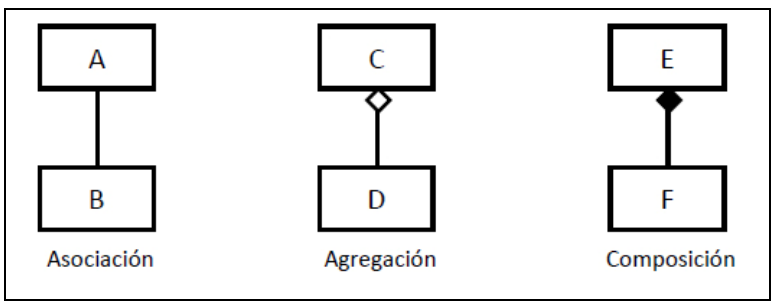


Figura 2.7 Relaciones entre clases

La relación de **asociación** se refiere a una relación de conexión entre clases. Por ejemplo, si tenemos una clase "Persona" y otra clase llamada

"Vehículo", podemos decir que una persona "posee" un vehículo asociado a ella (Figura 2.8).

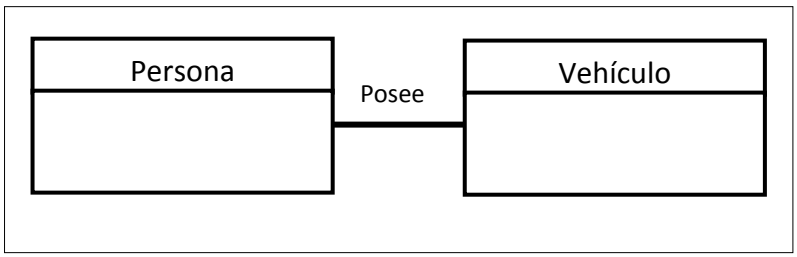


Figura 2.8 Ejemplo de relación de "Asociación".

Así mismo, en las relaciones se puede definir una **multiplicidad**, que no es más que la representación numérica de la cantidad de objetos de una clase que intervienen en la relación. En otras palabras, es el número de instancias de una clase que se relaciona a una instancia de otra clase.

Cada asociación posee 2 multiplicidades, una a cada extremo de la relación. Para especificar la multiplicidad de una relación, se debe definir la multiplicidad mínima y máxima. En la Tabla 2-1 se muestran los tipos de multiplicidades que existen.

Tabla 2-1 Multiplicidad entre relaciones de clases.

Mutiplicidad	Significado
1	Uno y sólo uno
0..1	Cero a uno
N..M	Desde N hasta M
*	Muchos
0..*	Cero o Muchos
1..*	Uno o Muchos (al menos uno)

En el ejemplo de la relación Persona-Vehículo (Figura 2.9), podemos decir que **una** persona puede tener de **cero o muchos** vehículos, pero un vehículo puede pertenecer sólo a una persona.

Figura 2.9 Ejemplo de multiplicidad

Por otro lado, las relaciones de agregación y composición se emplean cuando existe una relación parte/todo entre clases por ejemplo auto/rueda. La diferencia entre ellas radica en el tipo de dependencia entre las clases.

La relación de **composición** indica que las partes pueden existir sólo cuando el todo existe. Por dar un ejemplo, si hablamos de una Factura (todo) y su detalle (parte), si desapareciera la factura, el detalle también desaparecía. Gráficamente, la relación se muestra por una línea de conexión con un rombo sombreado al lado de la clase "todo" (Figura 2.10).

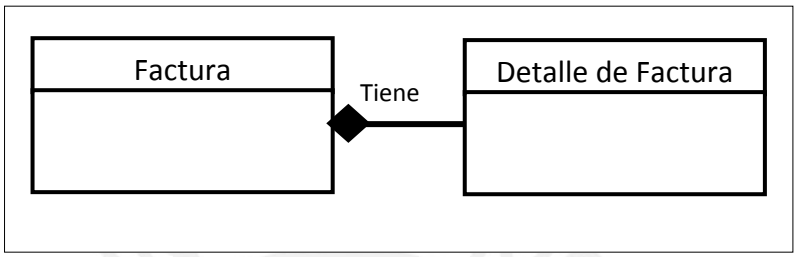


Figura 2.10 Ejemplo de relación de "Composición".

La relación de **agregación** también denota una dependencia parte/todo, pero esta dependencia no es tan fuerte. A pesar que el "todo" desaparezca, la "parte" puede seguir existiendo por sí sola. Por ejemplo, si tuviéramos la clase "Alumno" y otra clase llamada "Grupo de estudio" al cual un alumno podría pertenecer y desapareciera la entidad "Grupo de Estudio", la entidad "Alumno" puede seguir existiendo por sí misma (Figura 2.11).

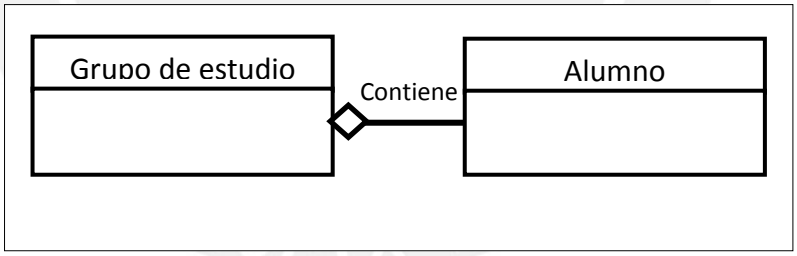


Figura 2.11 Ejemplo de relación de "Agregación".

2.3.2.3 Jerarquía de clases

Dentro de las relaciones de clases, podemos hablar de una jerarquía de clases, que consiste en tener una clase "padre" que albergará los atributos comunes de las clases hijas y estas últimas las heredarán formando nuevas clases con otros atributos particulares de cada una. En la mayoría de notaciones, la relación de herencia está representada gráficamente por una flecha desde cada clase hija hasta la clase padre.

Para explicar mejor este concepto, veremos el ejemplo de la #. Tenemos la clase padre "Vehículo" y las clases hijas "Auto", "Moto" y "Camión". Las clases hijas tienen atributos en común que heredarán de "Vehículo" como el número de placa, tipo, el color y velocidad. Sin embargo cada clase hija tendrá

atributos propios de ellas: un camión puede tener un atributo llamado “límite de carga” donde se pueda especificar cuánta carga máxima podría llevar; un auto podría tener un atributo “capote” que indicaría si es descapotable o no; una moto podría tener un atributo “maletero” para indicar si tiene o no un maletero incluido (Figura 2.12).

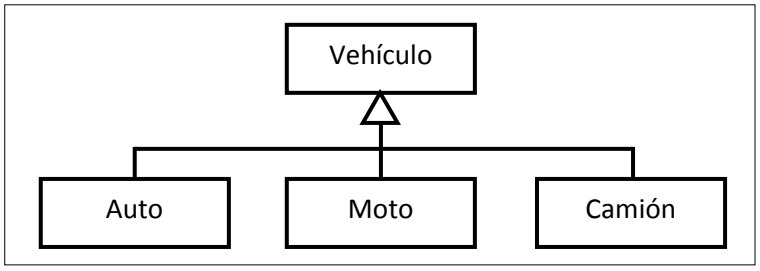


Figura 2.12 Ejemplo de herencia de clases

2.3.2.4 Clases “Asociación”

Dentro de las clases, existe un tipo de clase llamado de “Asociación”. Consiste en clases que nacen de la relación entre dos clases La clase asociación se crea cuando la asociación tiene más información como atributos adicionales.

En la Figura 2.13, “Asignación de Trabajo” es una clase asociación que pertenece a la relación de asociación de las clases Empleado y Trabajo.

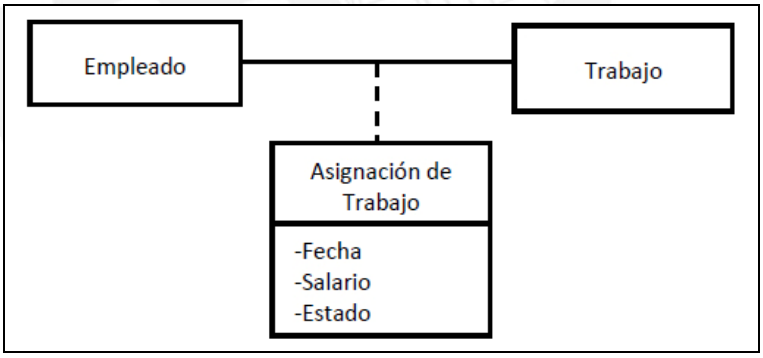


Figura 2.13 Ejemplo de clase "Asociación"

2.3.2.5 Patrones de Análisis

Una vez que se ha realizado la captura de requerimientos, se puede empezar con la fase de análisis, cuyo propósito es entender el sistema a implementar. Para ello, se realiza un diagrama de clase donde la estructura de clases del sistema se especifica, con relaciones entre clases y estructuras de herencia. Durante el análisis del sistema, el diagrama se desarrolla buscando una solución ideal. Durante el diseño, se usa el mismo diagrama, y se modifica para satisfacer los detalles de las implementaciones.

Este modelo conceptual o diagrama de clases a nivel de análisis ofrece la ventaja de concentrarnos en los objetos del dominio y no en las entidades de software. Este tipo de diagrama muestra conceptos, asociaciones entre conceptos y atributos de conceptos.

Los patrones de análisis ayudan en la especificación de requisitos y tienen el potencial de mitigar los riesgos y se ocupan en representar cómo los objetos y conceptos se relacionan en el mundo real.

2.3.3 Puntos de Función

Permiten medir el tamaño del software cuantificando la funcionalidad proporcionada al usuario basándose únicamente en un diseño lógico y en especificaciones funcionales

Esta herramienta originalmente concebía un método conocido como IFPUG – FPA pero conforme los proyectos iban masificándose aparecían distintos métodos que se adaptaban a las necesidades de los mismos. A partir de entonces los métodos de medición en puntos función se han diversificado y se han intentado adaptar a distintas necesidades del mercado y a solucionar distintos problemas que se presentaban para los primeros métodos definidos. Por ejemplo, COSMIC-FFP se ha adaptado para medir con mayor precisión aplicaciones en tiempo real.

2.3.4 Estimación de Esfuerzo con COCOMO II

El modelo original COCOMO se publicó por primera vez en 1981 por Barry Boehm (Boehm, y otros, 2009) y reflejaba las prácticas en desarrollo de software de aquel momento. En la década y media siguiente las técnicas de desarrollo software cambiaron drásticamente. Estos cambios incluyen el gasto de tanto esfuerzo en diseñar y gestionar el proceso de desarrollo software como en la creación del producto software, un giro total desde los mainframe que trabajan con procesos batch nocturnos hacia los sistemas en tiempo real y un énfasis creciente en la reutilización de software ya existente y en la construcción de nuevos sistemas que utilizan componentes software a medida.

COCOMO II es un modelo que permite estimar el coste, esfuerzo y tiempo cuando se planifica una nueva actividad de desarrollo software. Está asociado a los ciclos de vida modernos.

La principal diferencia entre los dos modelos es que COCOMO I supone que el modelo de desarrollo que se utiliza es en Cascada y utilizando lenguajes imperativos como C, Pascal, etc. y COCOMO II tiene en cuenta el modelo de desarrollo en espiral (prototipos) y para ello define varios niveles que permiten obtener estimaciones detalladas de forma incremental.

2.4 La experimentación en la Ingeniería de Software

Los grandes proyectos de software siguen un proceso con escenarios similares (requerimientos, diseño, desarrollo, etc.) Sin embargo las herramientas y la manera en que estos escenarios son realizados difieren bastante, aun así sin importar cuál sea el proceso una de las debilidades de la investigación en ingeniería de software es informar de las decisiones de estos procesos, y sus costos y beneficios son muy poco entendidos, a diferencia de otras ciencias (Perry, Porter, & Votta, 2000).

Los estudios empíricos son la clave para poder obtener esta información y realizar una mejor toma de decisiones. El estudio empírico tiene muchas formas se realiza con como experimentos, casos de estudio pero en general tratan de mostrar una comparación entre lo teórico y lo real para mejorar lo primero como resultado, estos estudios empíricos involucran los siguientes pasos: formular una hipótesis, observar la situación, abstraer la información en datos, analizar los datos y sacar conclusiones con respecto a la hipótesis.

Los beneficios de realizar bien un estudio empírico serian: El conocimiento hace que se codifique más rápido, las malas ideas o las que no aportan se descartan rápidamente, las buenas prácticas e ideas se reconocen y valorizan correctamente, los problemas importantes son considerados.

2.4.1 El estado de la investigación empírica

La Investigación Empírica se puede definir como la investigación basada en la observación para descubrir algo desconocido o probar una hipótesis. La Investigación Empírica se basa en la acumulación de datos que posteriormente se analizan para determinar su significado.

El diseño de una investigación incluye la totalidad de las etapas encaminadas a dar respuesta a las cuestiones principales en el marco de la investigación empírica.

Dentro de la ingeniería de software también se realiza investigación empírica (Juristo & Moreno, 2010). Las metodologías que puedan proponerse, la prueba de un software, las pruebas sobre una implementación, entre otros, deben someterse a prueba para poder verificar su eficiencia, rapidez, calidad, seguridad, entre otras cosas. Para ello es necesario obtener datos que nos puedan dar esta información, y estos datos serán obtenidos gracias a la experimentación.

2.4.1.1 Fortalezas Actuales

La ingeniería de software empírica está ganando reconocimiento en varias áreas y estudios en los últimos años.

2.4.1.2 Problemas Sistemáticos

Los estudios empíricos no solamente pueden ser usados para dar un valor retrospectivo sino también uno proactivo que apoye directamente la investigación

Uno de los problemas fundamentales de los estudios empíricos es que se dedican a crear datos pero no a utilizarlos, no entender porque la data es así, los datos deben responder preguntas no solamente llenar gráficos. Un aspecto fundamental de este problema es que muchos estudios empíricos carecen de hipótesis, si no poseen preguntas no tienen un fin definido.

2.4.2 Futuros Retos para los Estudios Empíricos

Cuanto más avance la tecnología, más compleja será la labor realizada en los estudios empíricos. Los investigadores deberán asumir nuevos retos.

2.4.2.1 Creando mejores estudios empíricos

Crear mejores estudios significa realizar estudios que dirijan mejor nuestra investigación lo que implica que debemos tener nuestros objetivos claros, diseñarlos con más efectividad y maximizar la información que obtenemos de ellos.

2.4.2.2 Interpretaciones creíbles

La credibilidad se refiere a la confianza que tenemos en sus conclusiones, para ello debemos considerar los siguientes problemas: Si tratamos de establecer la existencia de una relación casual, necesitamos también experimentos con alta validez es decir establecer conclusiones creíbles. Se debe evitar la tratar de medir todo con la mejor precisión. Algunas veces es necesario tan solamente acertar en algunos puntos, dependiendo del proyecto. Además necesitamos hacer públicos nuestros métodos para que otros puedan entenderlos, analizarlos y si es posible replicarlos.

2.4.2.3 Diseñando un estudio empírico

La conclusión es que ningún estudio es perfecto y el reto verdadero está en crear, diseñar y realizar un alto impacto, y estudios creíbles. Para esto debemos mejorar los siguientes puntos: tener precisión en nuestra interpretación, relevancia, impacto. Estos puntos van de la mano con la restricción de recursos y los riesgos.

2.5 Replicación de Experimentos en Ingeniería de Software

Según Forrest Shull (Shull, Carver, Vegas, & Juristo, 2008), las replicaciones experimentales se realizan para comprender mejor los fenómenos de Ingeniería de Software y mejorar las prácticas en el desarrollo del mismo. Las replicaciones abordan problemas de validación externa e interna. En el caso de la externa ayudan a los investigadores a mostrar los resultados que no dependen de

condiciones específicas del estudio original, en el caso de la interna ayudan a los investigadores a mostrar el rango de condiciones bajo los resultados experimentales.

2.5.1 Metas de la replicación en Ingeniería de Software Empírica

En general las replicaciones son útiles porque producen resultados que le permiten a la comunidad investigadora tener una percepción clara de porque los resultados fueron diferentes o no a los parámetros originales del experimento. El éxito de las replicaciones debe ser juzgado en la cantidad de información que aporta a la investigación.

Dentro de las replicaciones podemos encontrar 2 tipos: **las replicaciones exactas**, aquellas cuyos procedimientos son seguidos muy de cerca para determinar si se obtienen los mismos resultados; y por otro lado, **las replicaciones conceptuales**, aquellas donde la misma investigación o hipótesis es evaluada usando diferentes procedimientos experimentales, es decir muchas variables son alteradas.

Para los investigadores de ingeniería de software, hay dos posibles metas que tienen diferentes implicaciones para las replicaciones, las que serían:

- Probar que un resultado u observación es reproducible
- Entender las fuentes de variación que influyeron en un resultado

La meta principal de una replicación no debe estar limitada a valores hipotéticos y similares, mientras más datos se puedan introducir y controlar en una replicación y puedan generar más conocimiento a la investigación en el desarrollo de la ingeniería de software, es mejor. La meta es encontrar prácticas en diferentes ambientes que puedan tener efectos medibles para así poder tener resultados más claros en el desarrollo.

2.5.2 El rol de las réplicas independientes, disimilares

Cuando la meta de la investigación es mostrar que un efecto dado es mejor que otros, lo mejor es realizar la replicación en un estudio independiente y diferente a los valores dados en el estudio original. Este tipo de replicación está clasificada como replicación conceptual. En este caso si lo resultados del estudio original son reproducidos usando diferentes procedimientos experimentales, entonces la comunidad investigadora tendrá un alto grado de confianza de que el resultado es real, significativo e independiente. Si esta replica se realiza por personas diferentes, añadirá más confianza ya que la prueba sería más imparcial.

Además si los factores ambientales o el contexto son diferentes al estudio original, entonces los efectos no están limitados a una configuración en particular y este tipo de réplicas puede abordar validaciones tanto externas como internas.

2.5.3 El rol de las réplicas dependientes o similares

Cuando la meta de la replicación es identificar los factores que influyen en un efecto bajo condiciones diferentes, el replicador necesita tener control de las variaciones entre el estudio original y las réplicas. En este caso una réplica que tenga un menor número de factores de variación tiene una mejor oportunidad de proveer la información que se está buscando. Este tipo de réplicas son dependientes porque los replicadores confían en los diseños del estudio original como base para diseñar las réplicas. Los tipos de cambios que se realizan afectan al contexto en el que se realiza el estudio y no a los procedimientos. Estas replicaciones dependientes son del tipo replicaciones exactas descritos anteriormente.

2.5.4 El rol de los Paquetes de Laboratorio (Lab Packages)

Fuera de las metas de las replicaciones, una documentación exhaustiva del experimento original es una fuente valiosa datos. A menudo los investigadores encuentran que los journal o papers limitan datos relevantes y son excluidos de los reportes. Para esto se usan los llamados Paquetes de Laboratorio, la existencia de estos no implica que una replicación sea buena, simplemente añaden información detallada de los estudios originales los cuales se pueden usar en las replicaciones posteriores para así poder construir un historial. Tanto en las replicaciones dependientes, como en las independientes estos paquetes de laboratorio son una fuente de información valiosa.

Además los paquetes de laboratorio son útiles para aquellos investigadores que desean introducirse dentro del campo de ingeniería de software empírica o por investigadores que no tienen por costumbre ser empíricos. La presencia de estos documentos reduce significativamente los esfuerzos en la ejecución de los experimentos.

2.5.5 Variaciones entre replicaciones

Basados en los diferentes tipos de replicación encontrados, podemos formar tres grupos:

- Replicaciones que varían un poco con respecto al experimentos original
- Replicaciones que varían pero siguen los mismos protocoles que el experimentos original
- Replicaciones que usan un protocolo distinto para verificar los resultados del experimentos original (reproducciones)

2.5.6 Propósito de las Replicaciones En Ing. de Software

Los elementos del experimento que va ser replicado varían dependiendo de los propósitos de la replicación, se identifican los siguientes elementos:

- **Experimentadores**, pueden ser las mismas personas que realizaron el experimento original, distinto o ambas.
- **Sitios**, la replicación puede realizarse en otro lugar
- **Protocolos de Experimentación**
- **Operacionalismo Construido**, representa los tratamientos primarios que seran evaluados en el experimento
- Propiedades de la población

Basado en estos elementos que pueden variar una replicación, identificamos los siguientes propósitos:

- **Control para errores de muestra**, si los elementos básicos del experimento original se mantienen sin cambios el propósito es verificar que los resultados no son "por suerte".
- **Control para experimentadores**, si diferentes experimentadores realizan la réplica entonces, se apunta a verificar que los experimentadores no influyen en los resultados.
- **Control para el sitio**, si la replicación se realiza en otro sitio, entonces apuntamos a verificar si los resultados son independientes del sitio donde se realizó.
- **Control para Resultados Artificiales**, si el protocolo cambia el propósito de la replicación es verificar que los resultados no son artificiales y que reflejan la realidad de los resultados.
- **Determinar límites para Operaciones**, una replicación apunta a determinar el rango de variación de los tratamientos primarios (variables dependientes) y las medidas (variables independientes) usando un medidor de efectos
- **Determinar límites para las propiedades de la población**, si las propiedades de la población cambian el propósito de la replicación es determinar los sujetos u objetos para los cuales los resultados de la replicación se mantienen.

2.5.7 Proceso de Replicación

Para poder explotar los resultados de una replicación se necesita un enfoque riguroso en ciertos puntos, aquí se propone un procedimiento de replicación que asegura dejar resultados incluso sin las condiciones experimentales no son reproducidas de manera similar.

Se dividirá las actividades en 3 partes

2.5.7.1 Definición y planificación de la replicación (pre replica)

- Entendiendo los parámetros originales
- Analizando los nuevos parámetros
- Haciendo los cambios necesarios al experimento

- Analizando el impacto de los cambios en los resultados de la replicación

2.5.7.2 Operación y análisis de la replicación (replica)

- Ejecutando la replicación
- Análisis de Datos

2.5.7.3 Interpretación de la replicación (post replica)

- Comparando los resultados de la replicación con los anteriores
- Generando conocimiento de los cambios necesarios
- Generando conocimientos de los cambios no intencionales

Capítulo 3. Estado del Arte

3.1 Introducción

A continuación se definirán las técnicas a evaluar y los estudios previos realizados para su validación.

3.2 Técnicas de Estimación de Esfuerzo Proyectos de ciclo de vida incremental y paradigma orientado a objetos (TUPUY)

3.2.1 Definición

Tupuy (Pow-Sang J. A., 2012), cuyo nombre proviene de la palabra quechua que significa "medir" o "pesar", es un conjunto de técnicas que apoya en la estimación y planificación basada en Puntos de función para proyectos de desarrollo de software orientados a objetos que empleen un modelo de ciclo de vida incremental y que se encuentra conformada por 3 técnicas: UML_2FP, una técnica que permite realizar el cálculo de Puntos de Función empleando modelos orientados a objetos; UCPD, basado en diagramas de UML, que permite determinar la priorización o secuencia de casos de uso para su construcción; e INCREMENTAL-FP (Balbin, Ocrospoma, Soto, & Pow-Sang, 2009), técnica basada en Puntos de Función que permite calcular el esfuerzo requerido por cada incremento para un modelo de ciclo de vida incremental.

Para empezar a aplicar Tupuy, es necesario que previamente se hayan realizado los diagramas de clase análisis en notación UML así también como las especificaciones de cada caso de uso definido.

Con el diagrama de clase de análisis y las especificaciones de casos de uso, se utilizará la técnica UML_2FP para el cálculo de los Puntos de Función Sin Ajustar (PFSA). Paralelamente, utilizando las especificaciones de casos de uso, se podrá definir la secuencia de construcción de cada uno, utilizando el Diagrama de Precedencia de Casos de Uso (UCPD). Finalmente, utilizando la técnica Incremental-FP y con los resultados obtenidos de las técnicas que lo preceden, se definirán los incrementos a construir y la determinación de qué casos de uso se desarrollará en cada uno. Además se podrá estimar el esfuerzo que se requiere para cada incremento.

En la Figura 3.1 podemos observar el modelo completo de Tupuy y de las técnicas que la conforman así como el flujo de su procedimiento. Sin embargo, debemos aclarar que dentro del alcance de la presente investigación, sólo analizaremos lo relacionado a las técnicas UCPD y UML_2FP, las cuáles fueron objeto de la validación. Las líneas punteadas en la figura, muestran el alcance descrito.

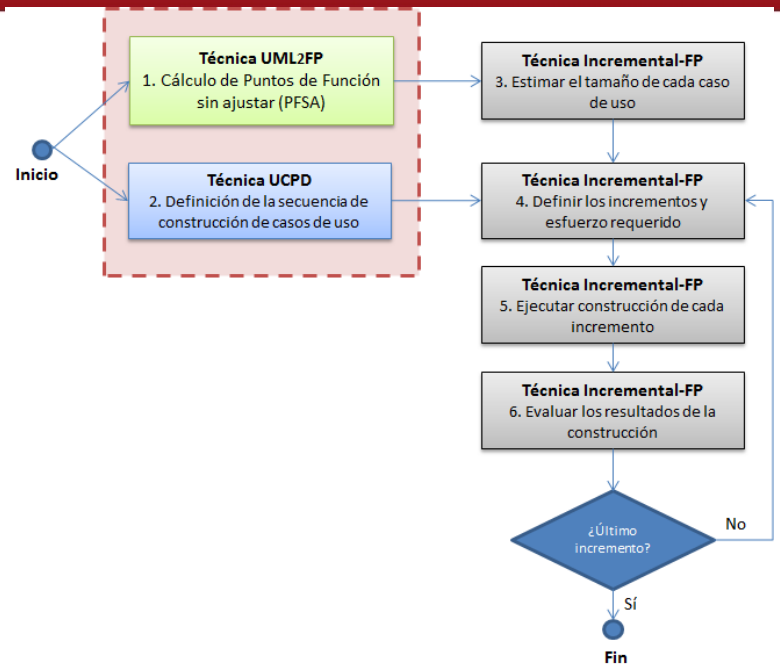


Figura 3.1 Tupuy y su procedimiento completo. **(Pow-Sang J. A., 2012)**

3.2.2 Diagrama de Precedencia de Casos de Uso (UCPD)

De acuerdo a UML y a lo explicado en el punto 2.3.1.4, las relaciones que pueden existir entre casos de uso son: “include”, “extend” y “generalization”. Adicionalmente a este estándar existe una relación llamada PRECEDENCIA y todas las relaciones serán mostradas en el diagrama de precedencia de Casos de Uso (UCPD).

El concepto de este diagrama de este diagrama fue tomado de Doug Rosenberg (Rosenberg & Stephens, 2007), quien propuso un diagrama similar, especificando las relaciones de “precede” e “invoca” para determinar los requerimientos de usuario. Aquí se muestra un ejemplo en la Figura 3.2.

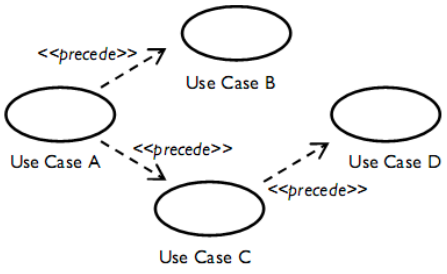


Figura 3.2 Ejemplo de Diagrama de UCPD.

(Pow-Sang, Nakasone, Moreno, & Imbert, 2008)

Para poder determinar las relaciones entre los casos de uso, se deben considerar las siguientes reglas:

3.2.2.1 Regla 1

Un caso de uso U1 precede a otro caso de uso U2 si hay una precondición que corresponde a la ejecución de un escenario en U1 que debe ser completado antes de ejecutar un escenario de U2 según la Figura 3.3.

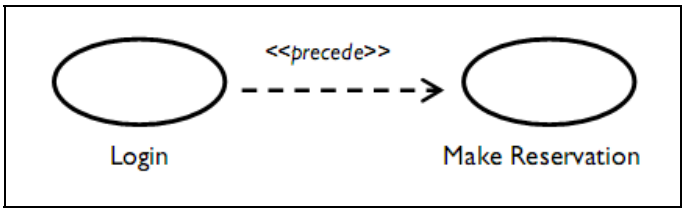


Figura 3.3 Regla 1 de Precedencia.

(Pow-Sang, Nakasone, Moreno, & Imbert, 2008)

3.2.2.2 Regla 2

Un caso de uso U1 precede a otro U2 si éste necesita información que es registrada por U1 según la Figura 3.4.

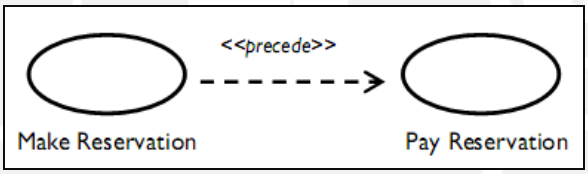


Figura 3.4 Regla 2 de Precedencia.

(Pow-Sang, Nakasone, Moreno, & Imbert, 2008)

Es importante resaltar que en el UCPD, los casos de uso tipo INCLUDE y EXTEND no son considerados ya que pueden ser parte de otros casos de uso que los refiere. Basado en este modelo, una secuencia de construcción es definida. Los casos de uso que están en el lado izquierda del diagrama deben ser implementados primero que los que están al lado derecho.

3.2.3 Técnica de Definición de Ficheros UML$_2$FP

De acuerdo a lo definido para Tupuy (Pow-Sang J. A., 2012), las reglas para calcular PF empleando modelos orientados a objetos, a la que se ha denominado en conjunto como UML$_2$FP, se han definido teniendo en cuenta dos elementos de PF: cálculo de ficheros y cálculo de transacciones.

3.2.3.1 Cálculo de Ficheros

Para determinar los ficheros y cuántos DET y RET tienen, se deben tener en cuenta las siguientes reglas.

- **Regla 1: Composición**

a) Si ocurre lo mostrado en la Figura 3.5, B no tiene una relación de asociación, agregación o composición con otra clase adicional a A, entonces contar 1 fichero por A y B con 2 RET. Se contará un DET por cada atributo de cada clase.

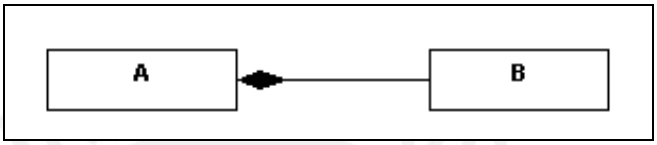


Figura 3.5 Diagrama de la regla 1a de Ficheros.

b) Si ocurre lo mostrado en la Figura 3.6 y considerando que entre B y C la multiplicidad para B es de muchos y de C es de máximo de 1 (0..1 ó 1), entonces contar 1 fichero con 2 RET por A y B. Contar un fichero con 1 RET por C. Este conteo también se aplica si B tiene una relación de agregación con C. Se contará un DET por cada atributo de cada clase y para el caso del fichero de A-B se añadirá un DET más debido a la multiplicidad de muchos con C.

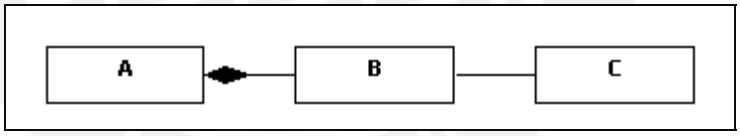


Figura 3.6 Diagrama de las reglas 1b y 1c de ficheros.

c) Si ocurre como lo mostrado en la Figura 3.6 y considerando que entre B y C la multiplicidad para C es de muchos: contar 1 fichero con 2 RET por A y B. Emplear la regla 2 (regla siguiente) para determinar si C es otro fichero o un RET del fichero conformado por A y B.

d) Si ocurre como lo mostrado en la Figura 3.7, se debe contar 1 Fichero con 3 RET por A, B y C. En el caso que hubiera otra clase más que tenga una relación de composición con C, siendo C el todo, contar un RET más. Considerar esto último para composiciones sucesivas.

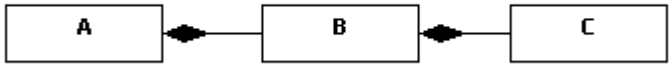


Figura 3.7 Diagrama de la regla 1d de ficheros.

- **Regla 2: Asociación y Agregación**

  En el caso que dos clases A y B estén relacionadas mediante asociación o agregación, seguir las reglas de la Tabla 3-1.

Tabla 3-1 Reglas de Asociación y Agregación para ficheros. **(Pow-Sang J. A., 2012)**

Multiplicidad A	Multiplicidad B	Condición	La identificación sería
0..1	0..*	Son independientes	2 Ficheros y cada uno con un RET
1	1..*	Si B es independiente de A	2 Ficheros y cada uno con un RET
		Si B es dependiente de A	1 fichero, 2 RET
1	0..*	Si B es independiente de A	2 Ficheros y cada uno con un RET
		Si B es dependiente de A	1 fichero, 2 RET
0..1	1..*	Si A es independiente de B	2 Ficheros y cada uno con un RET
		Si A es dependiente de B	1 fichero, 2 RET
0..1	0..1	Son independientes	2 Ficheros y cada uno con un RET
1	1	Son dependientes	1 fichero y 1 RET
1	0..1	Si B es independiente de A	2 Ficheros y cada uno con un RET
		Si B es dependiente de A	1 fichero, 1 ó 2 RET
0..*	0..*	Son independientes	2 Ficheros y cada uno con un RET
1..*	1..*	Si B es independiente de A	2 Ficheros y cada uno con un RET
		Si B es dependiente de A	1 fichero, 2 RET
1..*	0..*	Si B es independiente de A	2 Ficheros y cada uno con un RET
		Si B es dependiente de A	1 fichero, 2 RET

Contar un DET por cada atributo de la clase. Adicionalmente, considerar un DET por la asociación/agregación en el lado que contenga la multiplicidad de muchos (*) en la relación, si es que es independiente (no considerarla si es dependiente).

Esta regla cumple con los patrones en la que las clases tienen las relaciones de asociación o agregación.

- **Regla 3: Herencia**

  En el caso de herencia se deberá considerar lo siguiente:

- Si las clases derivadas (clases hijas) tienen atributos que los diferencian entre sí, entonces contar un fichero con un RET por cada clase hija. Contar como DET del fichero, los atributos de todas las clases que conforman la herencia.
- Si las clases derivadas tienen los mismos atributos, entonces contar un fichero con 1 RET. Contar como DET del fichero los atributos de todas las clases que conforman la herencia.

- **Regla 4: Clase Asociación**

En el caso de clases asociación puede considerarse como un fichero independiente o como un RET de una o ambas clases, dependiendo las reglas del negocio.

Si la clase asociación es un fichero independiente, contar como DET los atributos de la clase y añadir dos más debido a las relaciones con las otras clases. Si la clase asociación es dependiente, añadir como DET del fichero que se crea a los atributos de la clase asociación y agregar un RET más a ese fichero.

- **Regla 5: Consideraciones Adicionales**

Si luego de haber empleado las reglas anteriores para identificar dos o más relaciones entre clases y aún existen clases que aún no se han identificado como ficheros, entonces se deberá realizar lo que se indica a continuación.

a) Tomando como base lo mostrado en la Figura 3.8 y considerando que la relación entre A y B podría ser asociación, agregación, composición o herencia, y la relación entre A y C podría ser asociación, agregación o composición, pero no herencia; entonces:

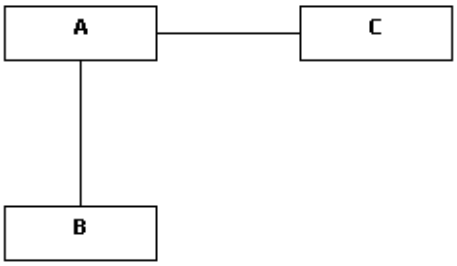


Figura 3.8 Diagrama de la regla 5 de ficheros.

Si ya se determinó que A es un fichero independiente o dependiente de B, entonces seguir las reglas 1 y 2 para determinar si C es otro fichero o forma parte del fichero en el que se incluye A. En el caso que C sea un fichero independiente y la multiplicidad de A entre la relación A-C es de muchos, entonces añadir un DET adicional al fichero que pertenece A.

b) Se aconseja que al inicio del cálculo determinar los ficheros debido a las relaciones de herencia y a las correspondiente a las clases asociación. Luego seguir las reglas anteriores, incluyendo a la 5a.

Finalmente, esta regla 5 deberá seguirse hasta que se hayan considerado todas las clases del diagrama.

- **Regla 6: Determinación de ILF o EIF y cálculo de PFSA**

Tras definir los ficheros con sus RET y DET, se tiene que determinar si es ILF o EIF. Para ello, se siguen las especificaciones definidas por el Ifpug: se considera ILF si el fichero es mantenido por el sistema y EIF si solo es consultado y es mantenido por otro sistema. En este caso, se deberán revisar las especificaciones de casos de uso para determinar si el fichero es mantenido o no por el sistema.

Luego, se deben emplear lo especificado en los manuales de PF (Internacional Function Point User Group, 2004), es decir que sabiendo qué fichero es ILF o EIF y conociendo sus DET y RET, se debe determinar la complejidad de cada uno de ellos. Para esta tarea, se debe emplear la Tabla 3-2

Tabla 3-2 Tabla para determinar la complejidad del ILF o EIF

RET/DET	1 a 19 DET	20 a 50 DET	51 o más DET
1 RET	Bajo	Bajo	Medio
2 a 5 RET	Bajo	Medio	Alto
6 a más RET	Medio	Alto	Alto

A continuación, y habiendo determinado la complejidad de cada fichero, se debe determinar el peso en PFSA de cada uno de ellos. Para ello, se debe utilizar la Tabla 3-3.

Tabla 3-3 Tabla para determinar el peso en PFSA de ILF en EIF

Complejidad de Fichero	PFSA para ILF	PFSA para EIF
Bajo	7	5
Medio	10	7
Alto	15	10

- **Regla 7: Notación gráfica para dependencia/independencia y para ficheros**

Esta regla es opcional y solo se emplearía si es que se quiere mostrar gráficamente la dependencia entre dos clases. Para ello, según UML, la notación de dependencia entre dos elementos es una flecha punteada.

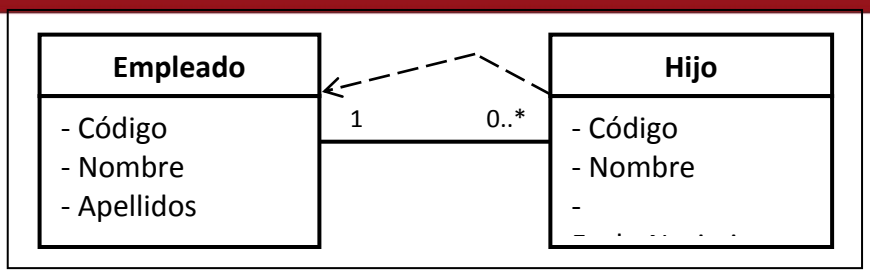


Figura 3.9 Ejemplo de Regla 7 con dependencias

En este ejemplo de la Figura 3.9, el hijo depende del Empleado; por lo tanto, adicionalmente a la asociación, se indica la relación de dependencia (flecha punteada). En el caso que la relación entre dos clases sea de composición, no es necesario indicar dependencia, ya que la composición representa una dependencia entre dos clases.

En cuanto a los ficheros, la representación más adecuada sería la de una clase a la que se le incluiría el estereotipo <<File>>, tal y como se muestra en la siguiente figura:

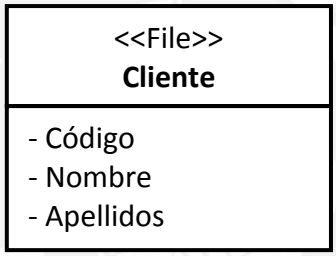


Figura 3.10 Ejemplo fichero de Puntos de función en UML

3.2.3.2 Cálculo de Transacciones

La transacción de PF, originalmente denominada como Transactional Function por la técnica, es un proceso elemental que provee funcionalidad al usuario al procesar datos. Las transacciones desempeñan los procesos de guardar, actualizar, obtener y mostrar datos.

Para poder identificar las transacciones, se deben emplear las especificaciones de casos de uso. Tras identificar las transacciones, se seguirá con lo que indican los manuales de Puntos de Función (Internacional Function Point User Group, 2004) para calcular el peso de cada una de las transacciones. Para ello, se tomará como base la parte de las especificaciones de caso de uso correspondiente a cada transacción o los prototipos de pantallas asociados a ellas. Este cálculo puede hacerse más simple con la ayuda de una herramienta integrada a un modelador de UML.

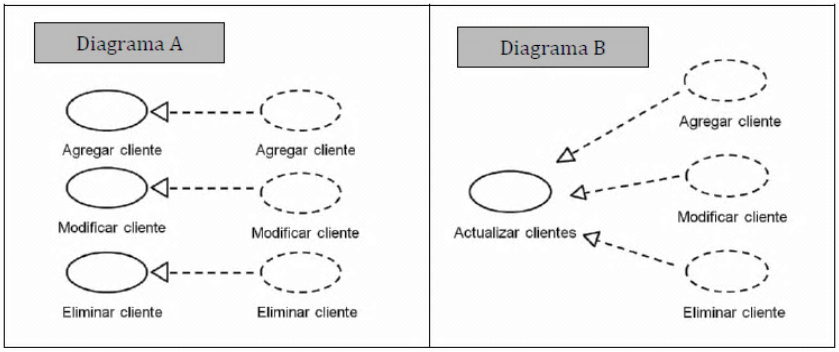


Figura 3.11 Cálculo de Transacciones.

3.2.4 Experimentos Previos Realizados con Tupuy.

TUPUY ha sido probada en proyectos con alumnos de la Pontificia Universidad Católica del Perú. A continuación explicaremos de manera resumida los experimentos realizados y sus resultados.

3.2.4.1 Evaluación previa de la técnica UML$_2$FP

La evaluación se realizó mediante experimentos controlados donde los participantes utilizaron la técnica original de Puntos de Función propuesta por la IFPUG (Internacional Function Point User Group, 2004) y la técnica propuesta UML$_2$FP para determinar la cantidad de ficheros y sus respectivos RETS. Esto permitía evaluar si UML$_2$FP era casi tan precisa como la técnica original para el cálculo de ficheros.

Se realizaron **cinco** experimentos controlados: uno con profesionales, uno con docentes y asistentes, dos con alumnos de posgrado y uno con alumnos de pregrado de la PUCP que se resumen de acuerdo a la siguiente tabla:

Tabla 3-4 Experimentos previos de técnica UML$_2$FP

N°	Aplicado a	Año	Número de Participantes
1	Profesionales	2008	16
2	Docentes y asistentes (Experimento piloto para comprobar la mejora de los instrumentos a utilizar)	2008	4
3	Estudiantes de pregrado	2009	22
4	Estudiantes de posgrado del Curso de Especialización en Ing. de Software	2009	14
5	Estudiantes de posgrado de la Maestría en Informática	2010	15
		TOTAL	**71**

Para los experimentos se formuló la siguiente pregunta de investigación:

"La precisión que se obtiene al aplicar UML$_2$FP para identificar ficheros es por lo menos similar que al aplicar la técnica definida por IFPUG?"

- **Resultados obtenidos de los experimentos**

En el **primer experimento**, se pudo comprobar estadísticamente que ambas técnicas producen los mismos resultados para los casos de las categorías composición/agregación, clase asociación y generalización. Sin embargo, para la categoría asociación, se obtienen mejores resultados con IFPUG; cuando se esperaba que se obtuvieran resultados parecidos, ya que ambas técnicas son similares. Por tal motivo se consideró que los casos presentados podrían tener algún inconveniente.

En el caso del **segundo experimento**, se trató de un experimento piloto en donde se quiso determinar si realmente había algún problema con la descripción de los casos, en especial con los empleados para la categoría de asociación. Con los resultados obtenidos, se concluyó que al caso 1 le faltaba más información.

Para el **tercer experimento**, y con las mejoras implementadas de acuerdo a lo detectado en el segundo experimento, se obtuvieron resultados favorables y se pudo afirmar con un 95% de nivel de confianza que UML$_2$FP produjo mejores resultados que la técnica del IFPUG para los casos de composición/agregación en este experimento.

El **cuarto experimento**, fue una réplica del tercer experimento, pero esta vez, contó con la participación de alumnos de posgrado. Con los resultados obtenidos se pudo afirmar con un 95% de nivel de confianza que UML$_2$FP produjo mejores resultados que la técnica del IFPUG para los casos de composición/agregación.

Finalmente, el **quinto experimento**, realizado también con alumnos de posgrado y utilizando el mismo método de los experimentos tres y cuatro, se pudo afirmar que se puede afirmar con un 95% de nivel de confianza que no hay diferencias significativas entre los resultados obtenidos con Ifpug y UML$_2$FP para todas las categorías. De acuerdo a los resultados obtenidos en los experimentos, se puede afirmar que la técnica UML$_2$FP provee por lo menos resultados igual de precisos que la técnica del IFPUG para determinar los ficheros y sus RET.

Finalmente, se presenta la Tabla 3-5 con el resumen de los resultados obtenidos de todos los experimentos para UML$_2$FP.

Tabla 3-5 Resultados obtenidos en cada experimento previo para UML_2FP

Experimento	Mejor técnica por categoría				Casos empleados
	Asociación	Composición/ Agregación	Clase Asociación	Generalización	
1	IFPUG	Igual	Igual	Igual	Inicial
2	**UML_2FP**	**UML_2FP**	Igual	Igual	Invertido al inicial
3	Igual	**UML_2FP**	Igual	Igual	Invertido al inicial mejorado
4	Igual	**UML_2FP**	Igual	Igual	Invertido al inicial mejorado
5	Igual	Igual	Igual	Igual	Inicial mejorado

Por tanto podemos concluir, que la técnica UML_2FP es igual o más precisa que la original de IFPUG en los experimentos previos realizados.

3.2.4.2 Evaluación previa de la técnica UCPD

Para la técnica UCPD, se realizaron experimentos con alumnos de pregrado, posgrado y profesionales. Se buscó evaluar la percepción y la efectividad de la técnica UCPD para priorizar los casos de uso a construir frente a las técnicas Ad hoc de los participantes.

Se tuvieron en cuenta 2 tipos de estudio: **1 Estudio cualitativo**, donde se evaluó la percepción de los participantes sobre la técnica UCPD basada en 3 constructos (Intención de Uso percibida, utilidad de uso percibida y facilidad de uso percibida); y **3 Experimentos Controlados** dónde los participantes tuvieron que emplear sus técnicas propias o Ad hoc para determinar la secuencia de construcción de casos de uso y luego UCPD. La serie de experimentos se desarrolló de la siguiente manera:

Tabla 3-6 Estudios previos de la técnica UCPD

N°	Aplicado a	Tipo de Estudio	Año	Número de Participantes
1	Alumnos de pregrado	Cualitativo	2005	31
2	Profesionales	Experimento controlado	2007	25
3	Alumnos de posgrado	Experimento controlado	2009	14
4	Alumnos de pregrado	Experimento controlado	2010	35
			TOTAL	105

- **Método y Materiales**

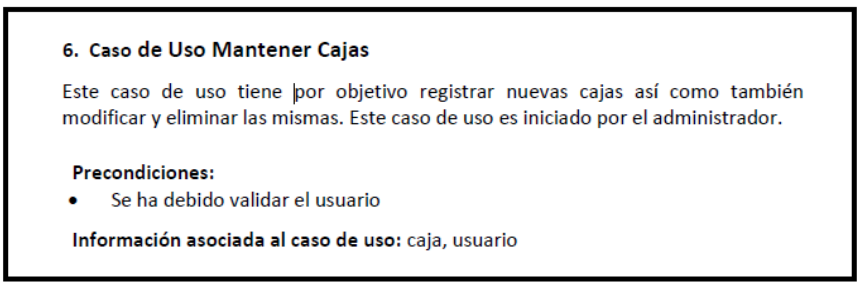
Para la **evaluación cualitativa** de la percepción se utilizó un cuestionario (encuesta) que contenía preguntas en escala de 5 puntos de Likert acerca de 3 constructos: Facilidad de uso percibida, utilidad percibida e intención de uso percibida.

Tabla 3-7 Cuestionario para la evaluación de percepción de UCPD

DIAGRAMA DE PRECEDENCIAS	-				+
1. Facilidad para construir o realizar el diagrama de precedencias	**0**	**1**	**2**	**3**	**4**
2. Utilidad del diagrama de precedencias para determinar la secuencia de construcción de software (Utilidad es el grado en el cual UCPD será efectivo para lograr los objetivos propuestos; es decir determinar la secuencia de construcción de software)	**0**	**1**	**2**	**3**	**4**
3. ¿Usaría el diagrama de precedencias para siguientes proyectos de desarrollo de software?	**0**	**1**	**2**	**3**	**4**

Este cuestionario tenía además una pregunta en donde el participante debía seleccionar el orden de construcción de los casos de uso de acuerdo a su tipo: maestra, transaccional o de reportes.

Para los **experimentos controlados**, se utilizaron 2 casos de estudio, y cuatro cuestionarios. El primer caso de estudio corresponde a un sistema de ventas en un restaurante y el segundo, un sistema de matrícula de un colegio. Cada caso de estudio contaba con un diagrama de casos de uso, descripción de cada caso de uso con sus precondiciones e información que se utiliza en cada caso de uso en términos de clases o entidades. En la **Figura 3.12** se muestra un ejemplo del contenido de la información brindada en cada caso y por cada caso de uso.



6. Caso de Uso Mantener Cajas

Este caso de uso tiene por objetivo registrar nuevas cajas así como también modificar y eliminar las mismas. Este caso de uso es iniciado por el administrador.

Precondiciones:

- Se ha debido validar el usuario

Información asociada al caso de uso: caja, usuario

Figura 3.12 **Ejemplo de Información de Casos de Uso.**

Los 3 primeros cuestionarios, contenían preguntas en las cuales los participantes debían decidir entre 2 casos de uso y responder cuál de ellos debía ser construido primero. Por ejemplo, para el primer caso de estudio, una de las preguntas fue la siguiente:

¿Construiría primero “mantener caja” antes de “abrir caja”?

a) Sí b) No c) Me es indiferente

Figura 3.13 Ejemplo de pregunta para los cuestionarios sobre la efectividad de UCPD

El cuarto cuestionario fue conocer la percepción de los participantes sobre la facilidad y utilidad de UCPD tal como se hizo en el primer estudio una vez que los participantes hubiesen utilizado la técnica.

Así mismo, la secuencia de actividades en cada experimento fue de la siguiente manera:

Tabla 3-8 Secuencia de actividades de experimentos controlado previos UCPD

Tarea	Actividad
1	Recibir el caso de estudio 1 y el cuestionario 1
2	Completar el cuestionario 1
3	Realizar la inducción a UCPD
4	Recibir el caso de estudio 2 y el cuestionario 2
5	Completar el cuestionario 2 utilizando UCPD
6	Elaborar UCPD para el caso 1
7	Completar el cuestionario 3
8	Completar el cuestionario de percepción

- **Resultados obtenidos de la evaluación de percepción**

El **primer estudio**, de tipo cualitativo, se realizó con alumnos de pregrado que habían participado en la construcción de proyectos de software usando Tupuy (Pow-Sang, Nakasone, Moreno, & Imbert, 2008) (Pow-Sang & Imbert, 2007) (Pow-Sang & Imbert, 2004).

Tabla 3-9 Resultados de Primer estudio de UCPD

Constructo	Media	Percepción
Facilidad de Uso	3.9	Fácil de Usar
Intención de uso	4.2	Existe intención de uso
Utilidad	4.4	Útil

Según la Tabla 3-9, para el constructo de Facilidad de uso percibida, se obtuvo una media de 3.9 puntos y se pudo afirmar con un 95% de nivel de confianza que UCPD es percibida como fácil de usar. Para el constructo de

Utilidad percibida, se obtuvo un valor medio de 4.4, con lo cual se pudo afirmar que existe una intención de usar UCPD en proyectos futuros. Finalmente para el constructo de Utilidad, se obtuvo una media de 4.2, lo cual permitió afirmar que UCPD es considerada como útil al momento de priorizar casos de uso para su construcción. Los resultados muestran de manera general una actitud positiva de los alumnos hacia la técnica.

- **Resultados obtenidos de los experimentos**

Los estudios posteriores, corresponde a experimentos controlados en donde los participantes aplicaron UCPD y técnicas Ad hoc para la priorización de casos de uso. Para comparar los resultados entre técnicas ad hoc y UCPD, se empleó el porcentaje de "respuestas correctas", que son aquellas respuestas en las que se considera que la secuencia más fácil de construcción de la funcionalidad de un sistema es maestros-transacciones-reportes.

En el **segundo estudio**, experimento controlado con profesionales, la mayoría de participantes seleccionaron que la secuencia que es más fácil para construir un software era maestros-transacciones-reportes, y, con los datos obtenidos con estos participantes, se pudo demostrar que UCPD produce mejores resultados que emplear las técnicas Ad hoc. También se procesaron los datos obtenidos con aquellos profesionales que seleccionaron otra secuencia de construcción y, para ellos, se pudo comprobar que UCPD sigue siendo efectiva, siempre y cuando se asuma que la secuencia más fácil para construir software es maestros-transacciones-reportes.

El **tercer estudio**, experimento controlado fue aplicado a estudiantes de posgrado. Los resultados obtenidos fueron satisfactorios, por lo que se comprueba empíricamente que UCPD provee mejores resultados que si se aplicaran las técnicas Ad hoc. Los resultados obtenidos fueron satisfactorios, por lo que se comprueba empíricamente que UCPD provee mejores resultado que si se aplicaran las técnicas Ad hoc.

Además, se realizó un **cuarto estudio**, un experimento controlado con estudiantes de pregrado. El material empleado fue el mismo que el utilizado en el tercer estudio con alumnos de posgrado. Los resultados obtenidos fueron satisfactorios, de manera similar a los obtenidos con los profesionales y los estudiantes de posgrado.

Finalmente se pudo afirmar de acuerdo a los tres experimentos controlados, que UCPD es más efectivo que las técnicas Ad hoc para la priorización de casos de uso (Tabla 3-10).

Tabla 3-10 Resumen de resultados de efectividad para los experimentos controlados de UCPD

Estudio	Aplicado a	Técnica más efectiva
2	Profesionales	UCPD
3	Alumnos de posgrado	UCPD
4	Alumnos de pregrado	UCPD

Así mismo, en la evaluación de la percepción de UCPD de estos 3 experimentos, se pudo observar, que los resultados obtenidos sobre la percepción de UCPD son muy similares a los obtenidos en el primer estudio correspondiente a la evaluación inicial con alumnos de pregrado.

3.3 Tupux

Para apoyar el uso de la técnica Tupuy se creó una herramienta desarrollada en Visual Studio .NET a modo de complemento de STAR UML (Plastic Software, Inc., 2008), herramienta de software libre para el uso de UML. Existen 4 módulos que lo conforman como se muestra en la Figura 3.14.

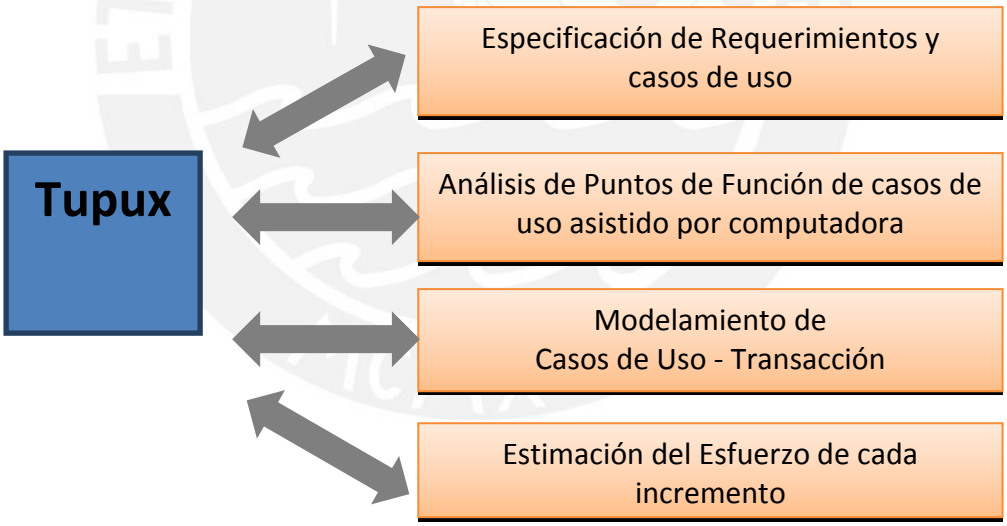


Figura 3.14 Módulos de TUPUX

3.3.1 Especificación de Requerimientos y Casos de Uso

Al inicio de cada proyecto de software se debe realizar el análisis de requerimientos. Dentro de la herramienta, esta labor es realizada en el módulo de ESPECIFICACIÓN DE REQUERIMIENTOS y CASOS DE USO. De acuerdo a (Balbin, Ocrospoma, Soto, & Pow-Sang, 2009), el documento de especificación de requerimientos de software y el diagrama de casos de uso deben ser desarrollados. La herramienta permite la integración de tanto la especificación

como el diagrama permitiendo al usuario asignar los requerimientos a sus correspondientes casos de uso.

Se incluye entonces un diagrama de requerimientos a STARUML donde se podrán establecer relaciones entre requerimientos.

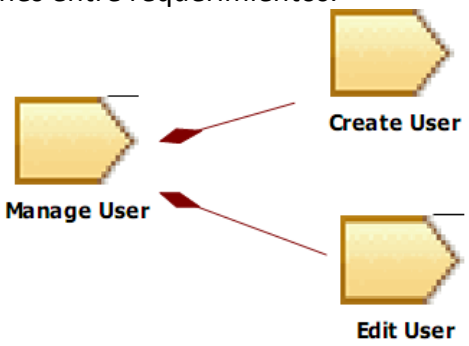


Figura 3.15 Relación entre requerimientos.

Se puede apreciar que el requerimiento de "ADMINISTRAR USUARIO" agrupa los requerimientos de "CREAR USUARIO" y "EDITAR USUARIO". Los casos de Uso también pueden estar relacionados a los requerimientos, como se muestra a continuación.

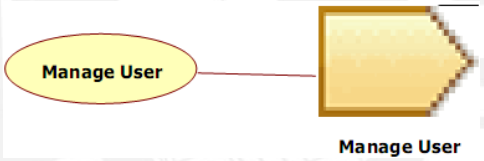


Figura 3.16 Relación entre casos de uso y requerimientos

La herramienta también permite la relación entre flujos de casos de uso y las transacciones que tiene. Con esto es posible definir que transacciones son usadas para completar los pasos de un flujo de casos de uso (Figura 3.17)

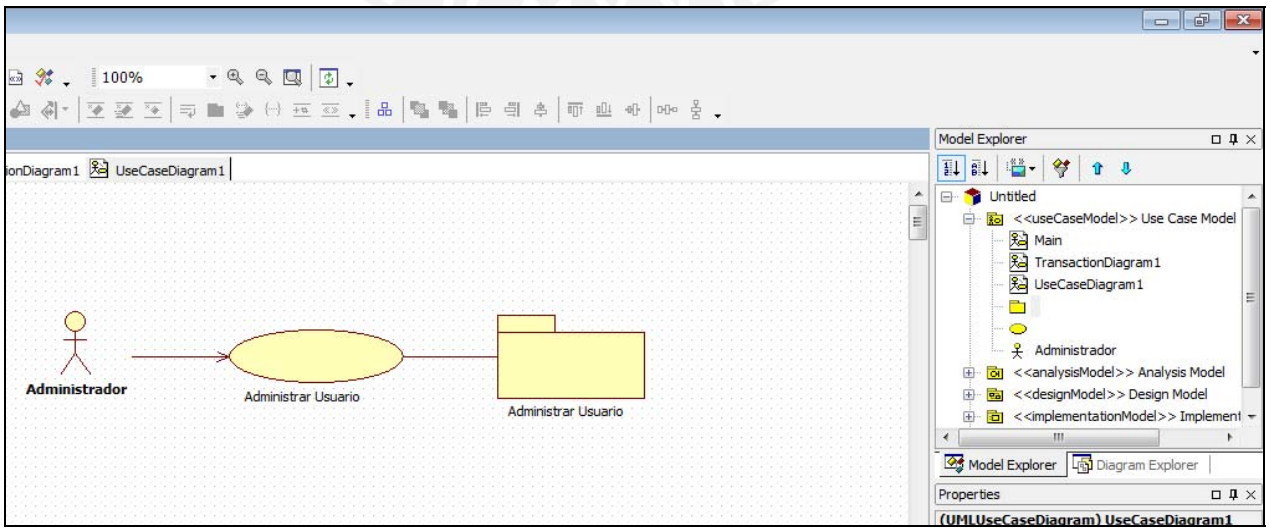


Figura 3.17 Paso 1 – Especificación de Requerimientos en TUPUX.

3.3.2 Análisis de Puntos de Función de casos de uso asistido por computadora (FPA)

Su propósito es facilitar la definición automatizada de FICHEROS (ILF o EIF) y de los RETS y DETS que los componen. Los diagramas de clase serán utilizados en este módulo pues en ellos se refleja la información el software va a usar.

La determinación de ficheros y RETS será hecho mediante el análisis de relaciones existentes entre clases. Y para dar flexibilidad a este proceso, las reglas para determinar la formación de FICHEROS, RETS y DETS son configurados en un archivo de texto. Se provee de métodos básicos de algoritmos de puntos e función y también de interfaces para personalizar los procedimientos que serán desarrollados.

3.3.2.1 Modelamiento de Casos de Uso - Transacción

Se definen las transacciones que tendrá el software. Se realiza un diagrama de transacciones. En este diagrama el usuario podrá definir relaciones entre casos de uso, transacciones y ficheros así como especificar cuál de los DETS de las transacciones usa. Aquí se muestra la relación entre una transacción y un fichero (ILF o EIF).

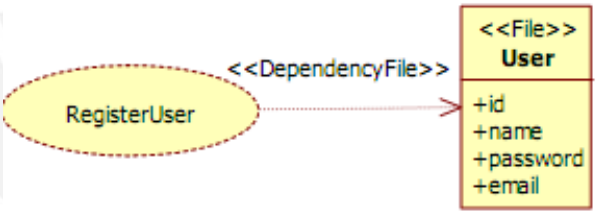


Figura 3.18 Modelo de Ficheros FPA y transacciones.

Un caso de uso puede tener 1 o más transacciones dependiendo de su especificación descrita en el punto 3.1.2. Usando este módulo se puede también especificar el tipo de fichero (ILF o EIF).

Antes de determinar el esfuerzo requerido por cada incremento, un diagrama de precedencia de Casos de Uso (UCPD) debe ser elaborado, con el cual la secuencia de construcción de los casos de uso se define.

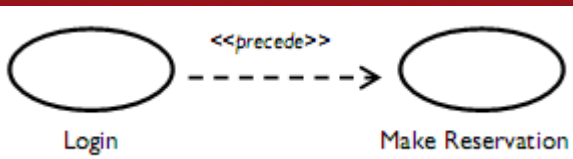


Figura 3.19 Modelo de Ficheros FPA y transacciones.

Este diagrama ayuda al desarrollador para definir visualmente el orden de construcción de los casos de uso y cuál debe ser incluido en cada incremento.

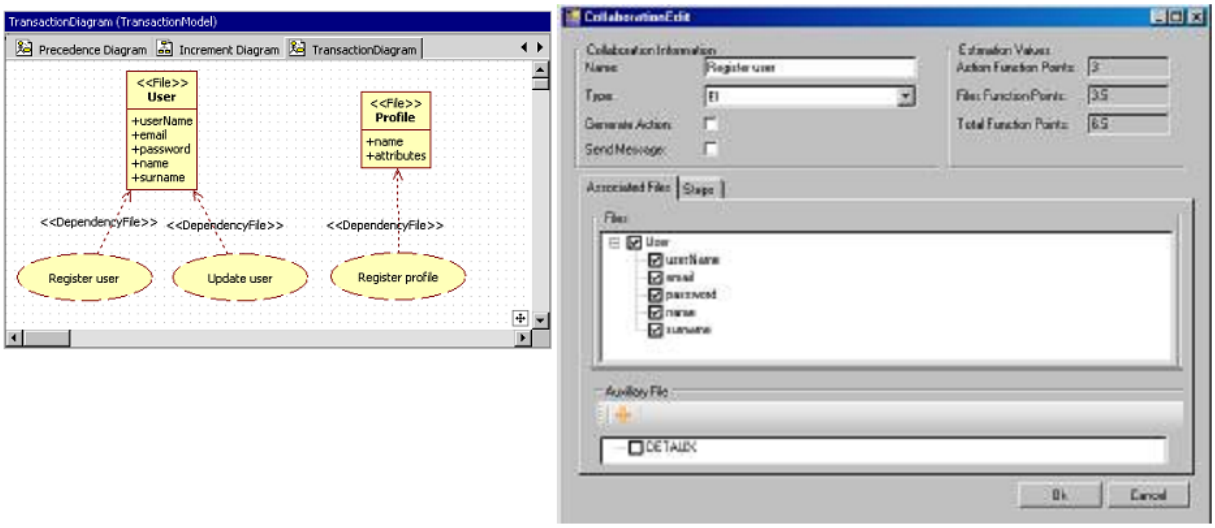


Figura 3.20 Paso 3 – Definir transacciones en TUPUX.

3.3.2.2 Estimación de esfuerzo del incremento

Este módulo permite el modelado de los incrementos del proyecto de software y los casos de uso relacionados (casos de uso que son desarrollados en cada incremento).

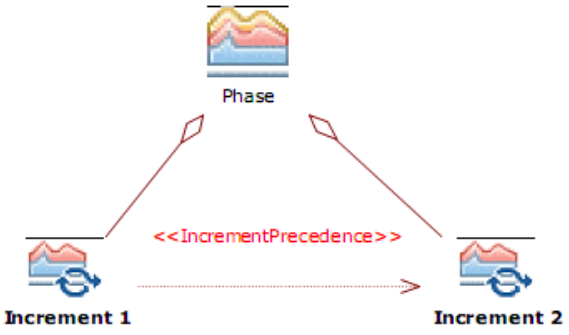


Figura 3.21 Relaciones entre una fase y 2 incrementos.

A continuación se muestra las relaciones entre los casos de uso y las transacciones. Como puede verse en la siguiente figura, dos transacciones pueden corresponder a un mismo caso de uso:

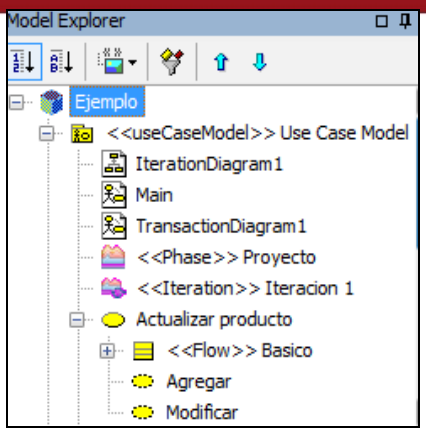


Figura 3.22 Vista de Casos de Uso y Transacciones.

Capítulo 4. Validación: Evaluación del diagrama de precedencia de casos de Uso

4.1 Introducción

En este capítulo realizaremos la validación del Diagrama de Precedencia de Casos de Uso comenzando con la evaluación de la efectividad mediante la experimentación y continuando con la evaluación cualitativa por medio de una encuesta de percepción de los participantes. Finalmente presentaremos el análisis de resultados y las conclusiones respectivas

4.2 Metodología de la evaluación

La evaluación del diagrama de precedencia de casos de Uso consta de 2 partes:

- **Evaluación de la efectividad mediante experimentos controlados:**

  Con la participación de alumnos de la PUCV se realizan una serie de experimentos que constan de la aplicación de técnicas propias (Ad hoc) de los participantes y de UCPD para definir la secuencialidad de construcción de casos de uso de 2 Casos de Estudio. Se aplica un cuestionario por cada caso de estudio. Previo a esto se realizó una inducción a la técnica UCPD.

- **Evaluación cualitativa mediante encuesta:**

  Se realiza una encuesta (cuestionario) para capturar la percepción de los participantes al usar la técnica UCPD. Para ello se tienen en cuenta 3 constructos a ser evaluados: FACILIDAD DE USO PERCIBIDA, UTILIDAD PERCIBIDA e INTENCIÓN DE USO PERCIBIDA.

El esquema definido para la evaluación se encuentra ilustrado en la Figura 4.1.

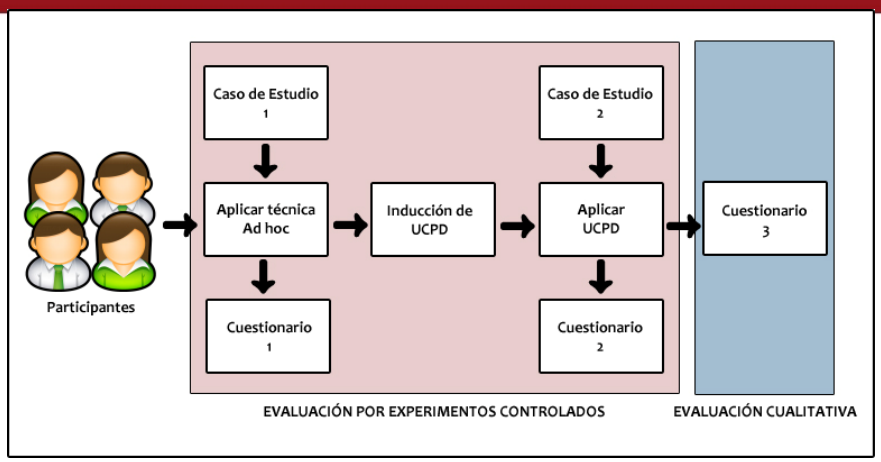


Figura 4.1 Esquema de la Metodología de Evaluación del Diagrama de Precedencia de Casos de Uso.

Así mismo, se puede mostrar la secuencialidad de las tareas realizadas mediante la Tabla 4-1.

Tabla 4-1 Tareas a realizar para Evaluar la técnica UCPD

Tarea	Descripción
1	**Recibir Caso de Estudio 1: VENTAS y Cuestionario 1** Los participantes recibirán un caso de estudio y un cuestionario sobre la priorización de los casos de uso que se describen en dicho caso de estudio
2	**Completar Cuestionario 1** Los participantes deberán completar el cuestionario, definiendo la secuencialidad de los casos de uso de acuerdo a la información proporcionada en el caso de estudio.
3	**Inducción a UCPD** Una vez que todos los participantes finalizan la resolución del Cuestionario 1, se procede a darles una inducción de la técnica UCPD.
4	**Recibir Caso de Estudio 2: COLEGIO y Cuestionario 2** Al finalizar la inducción, cada participante recibirá un segundo caso de estudio con su respectivo cuestionario. Se les pedirá ahora que apliquen la técnica UCPD para responder al cuestionario.
5	**Completar Cuestionario 2** Los participantes responderán el cuestionario 2 aplicando la técnica UCPD.
6	**Elaborar UCPD del Caso 2** Utilizando el mismo caso de estudio y lo aprendido, deberán elaborar un Diagrama de Precedencia de Casos de Uso
7	**Completar Cuestionario 3** Finalmente, se les entregará un tercer cuestionario (encuesta) sobre la percepción que tuvieron acerca de la técnica aprendida.

Se realizaron 3 experimentos a 3 grupos de estudiantes de pregrado de la carrera de Informática en diferentes niveles de la PUCV (Tabla 4-2).

Tabla 4-2 Lista de Experimentos realizados para la Evaluación de UCPD.

N° Exp	N° alumnos	FECHA	SEMESTRE	CURSO	CARRERA
EXP01	13	07-09-2012	10° de 12 semestres	Ing. de Software	Ing. Civil Informática
EXP02	31	13-09-2012	6° de 8 semestres	Ing. de Software	Ing. de Ejecución en Informática
EXP03	21	25-09-2012	7° y 8° de 8 semestres	Ing. de la Usabilidad	Ing. de Ejecución en Informática

4.3 Evaluación de la Efectividad del Diagrama de precedencia de Casos de Uso mediante experimentos controlados

4.3.1 Objetivo

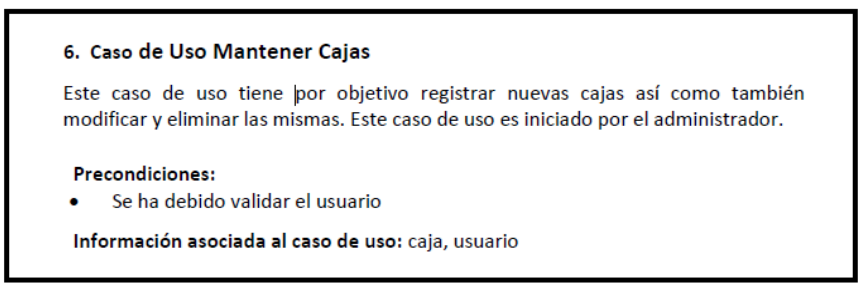
Medir la efectividad de la técnica UCPD para definir la secuencia de construcción de casos de uso desde el punto de vista del desarrollador frente a las técnicas propias empleadas (Ad hoc). Así mismo, se busca contrastar dichos resultados con los obtenidos en los experimentos en la PUCV.

4.3.2 Diseño e instrumentos utilizados para la evaluación

Los materiales empleados en el experimento fueron **dos casos de estudio** y **dos cuestionarios**, los mismos utilizados en la evaluación previa hecha en la PUCP (Punto 3.2.4.2).

El primer caso de estudio corresponde a la elaboración de un sistema para un restaurante y el segundo, a un sistema de matrícula para un colegio de educación secundaria. Ambos casos de estudio son del tipo sistema de información. En el anexo 1 y 2 al final de este documento, se han incluido todos los documentos que se utilizaron en este experimento controlado.

Cada caso de estudio contaba con la siguiente información: diagrama de casos de uso, descripción de cada caso de uso con sus precondiciones e información que se utiliza en cada caso de uso en términos de clases o entidades. En la Figura 4.2 se muestra un ejemplo del contenido de la información brindada en cada caso y por cada caso de uso.



6. Caso de Uso Mantener Cajas

Este caso de uso tiene por objetivo registrar nuevas cajas así como también modificar y eliminar las mismas. Este caso de uso es iniciado por el administrador.

Precondiciones:

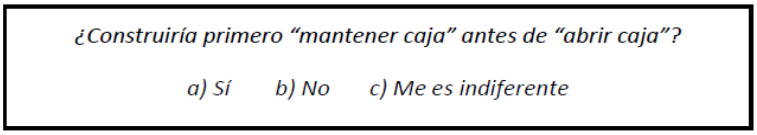
- Se ha debido validar el usuario

Información asociada al caso de uso: caja, usuario

Figura 4.2 Ejemplo de Información de Casos de Uso.

Los cuestionarios 1 y 2 corresponden a preguntas en donde los participantes tenían que decidir entre dos casos de uso y responder cuál de ellos debería ser construido primero.

En estos cuestionarios se incluyeron 12 preguntas sobre secuencias de construcción, a diferencia de la cantidad de preguntas empleadas en las evaluaciones previas que llegaron a 7 por cuestionario. En la Figura 4.3 podemos observar un ejemplo de las preguntas contenidas en cada cuestionario.



¿Construiría primero "mantener caja" antes de "abrir caja"?

a) Sí b) No c) Me es indiferente

Figura 4.3 Ejemplo de Pregunta de los Cuestionarios 1 y 2.

Las preguntas se diseñaron de tal manera que los participantes tuvieran que seleccionar entre los siguientes pares de tipos de casos de uso: maestros-transacciones, transacciones-transacciones, maestros-reportes y transacciones -reportes.

La herramienta que se empleó para el análisis estadístico de los experimentos controlados incluidos en esta sección es el SPSS (IBM Corporation)

4.3.3 Resultados

Para comparar los resultados entre técnicas Ad hoc y UCPD, se empleó el porcentaje de "respuestas correctas", que son aquellas respuestas en las que se considera que la secuencia más fácil de construcción de la funcionalidad de un sistema es maestros-transacciones-reportes.

A continuación mostraremos los resultados obtenidos en cada experimento realizado:

4.3.3.1 EXPERIMENTO 01

Contó con la participación de 13 estudiantes de pregrado de 10mo semestre de la carrera de Ing. Civil Informática de la PUCV. La Tabla 4-3 y la Tabla 4-4 muestran los resultados obtenidos en los dos primeros cuestionarios del estudio.

Tabla 4-3 Resultados del Cuestionario del Caso 1 para el EXPERIMENTO 01.

CASO 1: Registro de Ventas en Restaurante (AD HOC)													
Participante	**Pregunta**												**Total de Respuestas Correctas**
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	
1	1	1	0	0	0	1	1	0	0	0	1	1	6
2	1	0	0	1	0	1	1	0	1	1	0	1	7
3	1	1	1	1	0	0	0	0	0	0	0	1	5
4	1	1	0	0	1	0	0	0	0	1	0	1	5
5	1	1	0	0	0	0	0	0	1	1	0	0	4
6	1	0	1	0	1	1	1	0	0	1	1	0	7
7	1	0	0	1	0	1	0	0	0	0	1	1	5
8	1	0	0	1	1	1	1	1	0	0	1	1	8
9	1	0	1	1	0	1	0	1	1	1	1	1	9
10	1	1	0	0	0	1	0	1	0	0	1	0	5
11	1	1	1	1	0	0	1	1	0	1	1	1	9
12	1	1	0	1	0	1	0	1	0	0	0	1	6
13	1	1	1	0	0	1	1	0	1	1	0	1	8

Tabla 4-4 Resultados del Cuestionario del Caso 2 para el EXPERIMENTO 01.

CASO 2: Registro de Matrículas en Colegio (UCPD)													
Participante	**Pregunta**												**Total de Respuestas Correctas**
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	
1	1	1	1	1	1	1	1	1	1	1	1	1	12
2	0	1	1	0	1	1	1	1	1	1	0	1	9
3	1	1	0	0	1	1	1	1	1	1	0	1	9
4	1	1	1	0	1	1	1	1	1	1	0	1	10
5	1	1	0	0	1	0	0	1	1	0	1	1	7
6	1	1	1	1	1	1	0	1	1	1	1	1	11
7	1	0	1	0	0	1	1	1	0	1	1	1	8
8	1	1	0	0	1	1	1	1	1	1	0	0	8
9	1	1	0	1	1	1	0	1	1	1	0	1	9
10	1	1	1	0	1	1	1	0	1	1	0	1	9
11	1	1	1	0	1	1	1	1	1	1	1	1	11
12	1	1	0	1	1	1	0	0	1	1	1	1	9
13	1	1	1	0	1	1	1	1	0	1	0	0	8

El valor de “1” indica que el participante acertó en la respuesta y “0” que respondió de manera incorrecta. La primera columna muestra el número

de participante, la segunda columna hasta la décimo segunda presenta los resultados que obtuvo cada participante en cada pregunta y la última columna muestra el total de respuestas correctas por cada participante.

Analizando estadísticamente los resultados obtenidos podemos obtener el siguiente diagrama de caja de la Figura 4.4 con los resultados obtenidos al aplicar las técnicas Ad hoc (técnicas propias) de cada estudiante de pregrado con el caso de estudio 1 y al aplicar UCPD con el caso de estudio 2. Como se puede observar, el valor de las medianas es mayor al aplicar UCPD. También, el 75% de los datos al aplicar UCPD se encuentran entre 8 ó 10, a diferencia con las técnicas Ad hoc en las que el 75% se encuentra desde 5 a 7 aciertos. Como se puede observar en el diagrama, los resultados obtenidos al utilizar UCPD son mejores que al emplear las técnicas Ad hoc.

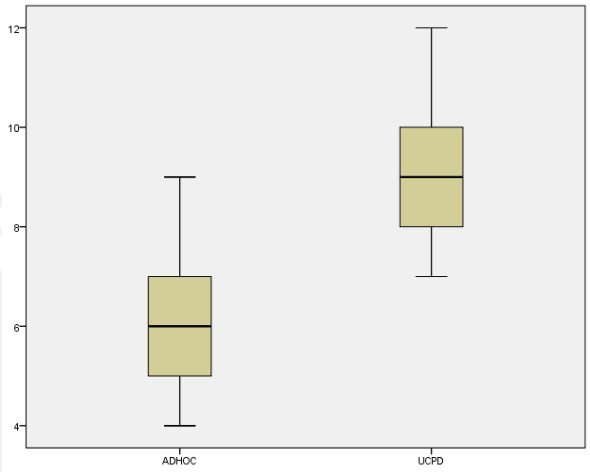


Figura 4.4 Diagramas de Caja de Experimento 01.

Aunque los resultados observados en la Figura 4.4 muestran que se obtuvieron mejores resultados al aplicar UCPD, se realizaron más pruebas estadísticas con la información recopilada en los cuestionarios.

La Tabla 4-5 presenta la información estadística descriptiva obtenida luego de procesar los datos de los cuestionarios que se muestran en la Tabla 4-3 y la Tabla 4-4. La segunda columna muestra los resultados obtenidos con las técnicas propias de los estudiantes de pregrado (técnicas Ad hoc) y la tercera, los resultados con UCPD. Para las técnicas Ad hoc se empleó el caso de estudio relacionado con un sistema de restaurantes (caso de estudio 1) y para UCPD se utilizó el caso de estudio relacionado con un sistema para un colegio de educación secundaria (caso de estudio 2).

Tabla 4-5 Análisis descriptivo estadístico para el EXPERIMENTO 01.

Variable	Técnica AD HOC con Caso 1	Técnica UCPD con Caso 2
Observaciones	13	13
Mínimo	4	7

Máximo	9	12
Media	6.15	9.23
Desviación Estándar	1.405	1.4023

Con el fin de determinar si los resultados obtenidos con cada técnica siguen una distribución normal, se aplicó la prueba de normalidad K-S.

Tabla 4-6 Prueba de Kolmogorov-Smirnov para EXPERIMENTO 01.

Variable	Técnica AD HOC Caso 1	Técnica UCPD Caso 2
K	0.179	0.941
p-valor	0.20	0.019

Según la Tabla 4-6, el p-valor = 0.019 que menor que el nivel de significación establecido 0.05, por lo tanto, se rechaza la hipótesis nula y concluimos que ambas muestras no siguen una distribución normal. Por ello, se tuvo que utilizar la prueba no paramétrica de los rangos señalados de Wilcoxon (Tabla 4-7).

Ho: Med1= Med2, Ambas medias son iguales.

Ha: Med1< Med2, La media en la técnica AD HOC es menor a la media de la técnica UCPD.

Tabla 4-7 Prueba de Wilcoxon AD HOC vs UCPD para EXPERIMENTO 01.

Variable	Resultado
Z	-3,081[b]
P-valor	0.02

De la Tabla 4-7 podemos obtener un p-valor de 0.02 menor al nivel de significación del 0.05. Por lo tanto se rechaza la hipótesis nula.

Concluimos que la media de la muestra aplicando la técnica UCPD es mayor a la de Ad Hoc. Esto quiere decir que se pudo comprobar empíricamente que UCPD produjo mejores resultados (efectividad mayor) que con las técnicas Ad hoc para la muestra obtenida con los estudiantes de pregrado del primer grupo.

4.3.3.2 EXPERIMENTO 02

Contó con la participación de 31 estudiantes de pregrado de 6to semestre de la carrera de Ing. de Ejecución en Informática de la PUCV.

La Tabla 4-8 y la Tabla 4-9 muestran los resultados obtenidos en los dos primeros cuestionarios del estudio. Cómo en el anterior experimento el valor de “1” indica que el participante acertó en la respuesta y “0” que respondió

de manera incorrecta. La primera columna muestra el número de participante, la segunda columna hasta la décimo segunda presenta los resultados que obtuvo cada participante en cada pregunta y la última columna muestra el total de respuestas correctas por cada participante.

Tabla 4-8 Resultados del Cuestionario del Caso 1 para el EXPERIMENTO 02.

CASO 1: Registro de Ventas en Restaurante (AD HOC)													
Participante	Pregunta												Total de Respuestas Correctas
	1	2	3	4	5	6	7	8	9	10	11	12	
1	1	1	0	1	1	1	0	0	0	0	1	1	**7**
2	1	1	1	0	0	0	1	1	0	0	0	0	**5**
3	1	1	0	1	1	1	0	0	0	0	1	1	**7**
4	1	1	0	1	1	1	0	0	0	0	1	1	**7**
5	1	1	1	1	1	1	1	0	0	1	1	1	**10**
6	0	1	0	1	0	0	0	0	0	1	0	1	**4**
7	1	0	1	1	0	1	1	0	0	1	0	1	**7**
8	1	0	0	0	0	0	1	1	0	1	0	1	**5**
9	1	0	0	1	0	1	1	0	0	1	0	1	**6**
10	1	0	0	1	0	0	0	1	0	0	0	1	**4**
11	1	0	1	1	0	1	1	1	1	1	1	1	**10**
12	1	0	0	0	1	0	1	1	0	0	1	1	**6**
13	1	1	0	1	0	1	1	1	0	0	0	1	**7**
14	1	1	1	1	0	0	1	1	0	0	0	0	**6**
15	0	1	0	1	0	1	0	1	0	0	0	1	**5**
16	1	0	0	1	0	0	1	0	0	1	0	1	**5**
17	1	1	1	0	0	0	1	0	1	1	1	1	**8**
18	1	1	0	1	1	1	0	1	0	1	0	1	**8**
19	1	0	0	1	0	1	1	1	1	0	1	1	**8**
20	1	1	0	1	0	1	1	1	0	1	0	1	**8**
21	1	1	1	1	0	1	1	1	1	1	1	1	**11**
22	1	0	1	1	0	1	1	0	1	1	1	1	**9**
23	1	0	1	1	0	1	1	1	1	1	1	1	**10**
24	1	1	1	1	0	1	0	1	0	1	0	1	**8**
25	0	1	1	1	0	0	1	1	1	1	0	0	**7**
26	0	0	1	1	0	1	1	0	0	0	0	1	**5**
27	1	1	1	1	0	1	1	0	0	0	1	1	**8**
28	1	1	0	1	0	1	1	0	1	0	0	1	**7**
29	1	1	1	0	0	1	0	1	0	1	0	0	**6**
30	1	1	0	1	0	1	1	1	1	1	1	1	**10**
31	1	1	1	1	0	1	1	1	1	1	1	1	**11**

Tabla 4-9 Resultados del Cuestionario del Caso 2 para el EXPERIMENTO 02.

CASO 2: Registro de Matrícula en Colegio (UCPD)													
Participante	**Pregunta**												**Total de Respuestas Correctas**
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	
1	1	1	0	1	1	1	1	1	1	1	1	1	**11**
2	1	1	1	0	0	0	1	1	1	1	1	0	**8**
3	1	1	0	1	1	1	0	0	1	1	1	1	**9**
4	1	1	0	1	1	1	0	0	1	0	1	1	**8**
5	1	1	1	1	1	1	1	0	0	1	1	1	**10**
6	0	1	0	1	0	1	1	0	1	1	1	1	**8**
7	1	0	1	1	1	1	1	0	0	1	1	1	**9**
8	1	1	0	0	0	0	1	1	1	1	1	1	**8**
9	1	0	0	1	1	1	1	1	1	1	1	1	**10**
10	1	0	0	1	1	0	0	1	1	1	1	1	**8**
11	1	0	1	1	0	1	1	1	1	1	1	1	**10**
12	1	1	1	1	1	0	1	1	1	1	1	1	**11**
13	1	1	0	1	0	1	1	1	1	1	1	1	**10**
14	1	1	1	1	0	0	1	1	0	1	1	0	**8**
15	0	1	0	1	1	1	0	1	1	1	1	1	**9**
16	1	0	1	1	1	0	1	0	0	0	0	0	**5**
17	1	1	1	0	1	1	1	0	1	1	1	1	**10**
18	1	1	0	1	1	1	0	1	0	1	0	1	**8**
19	1	0	1	1	1	1	1	1	1	1	1	1	**11**
20	1	1	0	1	1	1	1	1	1	1	0	1	**10**
21	1	1	1	1	1	1	1	1	1	1	1	1	**12**
22	1	0	1	1	1	1	1	1	1	1	1	1	**11**
23	1	0	1	1	1	1	1	1	1	1	1	1	**11**
24	1	1	1	1	0	1	1	1	1	1	0	1	**10**
25	1	0	0	0	0	0	1	1	0	1	1	1	**6**
26	1	0	1	1	1	1	1	0	0	0	1	1	**8**
27	1	1	1	1	1	1	1	1	1	0	1	1	**11**
28	1	1	1	1	0	1	1	0	1	1	1	1	**10**
29	1	1	1	1	1	1	0	1	0	1	1	1	**10**
30	1	1	0	1	0	1	1	1	1	1	1	1	**10**
31	1	1	1	1	0	1	1	1	1	1	1	1	**11**

Analizando estadísticamente los resultados obtenidos podemos obtener el siguiente diagrama de caja de la Figura 4.5 con los resultados obtenidos al aplicar las técnicas Ad hoc (técnicas propias) de cada estudiante de pregrado del segundo grupo con el caso de estudio 1 y al aplicar UCPD con el caso de estudio 2. Como se puede observar, el valor de las medianas es mayor al aplicar UCPD. También, el 75% de los datos al aplicar UCPD se encuentran entre 8 ó 11, a diferencia con las técnicas Ad hoc en las que el 75% se

encuentra desde 6 a 8 aciertos. Como se puede observar en el diagrama, los resultados obtenidos al utilizar UCPD son mejores que al emplear las técnicas Ad hoc al igual que el primer grupo evaluado.

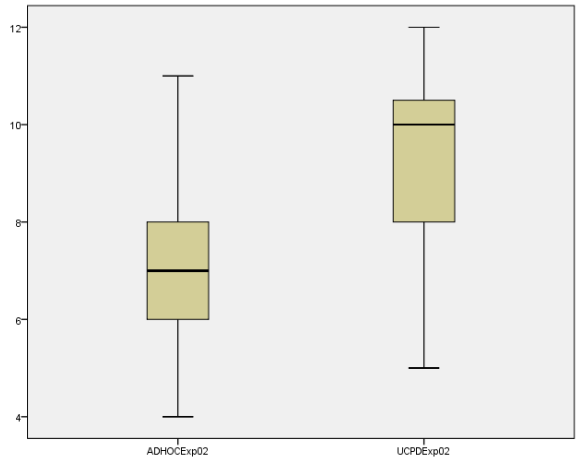


Figura 4.5 Diagramas de Caja de Experimento 02.

La Tabla 4-10 presenta la información estadística descriptiva obtenida luego de procesar los datos de los cuestionarios que se muestran en la Tabla 4-8 y la Tabla 4-9. La segunda columna muestra los resultados obtenidos con las técnicas propias de los estudiantes de pregrado (técnicas Ad hoc) y la tercera, los resultados con UCPD. Para las técnicas Ad hoc se empleó el caso de estudio relacionado con un sistema de restaurantes (caso de estudio 1) y para UCPD se utilizó el caso de estudio relacionado con un sistema para un colegio de educación secundaria (caso de estudio 2).

Tabla 4-10 Análisis descriptivo estadístico para el EXPERIMENTO 02.

Variable	Técnica AD HOC con Caso 1	Técnica UCPD con Caso 2
Observaciones	31	31
Mínimo	4	5
Máximo	11	12
Media	7.26	9.39
Desviación Estándar	1.983	1.585

Con el fin de determinar si los resultados obtenidos con cada técnica siguen una distribución normal, se aplicó la prueba de normalidad Shapiro-Wilk (Tabla 4-11) al tratarse de una muestra mayor a 30 participantes.

Tabla 4-11 Prueba de Shapiro-Wilk para EXPERIMENTO 02.

Variable	Técnica AD HOC Caso 1	Técnica UCPD Caso 2
W	0.945	0.898
p-valor	0.116	0.006

Según la Tabla 4-11, el p-valor de la técnica AD HOC es mayor al nivel de significación de 0.05, sin embargo el p-valor de la técnica UCPD es menor por lo que ambas muestras no siguen una relación normal. Por lo tanto aplicaremos la comparación de medias mediante Wilcoxon (Tabla 4-12).

Ho: Med1= Med2, Ambas medias son iguales.

Ha: Med1< Med2, La media en la técnica AD HOC es menor a la media de la técnica UCPD.

Tabla 4-12 Prueba de Wilcoxon AD HOC vs UCPD para EXPERIMENTO 02.

Variable	Resultado
Z	-4,336[b]
P-valor	0.00

Obtenemos un p-valor = 0.00 que es menor a 0.05, por lo tanto rechazamos la hipótesis nula y concluimos que la media de la técnica AD HOC es menor a la media de la técnica UCPD. Esto quiere decir que se pudo comprobar nuevamente empíricamente que UCPD produjo mejores resultados (efectividad mayor) que con las técnicas Ad hoc para la muestra obtenida con los estudiantes de pregrado.

4.3.3.3 EXPERIMENTO 03

El tercer experimento contó con la participación de 21 estudiantes de pregrado de 7mo y 8vo semestre de la carrera de Ing. De Ejecución en Informática de la PUCV.

La Tabla 4-13 y la Tabla 4-14 muestran los resultados obtenidos en los dos primeros cuestionarios del estudio. De igual manera, como en los anteriores experimentos, el valor de “1” indica que el participante acertó en la respuesta y “0” que respondió de manera incorrecta. La primera columna muestra el número de participante, la segunda columna hasta la décimo segunda presenta los resultados que obtuvo cada participante en cada pregunta y la última columna muestra el total de respuestas correctas por cada participante.

Tabla 4-13 Resultados del Cuestionario del Caso 1 para el EXPERIMENTO 03.

CASO 1: Registro de Ventas en Restaurante (AD HOC)													
Participante	Pregunta												Total de Respuestas Correctas
	1	2	3	4	5	6	7	8	9	10	11	12	
1	1	1	1	1	0	0	1	1	0	0	0	0	**6**
2	1	1	0	1	0	1	0	1	0	0	0	1	**6**
3	1	1	0	1	1	1	0	0	0	0	1	1	**7**
4	1	0	1	1	0	1	1	0	0	1	0	1	**7**
5	1	0	0	0	0	0	1	1	0	1	0	1	**5**

CASO 1: Registro de Ventas en Restaurante (AD HOC)													
Participante	Pregunta												Total de Respuestas Correctas
	1	2	3	4	5	6	7	8	9	10	11	12	
6	0	1	0	1	0	0	0	0	0	1	0	1	**4**
7	1	0	1	1	0	1	1	0	0	0	0	1	**6**
8	1	1	1	1	0	1	1	0	0	0	1	1	**8**
9	1	0	0	1	0	1	1	0	0	1	0	1	**6**
10	1	0	0	1	0	0	0	1	0	0	0	1	**4**
11	1	0	1	1	0	1	1	1	1	1	1	1	**10**
12	1	0	0	0	1	0	1	1	0	0	1	1	**6**
13	1	1	0	1	0	1	1	1	0	0	0	1	**7**
14	1	1	1	1	0	0	1	1	0	0	0	0	**6**
15	0	1	0	1	0	1	0	1	0	0	0	1	**5**
16	1	0	0	1	0	0	1	0	0	1	0	1	**5**
17	1	1	1	0	0	0	1	0	1	1	1	1	**8**
18	1	1	0	1	1	1	0	1	0	1	0	1	**8**
19	1	0	1	1	0	1	1	0	0	0	0	1	**6**
20	1	1	1	1	0	1	1	0	0	0	1	1	**8**
21	0	1	0	1	0	1	0	1	0	0	0	1	**5**

Tabla 4-14 Resultados del Cuestionario del Caso 2 para el EXPERIMENTO 03.

CASO 2: Registro de Matrícula en Colegio (UCPD)													
Participante	Pregunta												Total de Respuestas Correctas
	1	2	3	4	5	6	7	8	9	10	11	12	
1	1	1	0	0	0	0	1	1	1	1	1	1	**8**
2	1	0	0	1	1	1	1	1	1	1	1	1	**10**
3	1	0	0	1	1	0	0	1	1	1	1	1	**8**
4	1	0	1	1	0	1	1	1	1	1	1	1	**10**
5	1	1	1	1	1	0	1	1	1	1	1	1	**11**
6	1	1	0	1	0	1	1	1	1	1	1	1	**10**
7	1	1	1	1	0	0	1	1	0	1	1	0	**8**
8	0	1	0	1	1	1	0	1	1	1	1	1	**9**
9	1	0	1	1	1	0	1	0	0	0	0	0	**5**
10	1	1	1	0	1	1	1	0	1	1	1	1	**10**
11	1	1	0	1	1	1	0	1	0	1	0	1	**8**
12	1	0	0	0	0	0	1	1	0	1	1	1	**6**
13	1	0	1	1	1	1	1	0	0	0	1	1	**8**
14	1	1	0	1	1	1	0	0	1	1	1	1	**9**
15	1	1	0	1	1	1	0	0	1	0	1	1	**8**
16	1	1	1	1	1	1	1	0	0	1	1	1	**10**
17	1	1	1	1	0	0	1	1	0	1	1	0	**8**
18	0	1	0	1	1	1	0	1	1	1	1	1	**9**
19	1	0	1	1	1	0	1	0	0	0	0	0	**5**
20	1	1	1	0	1	1	1	0	1	1	1	1	**10**
21	1	1	0	1	1	1	0	1	0	1	0	1	**8**

Analizando estadísticamente los resultados obtenidos podemos obtener el siguiente diagrama de caja de la Figura 4.6 con los resultados obtenidos al aplicar las técnicas Ad hoc (técnicas propias) de cada estudiante de pregrado del segundo grupo con el caso de estudio 1 y al aplicar UCPD con el caso de estudio 2. Como se puede observar, el valor de las medianas es mayor al aplicar UCPD. También, el 75% de los datos al aplicar UCPD se encuentran entre 8 ó 10, a diferencia con las técnicas Ad hoc en las que el 75% se encuentra desde 5 a 7 aciertos. Como se puede observar en la Figura 4.6, los resultados obtenidos al utilizar UCPD son mejores que al emplear las técnicas Ad hoc al igual que el primer grupo evaluado.

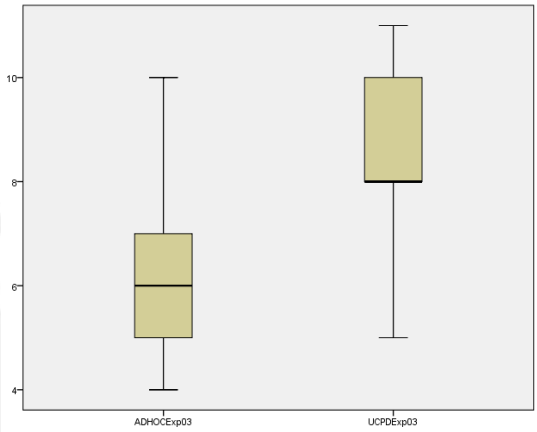


Figura 4.6 Diagramas de Caja de Experimento 03

La Tabla 4-15 presenta la información estadística descriptiva obtenida luego de procesar los datos de los cuestionarios que se muestran en la Tabla 4-13 y la Tabla 4-14. La segunda columna muestra los resultados obtenidos con las técnicas propias de los estudiantes de pregrado (técnicas Ad hoc) y la tercera, los resultados con UCPD. Para las técnicas Ad hoc se empleó el caso de estudio relacionado con un sistema de restaurantes (caso de estudio 1) y para UCPD se utilizó el caso de estudio relacionado con un sistema para un colegio de educación secundaria (caso de estudio 2).

Tabla 4-15 Análisis descriptivo estadístico para el EXPERIMENTO 03.

Variable	Técnica AD HOC con Caso 1	Técnica UCPD con Caso 2
Observaciones	21	21
Mínimo	4	5
Máximo	10	11
Media	6.36	8.48
Desviación Estándar	1.494	1.632

Con el fin de determinar si los resultados obtenidos con cada técnica siguen una distribución normal, se aplicó la prueba de normalidad K-S (Tabla 4-16).

Tabla 4-16 Prueba de Kolmogorov-Smirnov para EXPERIMENTO 03.

Variable	Técnica AD HOC Caso 1	Técnica UCPD Caso 2
K	0.207	0.242
p-valor	0.019	0.002

El p-valor de la técnica AD HOC es > a 0.05, sin embargo el p-valor de la técnica UCPD es < a 0.05 por lo que ambas muestras no siguen una relación normal. Por lo tanto aplicaremos la comparación de medias mediante Wilcoxon (Tabla 4-17).

Ho: Med1= Med2, Ambas medias son iguales.

Ha: Med1< Med2, La media en la técnica AD HOC es menor a la media de la técnica UCPD.

Tabla 4-17 Prueba de Wilcoxon AD HOC vs UCPD para EXPERIMENTO 03.

Variable	Resultado
Z	-3,219[b]
P-valor	0.001

Obtenemos un p-valor = 0.001 que es menor a 0.05, por lo tanto rechazamos la hipótesis nula y concluimos que la media de la técnica AD HOC es menor a la media de la técnica UCPD. Esto quiere decir que se pudo comprobar nuevamente empíricamente que UCPD produjo mejores resultados (efectividad mayor) que con las técnicas Ad hoc para la muestra obtenida con los estudiantes de pregrado.

4.3.4 Conclusiones y Comparación de la evaluación de efectividad de UCPD

Como podemos observar en la Tabla 4-18, en los 3 experimentos realizados en la PUCV hemos obtenido que la media de los valores correspondiente a la técnica UCPD es mayor que las técnicas Ad hoc. Esto demuestra que la técnica UCPD es más efectiva para definir una correcta secuencialidad de construcción de los casos de uso.

Tabla 4-18 Resumen de los resultados de la evaluación de efectividad de UCPD en la PUCV

N°	Número de Preguntas	Media Ad hoc	Media UCPD
Experimento 1	12	6.15	**9.23**
Experimento 2	12	7.26	**9.39**
Experimento 3	12	6.36	**8.48**

Comparando con los cuestionarios realizados previamente en la PUCP, en este caso se añadió más preguntas en cada cuestionario de los casos de estudio con el fin de obtener resultados más precisos y reales.

Tabla 4-19 Resumen de resultados de la evaluación de efectividad de UCPD en la PUCP

N°	Número de Preguntas	Media Ad hoc	Media UCPD
Estudio 2	6	4.638	**5.42**
Estudio 3	7	4	**5.85**
Estudio 4	7	4.74	**6.02**

Como podemos observar en la Tabla 4-18 y la Tabla 4-19, la media de las cantidades de respuestas correctas obtenidas para UCPD es mayor a la obtenida con la técnica Ad hoc.

Esto nos lleva a concluir que tanto en el contexto peruano como el chileno, la técnica UCPD obtiene resultados satisfactorios para definir la secuencia de construcción de los casos de uso.

Sin embargo, debemos tener en cuenta que la cantidad de preguntas a desarrollar es diferente entre los experimentos de la PUCV y los previos realizados en la PUCP.

Tabla 4-20 Resumen de % de aciertos para UCPD en la PUCV y la PUCP

N°	% de aciertos para UCPD
PUCV - Experimento 1	76.9%
PUCV – Experimento 2	78.3%
PUCV – Experimento 3	70.7%
PUCP – Estudio 2	90.3%
PUCP – Estudio 3	83.6%
PUCP – Estudio 4	86.0%

Para las réplicas en la PUCV, con 12 preguntas, se obtuvo menos % de aciertos que los experimentos previos en la PUCP con 6 y 7 preguntas. Esto nos lleva también a evaluar la posibilidad de mejorar los casos de estudio para futuras réplicas que pudiesen llevarse a cabo. A mayor cantidad de preguntas, menor porcentaje de aciertos. Esto puede deberse a algunos factores como la fatiga.

4.4 Evaluación de la percepción de UCPD.

4.4.1 Objetivo

Medir la percepción de la técnica UCPD de los participantes de cada experimento con respecto a 3 constructos: facilidad de uso, utilidad e intención de uso percibida.

4.4.2 Diseño e Instrumento

Con el fin de evaluar la percepción de participantes de los experimentos controlados sobre UCPD se aplicó un cuestionario una vez finalizada la solución de ambos casos de estudio en el que tuvieron que opinar sobre tres aspectos: facilidad de uso, utilidad e intención de uso de la técnica en otros proyectos. El cuestionario contiene de las siguientes preguntas:

Tabla 4-21 Cuestionario de percepción para UCPD

DIAGRAMA DE PRECEDENCIAS	-				+
4. Facilidad para construir o realizar el diagrama de precedencias	**0**	**1**	**2**	**3**	**4**
5. Utilidad del diagrama de precedencias para determinar la secuencia de construcción de software (Utilidad es el grado en el cual UCPD será efectivo para lograr los objetivos propuestos; es decir determinar la secuencia de construcción de software)	**0**	**1**	**2**	**3**	**4**
6. ¿Usaría el diagrama de precedencias para siguientes proyectos de desarrollo de software?	**0**	**1**	**2**	**3**	**4**
7. Importancia de establecer una correcta secuencia de construcción de casos de uso al iniciar un proyecto	**0**	**1**	**2**	**3**	**4**

Se agregó una pregunta más al cuestionario para determinar la percepción de la importancia sobre establecer una correcta secuencia de construcción previa al desarrollo de un proyecto. Esto nos puede ayudar con fines referenciales para interpretar algunos resultados.

Este cuestionario fue aplicado a los grupos de los 3 experimentos mencionados con anterioridad. Las preguntas de investigación que se han identificado son:

¿UCPD es percibida como fácil de usar?
¿UCPD es percibida como útil?
¿Existe la intención de usar UCPD en futuros proyectos?

La hipótesis estadística para evaluar cada pregunta son las siguientes:

$$Ho: \mu \leq 2, \alpha = 0.05$$

Ha: μ>2

4.4.3 Resultados

A continuación mostraremos los resultados obtenidos para la percepción de la técnica UCPD por parte de los participantes de cada experimento. Se evaluarán los 3 constructos por separado

4.4.3.1 Facilidad de Uso percibida

En cuanto a la pregunta de investigación sobre facilidad de uso de UCPD, la Tabla 4-22 presenta el resultado del análisis descriptivo realizado para los datos obtenidos de la pregunta sobre facilidad de uso de los 3 experimentos.

Tabla 4-22 Estadísticos descriptivos de Facilidad de Uso

FACILIDAD DE USO			
VARIABLES	**EXP 01**	**EXP 02**	**EXP03**
Observaciones	13	31	20
Mínimo	2	1	1
Máximo	3	4	3
Media	2.62	2.58	2.10
Desviación Estándar	0.506	0.848	0.718

Se puede observar que las medias son mayores al valor central 2. El rango de valores osciló entre 2 y 3 para EXP 01, entre 1 y 4 para EXP 02 y entre 1 y 3 para EXP 03.

Por lo tanto, podemos decir que UCPD fue percibida como fácil de usar.

4.4.3.2 Utilidad Percibida

En cuanto a la pregunta de investigación sobre la utilidad percibida de UCPD, la

Tabla 4-23 presenta el resultado del análisis descriptivo realizado para los datos obtenidos de la pregunta sobre utilidad de uso en los 3 experimentos.

Se puede observar que las medias son mayores al valor central 2. El rango de valores osciló entre 2 y 4 para todos los experimentos. Por lo tanto, podemos decir que UCPD fue percibida como útil en los 3 experimentos.

Tabla 4-23 Estadísticos descriptivos de Utilidad Percibida

UTILIDAD PERCIBIDA			
VARIABLES	**EXP 01**	**EXP 02**	**EXP03**
Observaciones	13	31	20
Mínimo	2	2	2
Máximo	4	4	4

UTILIDAD PERCIBIDA			
VARIABLES	**EXP 01**	**EXP 02**	**EXP03**
Media	3.46	3.48	2.85
Desviación Estándar	0.660	0.626	0.587

4.4.3.3 Intención de Uso Percibida

En cuanto a la pregunta de investigación sobre la utilidad percibida de UCPD, la Tabla 4-24 presenta el resultado del análisis descriptivo realizado para los datos obtenidos de la pregunta sobre utilidad de uso en los 3 experimentos.

Tabla 4-24 Estadísticos descriptivos de Intención de Uso Percibida

INTENCIÓN DE USO PERCIBIDA			
VARIABLES	**EXP 01**	**EXP 02**	**EXP03**
Observaciones	13	31	20
Mínimo	2	0	0
Máximo	4	4	4
Media	3.00	3.00	2.25
Desviación Estándar	0.707	1.238	1.070

Se puede observar que las medias son mayores al valor central 2. El rango de valores osciló entre 2 y 4 para EXP01, 0 y 4 para EXP02 y entre 0 y 4 para EXP03. Podemos observar en este caso que la intención de uso llegó a 0 y las desviaciones estándar de los últimos 2 experimentos suelen ser altas lo que indica una heterogeneidad para los valores resultantes.

Por lo tanto, podemos decir hay intención de uso para UCPD con alumnos de pregrado.

Así mismo, cabe aclarar que la aparición del valor 0 en intención de uso en los últimos 2 experimentos puede deberse a que dichos experimentos fueron realizados con alumnos de cursos cuya naturaleza no estaba 100% relacionada al estudio, lo cual podría verse reflejado en una falta de motivación e interés por parte de los participantes.

4.4.4 Conclusiones y Comparación de los resultados de la evaluación de percepción de UCPD

Los resultados obtenidos muestran de manera general una actitud positiva de los estudiantes hacia la técnica. Los participantes percibieron que UCPD es fácil de usar y útil, y que también la emplearían en futuros proyectos de software.

Se utilizó el cuestionario en su última versión, considerando que tuvo 3 versiones, cambiando para adaptarse a la situación y tipo de población a evaluar.

Comparando los resultados con los obtenidos en la PUCP, notamos que son similares. En ambos casos se muestra una actitud positiva para la técnica en los 3 aspectos mencionados.

Así mismo, de los comentarios obtenidos de los participantes podemos citar uno que personalmente considero importante:

- "Considero que la técnica es muy útil en el desarrollo de software. Los diagramas conocidos de UML por ejemplo, reflejan secuencia dentro de cada caso de uso (diagrama de secuencia) pero no existe uno que explique la secuencia de todo el sistema por lo que considero esta técnica es muy útil para ello".

Como se menciona, en UML, que podría decirse que es el lenguaje de modelamiento más conocido, no se considera un diagrama que exprese la temporalidad de un sistema, es decir, que refleje cuál es el orden de construcción que se definirá para el desarrollo de un software.

4.5 Conclusiones Finales de la Validación de UCPD

- UCPD es una técnica para la priorización de casos de uso
- Como se puede observar, los resultados obtenidos sobre la percepción de UCPD son muy similares a los obtenidos los estudios realizados en la PUCP.
- Del lado experimental, se puede apreciar que la técnica UCPD es más efectiva y este resultado se repite en los 3 experimentos realizados en esta investigación que, a su vez, coincide con lo obtenido en los experimentos previos realizados en la PUCP.
- En la evaluación de percepción, los resultados obtenidos en la PUCV para los 3 constructos es positiva en los 3 experimentos realizados. Estos resultados coinciden también con los obtenidos en la PUCP.
- Finalmente podemos afirmar se ha validado la efectividad y la percepción de la técnica UCPD en el contexto de la PUCV.
- Cabe resaltar que luego de los experimentos se logró realizar pequeñas entrevistas a algunos participantes sobre la técnica.
- La mayoría se vio interesado en obtener mayor información pues consideraban que es útil para poder generar un orden al momento de desarrollar proyectos pues por lo general esta secuencialidad de hace de manera "informal"

- En algunos casos, recomendaron poder determinar una plantilla que permita hacer la especificación de este diagrama para mejorar su entendimiento por parte de otros desarrolladores.

Capítulo 5. Validación: Evaluación de la técnica UML_2FP

5.1 Introducción

En este capítulo realizaremos la validación de la técnica UML_2FP por medio de un experimento con alumnos de la PUCV. El objetivo es medir la precisión del cálculo de ficheros usando UML_2FP de manera manual frente a los resultados que se obtendrían utilizando la herramienta Tupux (Ver apartado 3.3) que automatiza el cálculo.

5.2 Metodología de la evaluación

Se realizó un experimento con 17 alumnos de pregrado del curso de Ing. de Software de la PUCV. Se buscaba medir la precisión en el cálculo de ficheros utilizando la técnica UML_2FP utilizando un caso de estudio y un diagrama de clases.

A continuación se detalla la lista de tareas que se llevaron a cabo en el experimento:

Tarea	Descripción
1	**Inducción a UML_2FP** Los participantes recibieron una inducción a la técnica UML_2FP para el cálculo de ficheros. Se les explicó las reglas para la contabilización de ficheros y se les entregó material impreso con ejemplos prácticos.
2	**Recibir Caso de Estudio y plantilla de cálculo** Los participantes reciben un caso de estudio que consta de un diagrama de análisis. En base a este diagrama, debieron obtener el número de ficheros de acuerdo a las reglas de UML_2FP y la plantilla de cálculo entregada.

5.2.1 Diseño e instrumentos utilizados para la evaluación

5.2.1.1 Caso de estudio

Se trabajó con un caso de estudio que incluye un diagrama de clase análisis que representa la información sobre empleados, asignación de tareas y facturas y una plantilla para el cálculo correspondiente. Ambos instrumentos fueron los mismos que se utilizaron en las experimentaciones previas en la PUCP.

En el diagrama de clase análisis (Figura 5.1) se utiliza relaciones de asociación, agregación/composición, generalización, clase asociación y multiplicidad, de tal manera que puedan hacer uso de las reglas (Ver apartado 3.2.3) por cada tipo de relación entre clases. La descripción de cada clase se especifica en la Tabla 5-1.

Tabla 5-1 Descripción de Clases del Caso de Estudio UML$_2$FP

Clase	Atributos
Factura	Número, Monto
LíneaFactura	Número, Cantidad
Producto	Código, Nombre , Precio
Localización	Nombre, Dirección, Ciudad, Estado, País
Empleado	DNI, Nombre, NúmeroHijos
Hijo	DNI, Nombre, FechaNacimiento
EmpleadoTiempoCompleto	NivelSupervisión
EmpleadoTiempoParcial	HorasSemanaPromedio
AsignaciónTrabajo	Número, FechaAsignación, Salario
Trabajo	Código, Nombre, Descripción

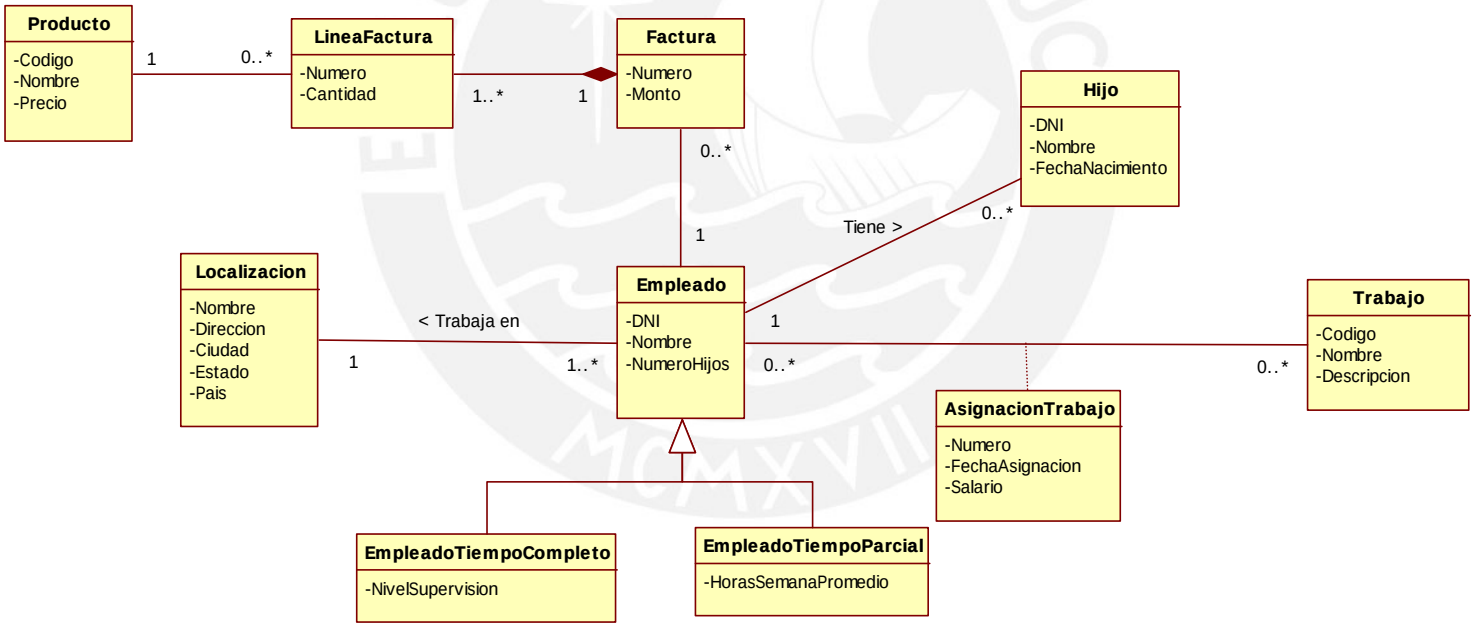


Figura 5.1 Diagrama de Clase Análisis de Caso de Estudio para UML$_2$FP

Para poder realizar el cálculo de ficheros, se entregó una plantilla donde deberán especificar los ficheros encontrados, el nombre de las DETs identificadas por cada uno, el número de DETs y RETs y el cálculo de PFSA. El diseño de la plantilla corresponde al siguiente diseño:

Tabla 5-2 Modelo de Plantilla para el Cálculo de Ficheros

Nombre Fichero	Nombre de DETs	Núm. DET	Núm. RET	PFSA

Utilizando las reglas de UML$_2$FP (Pow-Sang J. A., 2012), se debería identificar los siguientes casos:

- De acuerdo a la regla 5, se debe considerar la clase AsociaciónTrabajo cmo fichero, de acuerdo a la regla 4, la clase asociación AsociaciónTrabajo, Trabajo y Empleado son parte de ficheros diferentes cada uno.
- Usando la regla 3, Empleado, EmpleadoTiempoCompleto y EmpleadoTiempoParcial son parte de un solo fichero con 2 RETs (un RET por cada clase hija)
- De acuerdo a la regla 2, Hijo y Empleado son parte de un solo fichero, pues Hijo es dependiente de Empleado, Eso significa que Empleado, Hijo, EmpleadoTiempoCompleto y EmpleadoTiempoParcial cuentan como un mismo fichero que tiene 3 RETs (uno más porque existe la clase Hijo)
- Usando la regla 2, la Localización es independiente de Empleado. Por tal motivo, Localización forma otro fichero.
- De acuerdo a la regla 2, Factura es independiente de Empleado. Por lo tanto, Factura forma otro fichero.
- Usando la regla 1b, Factura y LíneaFactura son parte de un mismo fichero (que debería tener 2 RETs) y Producto es parte de otro fichero.

De acuerdo a lo descrito, los resultados que se deben obtener en el cálculo corresponden a la Tabla 5-3.

Tabla 5-3 Resultado del conteo de ficheros para el caso de estudio

Fichero	DET			RET	Complejidad
	Por atributo	Por multiplicidad de muchos	Total		
AsignaciónTrabajo	3	2	5	1	Baja
Trabajo	3	0	3	1	Baja
Empleado-EmpleadoTParcial-EmpleadoTCompleto-Hijo	8	1	9	3	Baja
Localización	5	0	5	1	Baja
Factura-LíneaFactura	4	2	6	2	Baja

Fichero	DET			RET	Complejidad
	Por atributo	Por multiplicidad de muchos	Total		
Producto	3	0	3	1	Baja

5.2.2 Resultados de la Evaluación

Una vez finalizado el experimento, cada participante entregó la plantilla de cálculo con la solución correspondiente. A continuación detallaremos los resultados de dicho cálculo.

5.2.2.1 Errores encontrados

De acuerdo a los resultados obtenidos (Tabla 5-4), se detectó que todos los participantes no identificaron la relación de dependencia entre Hijo y las clases Empleado, y consideraron a Hijo como un fichero aparte. Este error puede causar el incremento de 7 PFSA en el cálculo final, pues Empleado e Hijo deben ser considerados como un solo fichero.

Tabla 5-4 Resumen de Errores cometidos en el cálculo con UML_2FP para la PUCV

Errores en identificación de Ficheros		Errores en contabilización de RETs	Error en multiplicidad de muchos (DET)
Independencia Hijo-Empleado	***Otros***		
100% (17 alumnos)	52.94% (9 alumnos)	100% (17 alumnos)	70.5% (alumnos)

5.2.2.2 Resultados calculados con Tupux

Con la información obtenida de los alumnos, se hizo una simulación sobre los resultados que dichos alumnos podrían obtener si usaran Tupux. Si los participantes hubiesen usado Tupux, ellos sólo hubiesen tenido que identificar correctamente la dependencia entre Empleado e Hijo. Esto quiere decir, que el tiempo calculando PFSA hubiese sido menor usando Tupux.

Podemos observar en la Figura 5.2 que la dispersión de los PFSA calculados por los estudiantes de pregrado de la PUCV es mayor de los resultados obtenidos con Tupux. Todos los estudiantes de pregrado habrían obtenido 49 PFSA con Tupux, pero sin Tupux, número de PFSA varía de 20 a 70 PFSA. Sin embargo, la mediana, es decir 49 PFSA, es el mismo para ambas muestras.

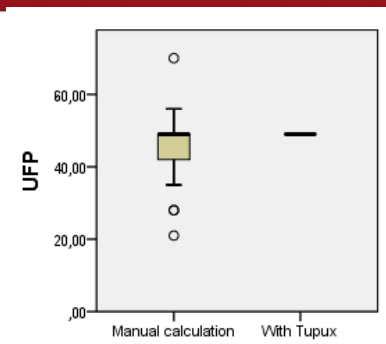


Figura 5.2 Diagrama de caja de PFSA calculados manualmente y con Tupux de los alumnos de la PUCV

A pesar que ambas medias de los PFSA son 49, el número de PFSA de acuerdo a IFPUG (Internacional Function Point User Group, 2004) debería ser 42.

Para incrementar la confiabilidad de resultados, se comparó lo encontrado en la PUCV con lo obtenido en los experimentos previos en la PUCP (Pow-Sang, Flores, & Villanueva, A Conversion Model and a Tool to Identify Function Point Logic Files using UML Analysis Class Diagrams, 2013) con grupos de 10 y 7 alumnos. De estos experimentos se obtuvo lo siguiente:

Tabla 5-5 Resultados de experimentos previos de UML_2FP en la PUCP

Grupo	**Errores en identificación de Ficheros**		**Errores en contabilización de RETs**	**Error en multiplicidad de muchos (DET)**
	Independencia Hijo-Empleado	***Otros***		
Practicantes de la PUCP Semestre 2012-1 (10 alumnos)	90.0% (9 alumnos)	30.0% (3 alumnos)	60.0% (6 alumnos)	80.0% (8 alumnos)
Practicantes de la PUCP Semestre 2012-2 (10 alumnos)	71.4 % (5 alumnos)	28.57% (2 alumnos)	42.86% (3 alumnos)	14.29% (1 alumnos)

Así mismo se pudo obtener la evaluación de sus medias con ayuda de un diagrama de caja.

Para el caso del Grupo de los alumnos de PUCP del semestre 2012-1 (Figura 5.3), se obtuvo resultados similares a los obtenidos con los alumnos de la PUCV. Todos los practicantes, excepto uno, habrían obtenido 49 PFSA con Tupux, pero sin Tupux, número de PFSA varía de 35 a 70 PFSA. Sin embargo, la mediana, es decir 49 PFSA, es el mismo para ambas muestras.

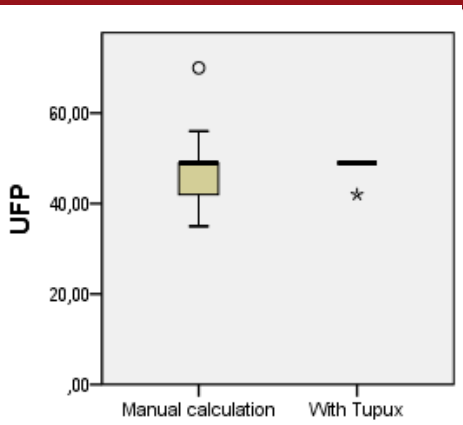


Figura 5.3 Diagrama de caja de PFSA calculados manualmente y con Tupux de los alumnos de la PUCP 2012-1

Para el caso del Grupo de los alumnos de PUCP del semestre 2012-2 (Figura 5.4), aunque la cantidad de PFSA calculados con y sin Tupux varía de 42 a 49, la dispersión de los datos mediante Tupux es menor que sin utilizar Tupux.

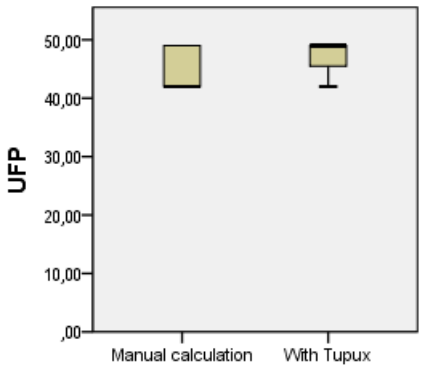


Figura 5.4 Diagrama de caja de PFSA calculados manualmente y con Tupux de los alumnos de la PUCP 2012-2

Por lo tanto, se compara la eficacia obtenida por el cálculo manual y con Tupux. Para hacer dicha comparación, utilizamos la Magnitud del Error relativo (MRE) de manera similar que es empleado por Abrahao (Abrahão, Insfran, Carsí, & Genero, 2011). La MRE de cada cálculo realizado con o sin Tupux se obtendrá mediante la siguiente ecuación:

$$MRE = \frac{\left| 42 - PFSA\_calculados \right|}{42}$$

Se estableció un nivel de significancia de 0.05 para probar estadísticamente los resultados obtenidos. Puesto que los resultados obtenidos (MREs calculados) siguen una distribución no normal, no se podía utilizar la prueba-t de muestras pareadas y, por lo tanto, elegimos que la

prueba de Wilcoxon, una alternativa no paramétrica. Las hipótesis estadísticas formuladas para comprobar los resultados obtenidos con y sin Tupux son:

- Ho: MRE obtenidos mediante Tupux y sin usar Tupux no son significativamente diferente.
- Ha: MRE obtenido con Tupux es menor que los resultados obtenidos sin Tupux.

Se realizaron los cálculos respectivos obteniendo la siguiente tabla:

Tabla 5-6 Análisis con Wilcoxon para la evaluación de la técnica UML_2FP

Variable	Estudiantes PUCV	Estudiantes PUCP 2012-1	Estudiantes PUCP 2012-2
Observaciones	17	10	7
p-value (one tailed)	0.197	0.297	0.0785

Según los resultados presentados en la Tabla 5-6, el p-valor calculado es mayor que el nivel de significancia de 0.05 en las tres muestras; por lo tanto, nosotros no podemos rechazar la hipótesis nula Ho. Esto significa que nosotros no podemos empíricamente corroborar en estas muestras que podemos obtener resultados más precisos con Tupux que con los cálculos manuales.

Sin embargo, podemos observar que algunos participantes podrían obtener 42 PFSA, pero han identificado incorrectamente algunos ficheros. Por lo tanto, estos resultados deben tomarse con precaución y nos sugieren que debemos ejecutar más experimentos con diferentes casos de estudio.

5.2.3 Conclusiones de la Experimentación con UML_2FP

Se ha buscado evaluar el cálculo manual de PFSA utilizando UML_2FP versus el cálculo automático para medir cuál es más preciso.

En la presente investigación, se realizó un experimento con un grupo de 17 estudiantes de la PUCV. Sin embargo, para fortalecer los resultados, se decidió considerar además los resultados obtenidos de los experimentos realizados en la PUCP el año 2012 (Pow-Sang, Flores, & Villanueva, 2013).

En los resultados, se pudo notar que los alumnos pueden cometer error al momento de identificar los ficheros, causando que la cantidad de PFSA no sea la correcta. En el caso de la PUCV, el error en la clase de herencia “Empleado-Hijo” fue del 100% de los participantes.

Se comparó los resultados obtenidos con Tupux, basada en las técnicas que conforman Tupuy y que para estos experimentos, se utilizó para automatizar el cálculo de PFSA del caso de estudio.

Sin embargo, los PFSA que hubiesen obtenido los alumnos usando Tupux varían entre 42 y 49, cuando la cantidad de PFSA calculados de acuerdo a IFPUG es de 42.

Se utilizó la prueba estadística de Wilcoxon para evaluar las MRE obtenidas de cada grupo concluyendo finalmente que no se puede demostrar

empíricamente que Tupux pueda realizar cálculos más precisos que el método manual.

Por tanto se puede concluir que es necesario desarrollar nuevos casos de estudio e inducciones a mayor detalle para evitar los errores en la identificación de ficheros que a su vez podrían repercutir en el cálculo final de PFSA así como presentar mejores ejemplos que ayuden a identificar cada caso posible que puedan presentarse en un diagrama de clase.

Conclusiones

Existen diversas técnicas de estimación pero no todas son efectivas para tipos de proyecto en específico. Por esta razón, nació Tupuy, un conjunto de técnicas que apoyan a la estimación del esfuerzo requerido para el desarrollo de un proyecto de software. Está conformada por 3 técnicas: UML_2FP, UCPD e Incremental-FP

Tupuy está conformado por 3 técnicas: UML_2FP, para el cálculo de puntos de función a partir de un diagrama de clases de análisis; Use Case Precedence Diagram (UCPD) (Pow-Sang, Nakasone, Moreno, & Imbert, 2008), técnica basada en diagramas UML y que se utilizará para definir la secuencia de construcción de los casos de uso; e Incremental-FP, que utilizando los resultados de las dos primeras técnicas, definirá los incrementos y el esfuerzo necesario para cada uno de ellos.

Estas técnicas fueron validadas en la PUCP con estudiantes de posgrado y pregrado, así como profesionales de la industria mostrando resultados alentadores. Sin embargo, se consideró necesario validar estas técnicas en otro contexto, en este caso la PUCV en Chile.

Una replicación exitosa ayuda a construir confianza en nuestros resultados, y de no ser así, los resultados deben ser analizados para conocer las nuevas variables que aparecieron y que no nos permitieron realizar una réplica exitosa.

Esta idea motivó a realizar réplicas sobre los experimentos realizados previamente en la PUCP para las técnicas de UML_2FP y UCPD para validar los resultados obtenidos.

Para la evaluación de la técnica UCPD, se realizó una evaluación de efectividad frente a técnicas Ad hoc y una evaluación de la percepción de su uso por los participantes. Para evaluar la efectividad, se utilizaron los mismos casos de estudio, pero se aumentaron más preguntas en los cuestionarios para fortalecer la confiabilidad de los resultados. Para la evaluación de percepción de UCPD, se utilizó la última versión del cuestionario que fue utilizado en los experimentos previos.

De estos experimentos, se pudo comprobar y corroborar los resultados obtenidos en los previos experimentos: UCPD es más efectiva que las técnicas Ad hoc para la priorización de los casos de uso y así mismo, es considerada fácil de usar, útil y los participantes tienen intención de utilizarla en proyectos futuros. Por tanto podríamos concluir que la técnica UCPD puede ser aplicada en otros contextos.

Para la evaluación de la técnica UML_2FP, se realizó un experimento con 17 alumnos de la PUCV. Los resultados obtenidos se contrastaron con los obtenidos en los experimentos previos de la PUCP con el fin de evaluar si la herramienta Tupux realizaba el cálculo de PFSA de manera más precisa que el método manual. Sin embargo, no se pudo demostrar empíricamente que Tupux es más preciso para el cálculo de PFSA. Con estos resultamos, podemos concluir que UML2FP necesitaría más

experimentación con nuevos casos de estudio. Así mismo, Tupux es una herramienta que puede seguir mejorando para poder obtener resultados más precisos para el cálculo de ficheros.

Como limitaciones, tanto para la evaluación de UCPD como de UML_2FP podemos considerar que no se pudo contar con la participación de alumnos de posgrado debido a que se pudo obtener una población pequeña no significativa. Esto hubiese permitido obtener mayor conocimiento de la efectividad y percepción de las técnicas a un nivel más profesional, tal como se realizó en los experimentos previos.

En el caso de UML_2FP no se pudo contar con la participación de más alumnos, ni con la presentación más a detalle de la herramienta que pudo haber causado un mayor interés entre los alumnos en participar de las experimentaciones.

Como trabajo a futuro, podemos plantear que:

- La técnica Tupuy debería seguir validándose considerando algunas mejoras en los instrumentos de evaluación y otros enfoques como las metodologías ágiles, muy utilizadas en la actualidad. Esto permitirá mejorar la herramienta con el tiempo y hacerla adaptable a enfoques de desarrollo más generales.

- Así mismo, deberían existir más validaciones no sólo de las técnicas que conforman Tupuy de manera individual, sino en su totalidad, con proyectos reales en la industria del desarrollo de software.

- La herramienta Tupux necesita ser probada para mejorar el cálculo de Ficheros y ajustarse a los PFSA reales calculados con IFPUG así como también en los otros módulos que posee para la automatización de las técnicas que conforman Tupuy.

Bibliografía

Abrahão, S., Insfran, E., Carsí, J., & Genero, M. (2011). Evaluating requirements modeling methods based on user perceptions: A family of experiments. *Information Sciences: an International Journal, 181*(16), 3356-3378.

Atkinson, C., & Hummel, O. (2012). Iterative and incremental development of component-based software architectures. *Proceedings of the 15th ACM SIGSOFT symposium on Component Based Software Engineering*, 77-82.

Balbin, D., Ocrospoma, M., Soto, E., & Pow-Sang, J. A. (2009). TUPUX: An Estimation Tool for Incremental Software Development Projects. *Advanced Science and Technology, 2009. AST '09. International e-Conference on*, 39-43.

Basili, V., Shull, F., & Lanubile, F. (1999). Building Knowledge through Families of Experiments. *IEEE Transactions on Software Engineering, 25*(4), 456-473.

Boehm, B. W., Abts, C., Brown, A. W., Chulani, S., Clark, B. K., Horowitz, E., . . . Steece, B. (2009). *Software Cost Estimation with COCOMO II.* New York: Prentice Hall.

IBM Corporation. (s.f.). *IBM SPSS Software (Predictive analytics software and solutions)*. (IBM Corporation) Recuperado el 12 de 05 de 2013, de http://www.spss.com/

Internacional Function Point User Group. (2004). *Function Point Counting Practices Manual* (Vol. 4.2.1). New York: IFPUG.

ISO. (1998). ISO. ISO/IEC TR 15271 - Guide for ISO/IEC 12207 (Software Life).

Jacobson, I. (1992). *Object Oriented Software Engineering. A Use Case Driven.* Addison Wesley.

Juristo, N., & Gómez, O. (2012). Replication of software engineering experiments. En N. Juristo, O. S. Gómez, & S. B. Heidelberg (Ed.), *Empirical Software Engineering and Verification* (págs. 60-88). Elba Island: Springer.

Juristo, N., & Moreno, A. M. (2010). *Basics of Software Engineering Experimentation.* Springer Publishing Company.

Juristo, N., & Vegas, S. (2009). Using differences among replications of software engineering experiments to gain knowledge. *Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement (ESEM '09)*, 356-366.

Moody, D. (2001). *Dealing with Complexity: A Practical Method for Representing Large Entity Relationship Models.* Tesis Doctoral, University of Melbourne, Dept. Information Systems, Australia.

Ochodek, M., Nawrocki, J., & Kwarciak, K. (2011). Simplifying effort estimation based on Use Case Points. *Information and Software Technology, 53*(3), 200-213.

Perry, D., Porter, A., & Votta, L. (2000). Empirical Studies of Software Engineering: A Roadmap. *ICSE '00 Proceedings of the Conference on The Future of Software*, 345-355.

Pfleeger, S. (2002). *Ingeniería de Software: Teoría y Práctica.* Buenos Aires: Prentice Hall.

Plastic Software, Inc. (2008). *Star UML - The Open Source UML/MDA Platform*. (Plastic Software, Inc.) Obtenido de http://staruml.sourceforge.net/en/

Pow-Sang, J. A. (2012). *Técnicas para la estimación y planificación de proyectos de software con ciclos de vida incremental y paradigma orientado a objetos.* Tesis Doctoral, Universidad de Barcelona, Barcelona.

Pow-Sang, J. A., Nakasone, A., Moreno, A. M., & Imbert, R. (2008). An Approach to Determine Software Requirement Construction Sequences based on Use Cases. *Proceedings ASEA 2008, IEEE Computer Society*.

Pow-Sang, J., & Imbert, R. (2004). Estimación y Planificación de Proyectos Software con Ciclo de Vida Iterativo-Incremental y empleo de Casos de Uso. *Proceedings IDEAS 2004*.

Pow-Sang, J., & Imbert, R. (2007). Including the Composition Relationship among Classes to Improve Function Point Analysis. *Proceedings VI Jornadas Peruanas de Computación-JPC'07*.

Pow-Sang, J., & Jolay-Vázquez, E. (2006). An Approach of a Technique for Effort Estimation of Iterations in Software Projects. *APSEC '06 Proceedings of the XIII Asia Pacific Software Engineering Conference*, 367-376.

Pow-Sang, J., Flores, L., & Villanueva, D. (2013). A Conversion Model and a Tool to Identify Function Point Logic Files using UML Analysis Class Diagrams. *IWSM-MENSURA*.

Project Management Institute. (2013). *A Guide to the Project Management Body of Knowledge: Pmbok guide* (Quinta ed.).

Rosenberg, D., & Stephens, M. (2007). *Use Case Driven Object Modeling with UML.* Apress.

Royce, W. (1987). Managing the Development of Large Software Systems: Concepts and Techniques. *ICSE '87 Proceedings of the 9th international conference on Software* , 328-338.

Shepperd, M., Schofield, C., & Kitchenham, B. (1996). Effort Estimation Using Analogy. *ICSE '96 Proceedings of the 18th international conference on Software engineering*, 170-178.

Shull, F., Carver, J., Vegas, S., & Juristo, N. (2008). The role of replications in Empirical Software Engineering. *Empirical Software Engineering, 13*(2), 211-218.

Tichy, W. (1998). Should Computer Scientists Experiment More? *Computer, 31*(5), 32-40.