PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

ESCUELA DE POSGRADO

Aplicación Práctica de Técnicas para la Estimación y Planificación de Proyectos de Software – TUPUY

Tesis para optar el grado de Magíster en Informática que presenta

Magaly INCA CHIROQUE

Dirigido por

DR. José Antonio POW SANG PORTILLO

Jurado

Mg. Claudia ZAPATA DEL RÍO

Dr. Andrés MELGAR SASIETA

Dr. José Antonio POW SANG PORTILLO

LIMA-PERÚ

2015

RESUMEN

El presente trabajo tiene por finalidad determinar la confiabilidad de la técnica Tupuy propuesta por el Dr. Pow Sang (Pow Sang Portillo, 2012), que mide el esfuerzo de un proyecto de software orientado a objetos con ciclo de vida incremental en su desarrollo, para efectos de planificación. Esta propuesta está conformada por tres técnicas: UML2FP, Diagrama de Precedencia de Casos de Uso (UCPD) e Incremental-FP. La aplicación práctica de la técnica Tupuy se realizó sobre base histórica obtenida de los alumnos de pregrado de la especialidad de Ingeniería Informática, quienes desarrollaron un proyecto de software de un sistema de información para una cadena de hoteles que recién iniciaba su funcionamiento. De los resultados obtenidos se confirma lo propuesto en la tesis del Dr. Pow Sang, al comparar los resultados obtenidos con pruebas previas, la diferencia entre el esfuerzo estimado y real de los proyectos, medidos con la Magnitud del Error Relativo (MRE) para todas las iteraciones, fueron menores al 20%. Por lo tanto, se puede utilizar la técnica Tupuy con toda confianza para fines de planificación.

PALABRAS CLAVE: Estimación de esfuerzo, Puntos de Función, Desarrollo Incremental, Proyectos de Software, Técnicas de Estimación.

- A Dios, por todas las bendiciones que me da cada día.
- A mi madre, Lidia Chiroque Silva, por su entrega y amor incondicional.
- A mi padre, Florentino Inca Heredia, por su ejemplo de perseverancia y disciplina.
- A mis hermanas, Julita y Marylú Inca Chiroque, por su comprensión, amor y enseñanzas en todo momento.
- A Ténesis Alfaro Meza, por su comprensión y apoyo constante.
- A mi asesor Dr. José Antonio Pow Sang Portillo, y a los miembros del jurado Mg. Claudia Zapata del Río y Dr. Andrés Melgar Sasieta, por su paciencia y recomendaciones impartidas.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

A pesar que la industria de software ha experimentado un gran crecimiento y expansión desde su nacimiento, hoy en día continua enfrentando problemas en su evolución, siendo su gran reto la correcta estimación del esfuerzo y por lo tanto de la estimación del costo de software.

Debido a la naturaleza dinámica del desarrollo de software, se hace más difícil obtener una estimación correcta del esfuerzo y costos del mismo, factor importante que hace al software más competitivo y es esencial para el control de los costos de desarrollo de software. La estimación de costos de software es una de las actividades más difícil de la gestión de proyectos, porque los valores de muchas de las variables no se conocen y no son fáciles de predecir en una etapa temprana de desarrollo de software. Un modelo ideal de estimación de costos de software debe proporcionar confianza, precisión y exactitud de sus predicciones, ya que puede afectar considerablemente la planificación y programación de un proyecto de software.

Por otro lado, se están desarrollando en los últimos años, diferentes métricas, técnicas de estimación, modelos y procesos de desarrollo en la ingeniería de software. El

proceso tradicional de cascada está perdiendo vigencia dado a que se presenta como un flujo lento y riguroso en la producción de software. Para reducir los riesgos de desarrollo y entrega a tiempo del producto se están empleando modelos de desarrollo Iterativo-Incremental (Siglas en inglés IID), métodos algorítmicos y no algorítmicos para estimar el esfuerzo de desarrollo en las primeras etapas del proyecto tales como Puntos de Función (Siglas en inglés PF), Puntos de Casos de Uso (Siglas en inglés UCP), el Modelo constructivo de Costes II - COCOMO II, entre otros.

El presente trabajo consiste en la aplicación práctica de Tupuy, el cual es un conjunto de técnicas que apoyan 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. Esta propuesta Tupuy está conformada por tres técnicas: UML2FP, Diagrama de Precedencia de Casos de uso (Siglas en inglés UCPD) e Incremental-FP. El presente trabajo consta de cuatro capítulos:

En el primer capítulo se describe la problemática que afronta un administrador de proyectos para determinar el esfuerzo de desarrollo de proyectos de software, se plantea el objetivo, los resultados esperados, la justificación, la metodología y los alcances para la elaboración del trabajo de tesis.

En el segundo capítulo se presenta un marco de referencia para contextualizar el problema de investigación mediante una perspectiva teórica de métricas, métodos, procedimientos y modelos de desarrollo de proyectos de software, como el desarrollo Incremental.

En el tercer capítulo se detallan estudios previos de las implementaciones de métodos que permiten estimar el esfuerzo entre los incrementos o iteraciones en el desarrollo de software en áreas de la industria en los últimos años.

Finalmente el cuarto capítulo comprende la aplicación práctica de la técnica Tupuy, sobre datos históricos de tres sistemas de hoteles recolectados de estudiantes del programa de Ingeniería informática en la Pontificia Universidad Católica del Perú (PUCP), matriculados en el curso de Ingeniería de Software en el año 2009.

Sabiendo que la estimación de los costos de desarrollo de software es una de las actividades más desafiantes en la gestión de proyectos de software, con el presente trabajo se pretende aportar evidencia empírica de una técnica confiable de estimación del esfuerzo de desarrollo de software para propósitos de planificación.

CAPITULO I: PRESENTACIÓN DEL PROYECTO

1.1. Definición del problema

Algunas actividades de la gestión de proyectos de software están relacionadas a los plazos de entrega del producto, manejo de habilidades del equipo, definición de la visión y objetivos, comunicación con los clientes y usuarios, quienes muchas veces no están involucrados en el proyecto como debería ser por cuestiones de tiempo y horarios (Mishra J. , 2011). De no se gestionarse adecuadamente estas actividades, originarían en muchas ocasiones que un proyecto de software falle, teniendo un impacto en la organización en el ámbito financiero, en la credibilidad de la organización con los clientes, en su posición en el mercado y en su ventaja competitiva (Chemuturi, 2010).

Frente a esta situación, en Ingeniería de Software se ha adoptado en el proceso de construcción y desarrollo de software algunas metodologías más utilizadas como el Proceso de Desarrollo Unificado (Siglas en inglés RUP) que utiliza la notación de Lenguaje Unificado de Modelado (Siglas en inglés UML) (Pender, 2003) para sistemas orientados a objetos, metodologías ágiles como *Scrum* (Dimes, 2015), así como estándares de gestión de proyectos como la Oficina de Gestión de Proyectos (Siglas en inglés PMO) (Karkukly, 2012) y organizaciones como el Instituto de Administración de Proyectos (Siglas en inglés PMI) (Sater Carstens, 2013), que persiguen la estimación de indicadores para predecir recursos, costos y tiempos durante la etapa de planificación de proyectos.

Desde la década del 1950 que fue donde se difundió el concepto de gestión de proyectos y hasta nuestros días, el mundo está cambiando continuamente y la industria del software se encuentra en esta tendencia. El rápido crecimiento de la tecnología en combinación con la dependencia de productos y servicios en software aumenta la demanda de proyectos de desarrollo de software seguros y confiables (Ruhe, 2014).

Para desarrollar el software que los clientes necesitan se recomienda el uso de modelos de desarrollo de software para agilizar su proceso de desarrollo. Algunos de estos modelos son el de cascada, espiral, desarrollo iterativo e incremental y el desarrollo ágil (Roebuck, 2012).

El modelo de cascada de un ciclo de vida de software, donde el desarrollador debe seguir las fases de especificación de requerimientos, diseño, integración, pruebas, instalación y mantenimiento del software en ese mismo orden y sin poder revisar una fase previa una vez se haya completado (Roebuck, 2012), ha quedado prácticamente obsoleto, por lo que en el presente trabajo se tendrá en cuenta el ciclo de vida incremental en desarrollo de proyectos de software, que permite la construcción de pequeñas porciones (incrementos) de un proyecto de software para ayudar a los involucrados a descubrir temas importantes en forma temprana antes de los problemas o suposiciones erróneas que puedan llevar a un desastre (Roebuck, 2012) (Tan, 2009).

Para realizar la estimación del esfuerzo en proyectos de software se utilizan técnicas tales como Puntos de función (Siglas en inglés FP), que mide el software cuantificando la funcionalidad que éste proporciona al usuario basándose en el diseño lógico (IFPUG: International Function Point User Group, 2009), y Puntos de Casos de Uso (Siglas en inglés UCP), el cual es una extensión de los puntos de función y se basa en especificaciones de casos de uso (Awan N. M., 2010), para proyectos que siguen un modelo de ciclo de vida en cascada. Sin embargo, aún no se ha definido indicadores con alta precisión para la estimación del esfuerzo de desarrollo de software en el ciclo de vida incremental de un proyecto y en el desarrollo orientado a objetos (Pow Sang, 2006).

En el 2012, el Dr. Pow Sang Portillo (Pow Sang Portillo, 2012) propuso una técnica denominada Tupuy, la cual estima el esfuerzo de desarrollo temprano en el ciclo de vida de un software, se compone de tres técnicas: UML2FP, Diagrama de Precedencia de Casos de uso (Siglas en inglés UCPD) e Incremental-FP. La técnica UML2FP permite el cálculo de los puntos de función basado en modelos orientados a objetos, utilizando para ello el diagrama de clases de análisis (Pow Sang Portillo, 2012).La técnica UCPD (Diagrama de precedencia de casos de uso), ayuda al programador a definir visualmente el orden de construcción de los casos de uso y, por tanto inferir cuáles de ellos deben ser incluidos en cada incremento (Balbin D. S., 2009). Finalmente, la técnica Incremental – FP se determina en base a los resultados obtenidos al emplear las técnicas UML2FP y UCPD, la cual comprende la ejecución de cuatro actividades que permiten definir los incrementos a construir, qué casos de uso se van a desarrollar en cada incremento y estimar el esfuerzo que se requiere para concluir cada incremento (Pow Sang Portillo, 2012).

Tupuy compara el esfuerzo real contra el esfuerzo estimado de un proyecto de desarrollo de software, utilizando la medida denominada Magnitud del Error Relativo (Siglas en inglés MRE) que se utiliza para evaluar la eficiencia de las estimaciones (Heričko M. , 2008).Por lo tanto, siendo Tupuy una técnica relativamente nueva en la industria de software, es necesaria su comprobación debido a que la fiabilidad de un indicador para la ingeniería de software es crucial en el grado en que un experimento, prueba, o cualquier procedimiento de medición arroja los mismos resultados en pruebas repetidas (Carmines E. G., 1979).

Una buena estimación del esfuerzo es uno de los soportes del administrador del proyecto para lograr una gestión exitosa y finalización del proyecto de desarrollo de software. La práctica cotidiana demuestra que muchas organizaciones de software aún proponen costos de software poco realistas, calendarios de trabajo apretados, terminando sus proyectos excediéndose de calendarios y presupuestos, o no lo completan en absoluto. Por lo tanto, un buen método de estimación es el que proporciona ese apoyo (Trendowicz, 2014).

En el presente trabajo se comprobará la técnica Tupuy, teniendo en cuenta los márgenes de error en las estimaciones obtenidas. En el caso que se tenga un margen de error menor que 20% se puede concluir que la técnica puede ser considerada predictiva y ser empleada con confianza para la planificación (Hastings T. S., 2001).

1.2. Objetivo General

Determinar la confiabilidad de la técnica Tupuy, que mide el esfuerzo de un proyecto de software orientado a objetos con ciclo de vida incremental en su desarrollo; para efectos de planificación, al ser aplicado en datos recolectados de estudiantes de la Pontificia Universidad Católica del Perú.

1.3. Objetivos Específicos

OE1 Determinar el esfuerzo estimado en proyectos de software con ciclo de vida incremental y paradigma orientado a objetos, aplicando la técnica Tupuy.

OE2 Analizar si el margen de error relativo (Siglas en inglés MRE) es menor al 20% en cada incremento.

OE3 Comprobar la fiabilidad de la técnica Tupuy al compararla en pruebas previas.

1.4. Resultados Esperados

OE1 Determinar el esfuerzo estimado en proyectos de software con ciclo de vida incremental y paradigma orientado a objetos, aplicando la técnica Tupuy.

R1 Obtener el esfuerzo estimado base para aplicar el margen de error relativo.

OE2 Analizar si el margen de error relativo (MRE) es menor al 20% en cada incremento.

R2 Confirmar que la técnica sea considerada predictiva y pueda ser empleada con confianza para la planificación.

OE3 Comprobar la fiabilidad de la técnica Tupuy al compararla en pruebas previas.

R3 Confirmar los resultados presentados en la tesis del Dr. Pow Sang.

1.5. Metodología del proyecto

En el presente trabajo se aplicará el enfoque cuantitativo, debido a que la técnica Tupuy es aplicada en datos numéricos recolectados de alumnos matriculados en el curso de Ingeniería de Software en el año 2009 en la Pontificia Universidad Católica del Perú. Estos proyectos fueron desarrollados para implementar un sistema de información en diferentes cadenas de hoteles. La duración del proyecto fue de 14 semanas, que es la duración de un semestre académico de la PUCP. Los alumnos desarrollaron los sistemas de hoteles: Hotel 1, Hotel 2 y Hotel 3 de la siguiente manera:

- Divididos en grupos de 11 a 12 personas, los alumnos registraron en una hoja de cálculo, las horas utilizadas para realizar cada actividad por día de la semana y el grado de avance en porcentaje de los casos de uso.
- Todos los equipos realizaron tres iteraciones en la fase de construcción, es decir; tres incrementos, cada uno de los cuales duraba aproximadamente dos semanas.
- Para el modelamiento de los casos de uso utilizaron como herramienta StarUML.
- Elaboraron los siguientes entregables: Plan de Proyecto, Documento de visión y Especificación de Requisitos de software.

Con los entregables elaborados por los alumnos (Plan de Proyecto, Diagrama de clase, Diagrama de Precedencias, Especificación de Requisitos de software, Formato de horas) se aplicó la técnica Tupuy de la siguiente manera:

1) En base al diagrama de clases de análisis de UML se aplicaron las reglas para determinar los Puntos de Función Sin Ajustar (Siglas en inglés PFSA) utilizando el componente adicional Tupux (Balbin D. M.-S., 2009) que facilita la planificación y estimación utilizando puntos de función en los proyectos de software que adoptan el modelo incremental.
2) En base al diagrama de precedencias se aplicó la técnica UCPD, para determinar los casos de uso por cada incremento.
3) Con los resultados obtenidos de aplicar las técnicas UML2FP y UCPD se aplicó la técnica Incremental-FP.
4) Una vez determinado el esfuerzo estimado (utilizando la técnica Tupuy), y en base al esfuerzo real (horas utilizadas registradas en las hojas de cálculo), se aplica la Magnitud del error relativo (Siglas en inglés MRE) para cada incremento.

La Magnitud del error relativo (Siglas en inglés MRE) es una medida normalizada de la discrepancia entre los valores reales y estimados más utilizada en la literatura (Awan N. M., 2010) (Lo, 2007). Los siguientes márgenes de error se pueden utilizar para propósitos de planificación (Hastings T. S., 2001):

- MRE ≤ 20% indica que la técnica puede ser considerada predictiva y puede ser empleada con confianza para la planificación;
- 20% < MRE ≤ 50% indica que es aceptable, pero debe ser empleada con precaución;
- MRE > 50% indica que no es confiable para propósitos de planificación.

5) Finalmente, se analizan los resultados obtenidos de calcular el MRE para cada incremento por hotel, para confirmar la confiabilidad de la técnica Tupuy.

Es importante mencionar que en el presente trabajo no se ha involucrado con los participantes, en este caso los alumnos, y lo que se pretende es generalizar los resultados encontrados para ser aplicados con confianza en equipos de desarrollo de proyectos de software para efectos de planificación.

1.6. Justificación

Se justifica el presente trabajo debido a que muchas organizaciones de software aún proponen costos de software poco realistas, calendarios de trabajo apretados, terminando sus proyectos excediéndose de calendarios y presupuestos, o no lo completan en absoluto, siendo necesario un buen método de estimación que proporciona ese apoyo (Trendowicz, 2014).

Debido a que ningún método es el mejor para todos los proyectos y con la finalidad de obtener resultados fiables de estimación, en la tesis del Dr. Pow Sang se propuso un enfoque denominado Tupuy, el cual 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. Esta propuesta está conformada por tres técnicas: UML2FP, Diagrama de Precedencia de Casos de uso (Siglas en inglés UCPD) e Incremental-FP (Pow Sang Portillo, 2012).

En el presente trabajo de tesis se evaluará si la técnica Tupuy es confiable para estimar el esfuerzo en proyectos de desarrollo de software, al comparar el esfuerzo real contra los esfuerzos estimados, teniendo en cuenta los márgenes de error durante la planificación de un proyecto. Con ello, se apoya lo planteado en la tesis del Dr. Pow Sang Portillo al confirmar que la técnica es confiable para la planificación de proyectos de desarrollo de software si arroja los mismos resultados.

1.7. Alcance del Proyecto

El alcance del presente trabajo abarcará el estudio descriptivo, correlacional y explicativo.
Con el estudio descriptivo se definirá las técnicas que conforman la técnica Tupuy: UML2FP, Diagrama de Precedencias de casos de uso (UCPD) e Incremental-FP.

Con el estudio correlacional se mostrará como los cambios en el contexto de un proyecto, tales como la experiencia en el desarrollo de aplicaciones o la experiencia en la herramienta de programación y del lenguaje, afecta el resultado del esfuerzo estimado.

Finalmente, en base al estudio explicativo se responderá si la técnica Tupuy es confiable para efectos de planificación, para ello se aplicará la técnica sobre datos

históricos obtenidos de un trabajo de curso de alumnos de pregrado de la especialidad de Ingeniería Informática matriculados en el curso de Ingeniería de Software del año 2009 en la Pontificia Universidad Católica del Perú, durante 14 semanas.

Las etapas consideradas en la aplicación de la técnica Tupuy son las etapas de diseño detallado, construcción de software, programación y pruebas, no considerará el esfuerzo que se genera por las reuniones internas con los *stakesholders* para el seguimiento o control del proyecto.

CAPITULO II: MARCO CONCEPTUAL

El presente trabajo se enmarca dentro de lo que se conoce como Gestión de Proyectos de Desarrollo de Software, mediante la aplicación práctica de la técnica Tupuy (Pow Sang Portillo, 2012), conformada por las técnicas: UML2FP, UCPD e Incremental-FP.

En este capítulo, se presenta un marco de referencia para contextualizar el problema de investigación mediante una perspectiva teórica de los conceptos básicos utilizados en la estimación del esfuerzo en proyectos de desarrollo de software así como en la técnica Tupuy propuesta por el Dr. Pow sang. Estos conceptos incluyen lo siguiente:

Los modelos de ciclo de vida, detallando el Modelo Incremental; que es el modelo de ciclo de vida donde es conveniente la aplicación de la técnica Tupuy.

También se detallan los conceptos de los diagramas en el Lenguaje Unificado de Modelado (Siglas en inglés UML) como el Diagrama de Casos de Uso y el Diagrama de Clases de Análisis, Puntos de Función, el Factor del Ajuste del Esfuerzo (Siglas en inglés EAF) y el Modelo de Costo Constructivo (Siglas en inglés COCOMO).

Finalmente se detallan, los indicadores de medición para evaluar la eficiencia de las estimaciones como es la Magnitud del de Error Relativo (Siglas en inglés MRE).

2.1. Modelos de Ciclo de Vida

Ruparelia (Ruparelia, 2010) ofrece un recorrido sobre los modelos de ciclo de vida de desarrollo de software, definiendo los siguientes conceptos:

- Un ciclo de vida cubre todas las etapas de software desde su inicio con la definición de los requisitos hasta su mantenimiento.
- Un Modelo de Ciclo de Vida de Desarrollo de Software (Siglas en inglés SDLC) es un marco conceptual o proceso que describe la estructura de las etapas involucradas en el desarrollo de una aplicación a partir de su estudio de viabilidad inicial hasta su implementación en el campo y mantenimiento.

Esta tesis se basa en la teoría de (Sommerville, 2005), que estudia el modelo de desarrollo de software en cascada, y realiza un enfoque del desarrollo incremental de un software.

2.1.1 El Modelo en Cascada

Se le conoce como Modelo en Cascada o como Ciclo de vida del software, debido a la cascada de una fase a otra. La fase siguiente no debe empezar hasta que la fase previa haya finalizado. Presenta las siguientes actividades fundamentales de desarrollo: Análisis y definición de requerimientos, diseño del sistema y del software, implementación y prueba de unidades, integración y prueba del sistema, funcionamiento y mantenimiento (Sommerville, 2005).

2.1.2 Desarrollo Incremental

La entrega incremental es un enfoque intermedio que combina la ventaja de los modelos del desarrollo en cascada y del desarrollo evolutivo. En un proceso de desarrollo incremental, los clientes identifican, a grandes rasgos, los servicios que proporcionará el sistema. Identifican qué servicios son más importantes y cuáles menos. Entonces, se definen varios incrementos en donde cada uno proporciona un subconjunto de la funcionalidad del sistema.

Una vez que los incrementos del sistema se han identificado, los requerimientos para los servicios que se van a entregar en el primer incremento se definen en detalle, y éste se desarrolla. Durante el desarrollo, se puede llevar a cabo un análisis adicional de requerimiento para los requerimientos posteriores, pero no se aceptan cambios en los requerimientos para el incremento actual.

Una vez que un incremento se completa y entrega, los clientes pueden ponerlo en servicio. Esto significa que tienen una entrega temprana de parte de la funcionalidad del sistema. Pueden experimentar con el sistema, lo cual les ayuda a clarificar sus requerimientos para los incrementos posteriores y para las últimas versiones del incremento actual. Tan pronto como se completan los nuevos incrementos, se integran en los existentes de tal forma que la funcionalidad del sistema mejora con cada incremento entregado. Los servicios comunes se pueden implementar al inicio del proceso o de forma incremental tan pronto como sean requeridos por un incremento (Sommerville, 2005).

2.2 Especificación de requisitos con casos de uso

Los Casos de Uso fueron introducidos en 1987 como una herramienta de la técnica *Objectory*. Su utilización en los procesos de Ingeniería de Software fue propuesta por Ivar Jacobson y publicada en su libro “Object Oriented Software Engineering”

(Magnus, 1992). Se representan en un modelado del diseño de aplicaciones del software antes de la codificación. (OMG Unified Modeling Language, 2008).

Por otro lado, el Lenguaje Unificado de Modelado (Siglas en inglés UML) se ha convertido en el estándar para el modelado orientado a objetos de software y ha sido adoptado por empresas de todo el mundo (Pender, 2003).

2.2.1 Diagramas de casos de uso

Los casos de uso representan la interacción que el sistema realiza para sus actores, tienen un nombre y una breve descripción acerca de cómo los actores usan el sistema para llevar a cabo algún proceso que consideran importante, y qué hace el sistema para satisfacer estas necesidades (Kurt Bittner, 2003).

Los actores y casos de uso se representan en el diagrama de casos de uso, los actores están representados por monigotes y los casos de uso por óvalos. Las flechas (que representan las relaciones), conectan a los actores y los casos de uso que interactúan (Kurt Bittner, 2003).

Las relaciones principales entre los casos de uso son: Inclusión (*Include*) donde un caso de uso fuente incorpora el comportamiento de un caso de uso de destino, Extensión (*Extend*) donde un caso de uso fuente puede incorporar el comportamiento de un caso de uso de destino, es decir el caso de uso destino puede darse sin alterar en caso de uso fuente y Generalización donde los comportamientos, contenidos o propiedades de un caso de uso se heredan a otros, siendo los tipos de generalización entre casos de uso o entre los actores. Los casos de uso de destino deben suceder de forma obligatoria antes del caso de uso fuente (Knoernschild, 2002).

En la figura 2-1 se muestra los casos de uso que corresponden a la Administración de Clientes: Mantenimiento de Clientes, Buscar Clientes, Mantenimiento del Tipo de Cliente. El caso de uso Mantenimiento de Clientes (caso de uso fuente) tiene una relación de inclusión con el caso de uso Buscar Clientes (Caso de uso destino).

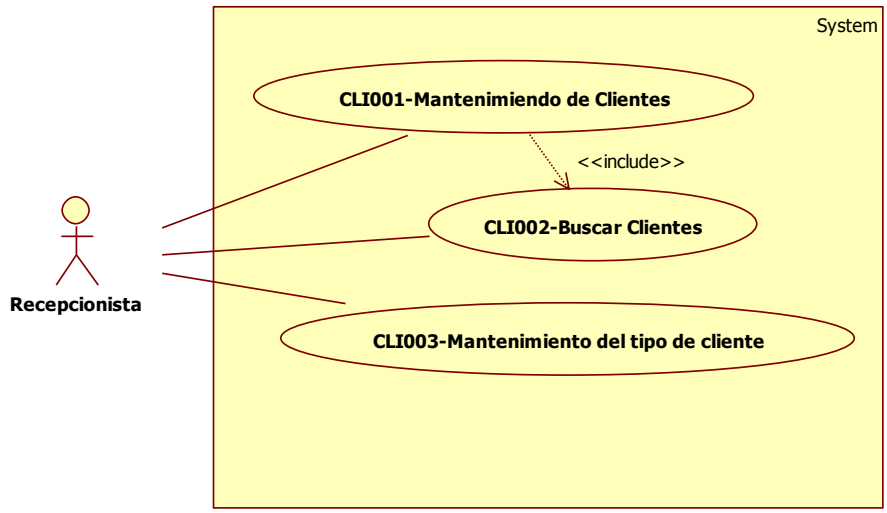

Figura 2-1: Diagrama de Casos de uso del sub paquete de Administración de Clientes elaborado por los alumnos

2.3 Diagramas de clases de análisis

Según (Uemura, 2001) los diagramas de clases describen la estructura estática del modelo, es decir, los objetos, las clases, y las relaciones entre estas entidades, incluyendo la generalización y agregación. También representan los atributos y operaciones de la clase.

El diagrama de clases se encuentra en el centro del proceso de modelado de objetos, y es el diagrama principal para la captura de todas las reglas que rigen la definición y el uso de objetos (Pender, 2003).

Se distinguen dos tipos de relaciones: Generalizaciones que representan la herencia en las jerarquías de clase y Asociaciones como agregaciones y composiciones. En los diagramas de clases UML, las generalizaciones se dibujan normalmente estrictamente mostrando jerárquicamente la estructura de herencia de las clases, las asociaciones son los elementos no-jerárquicos del esquema (Gutwenger, 2003).

La figura 2-2 muestra una asociación por composición, en la cual la clase universidad está compuesta por las clases facultades y especialidad, si se destruye la clase facultad se destruye la clase especialidad, debido a que el tiempo de vida de la clase especialidad depende de la clase facultad. Por otro lado, la clase estudiante forma parte de la clase universidad, si se destruye la clase universidad no afecta a la clase

estudiante por que los estudiantes pueden irse a otra universidad, en este caso es una asociación por agregación.

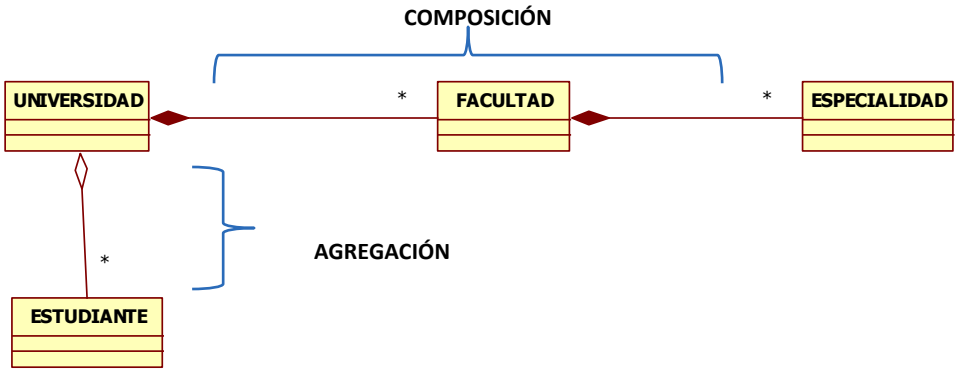

Figura 2-2: Ejemplo Agregación y Composición

La figura 2-3 muestra un ejemplo de generalización, la superclase (clase principal) hereda atributos a las subclases. Los atributos de la clase empleado (nombre, apellido, dirección y teléfono) van a ser directamente heredados a la clase ingeniero y a la clase contador.

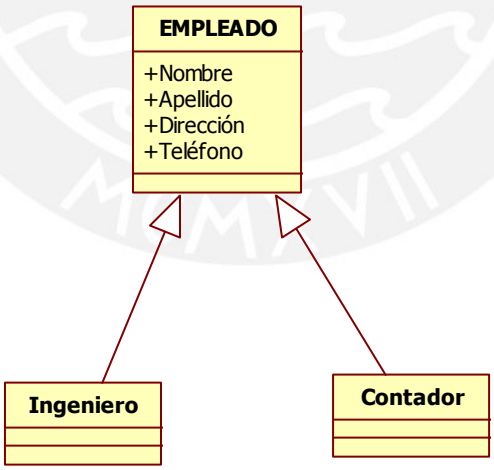

Figura 2-3: Ejemplo de Generalización

2.4 Puntos de Función

Un Método de Medición de Tamaño Funcional (Siglas en inglés FSM) mide la vista externa lógica del software desde la perspectiva de los usuarios mediante la evaluación de la cantidad de funcionalidad para ser entregado. El método FSM más utilizado es el Análisis de Puntos de Función (Siglas en inglés FPA) basada en el método propuesto por Alan Albrecht (Albrecht, 1979). Esta técnica fue desarrollada específicamente para medir la cantidad de datos que accede a cada función como un indicador de tamaño funcional. Sin embargo, supone el uso de metodologías de desarrollo de software tradicionales tales como el análisis y diseño estructurado (Abrahão, 2006). Se puede determinar a partir de la especificación de requisitos, de la especificación de diseño, o desde el código de programa. Debe ser independiente de la tecnología y el lenguaje utilizado para diseñar e implementar el software y los cambios en el software (Uemura, 2001).

A continuación se definen los conceptos y medición de los Puntos de Función teniendo como base el Manual de Prácticas de Medición de Puntos de Función (IFPUG: International Function Point User Group, 2009).

2.4.1 Puntos de Función no Ajustados

Los Puntos de Función No Ajustados (Siglas en inglés UFPC) refleja la funcionalidad específica medible proporcionada al usuario por el proyecto o aplicación. La funcionalidad de usuario específica de la aplicación es evaluada en términos de qué es lo que se entrega con la aplicación, no de cómo se entrega. Solamente se miden los componentes solicitados y definidos por el usuario.
La medición de puntos de función no ajustados tiene dos tipos de función: de datos y transaccional.

2.4.2 Funciones de Datos

2.4.2.1 Definiciones

Las funciones de datos representan la funcionalidad proporcionada al usuario para satisfacer los requisitos de datos internos y externos. Los tipos de funciones de datos se definen como ficheros lógicos internos (Siglas en inglés ILFs) y ficheros de interfaz externos (Siglas en inglés EIFs).

- **Ficheros Lógicos Internos (ILFs)**

Un fichero lógico interno (Siglas en inglés ILF) es un grupo de datos relacionados lógicamente o información de control, identificable por el usuario, mantenido dentro de los límites de la aplicación. El propósito principal de un ILF es almacenar datos mantenidos a través de uno o más procesos elementales de la aplicación que se está midiendo.

- **Ficheros de Interfaz Externos (EIFs)**

Un fichero de interfaz externo (Siglas en inglés EIF) es un grupo de datos relacionados lógicamente o información de control, identificable por el usuario, referenciado por la aplicación, pero mantenido dentro de los límites de otra aplicación. El propósito principal de un EIF es almacenar datos referenciados por uno o más procesos elementales dentro de los límites de la aplicación medida. Esto significa que un EIF medido para una aplicación debe ser un ILF en otra aplicación.

Diferencia entre ILFs y EIFs

La diferencia principal entre un fichero lógico interno y un fichero de interfaz externo es que un EIF no está mantenido por la aplicación que se está midiendo, mientras que un ILF sí lo está.

En la figura 2-4 se muestra que al ser la información de empleados mantenido en la aplicación Recursos Humanos es un Fichero Lógico Interno (ILF), mientras que la información de la tasa de conversión, mantenida por la Aplicación de Divisas, al ser utilizada por la Aplicación de Recursos Humanos es un Fichero de Interfaz Externa (EIF).

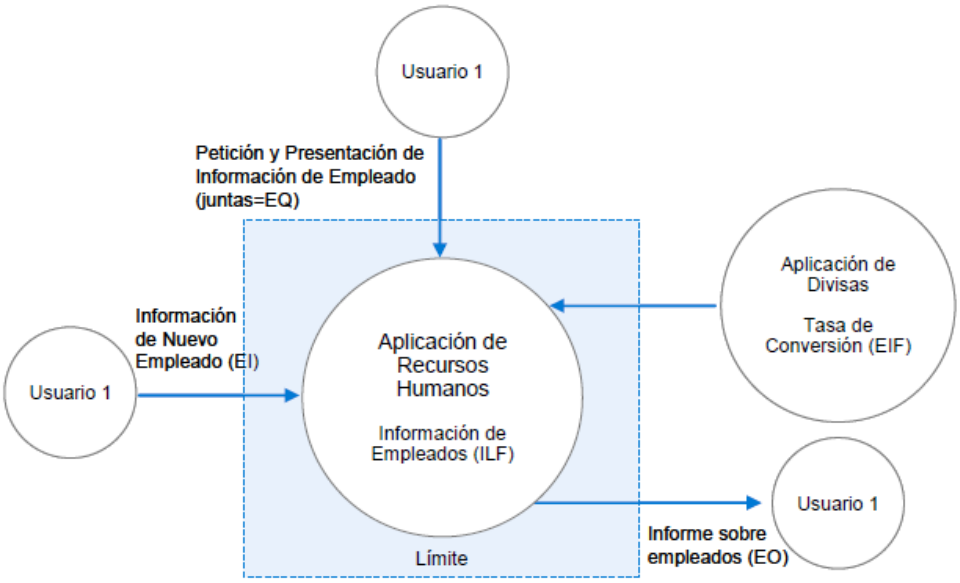

Figura 2-4: Diagrama Ejemplo diferencia entre ILFs y EIFs

2.4.2.2 Procedimientos de medición de ILFs/EIFs

1. Identificar Ficheros Lógicos internos.
2. Identificar Ficheros de Interfaz Externos.
3. Determinar Complejidad y Contribución

2.4.2.3 Procedimiento de Complejidad y Contribución

1.- Identificar y contar el número de DETs y RETs.

Definición de Tipo de Dato Elemental (DET)

Un tipo de elemento de datos (Siglas en inglés DET) es un campo único, no repetido, reconocible por el usuario.

En la figura 2-4 el fichero Información de Empleados (ILF) que debe ser mantenido por el usuario de RRHH incluye: Código de empleado, Nombre del empleado, Dirección, Nivel Salarial y Categoría del Empleado.Para determinar los DETs se debe contar cada campo único es este caso serían 5 DETs.

Definición de Tipo de Elementos de Registro (RET)

Un Tipo de Elemento de Registro (Siglas en inglés RET) es un subgrupo de elementos de datos, reconocible por el usuario, dentro de un ILF o EIF. Para mayor detalle de las

reglas del conteo de DETs y RETs veáse el Manual de prácticas de medición de puntos de función (IFPUG: International Function Point User Group, 2009).

En la figura 2-4 no hay sub grupos por lo tanto se debe contar un RET para el ILF de empleado en la aplicación de RR.HH.

2.- Valorar la complejidad funcional utilizando la siguiente matriz de complejidad.

	1 a 19 DET	20 a 50 DET	51 o más DET
1 RET	Baja	Baja	Media
2 a 5 RET	Baja	Media	Alta
6 o más RET	Media	Alta	Alta

Tabla 2-1 : Matriz de complejidad (IFPUG: International Function Point User Group, 2009)

En la figura 2-4 en el fichero Información de Empleados (ILF) se cuentan 5 DETs como campos únicos y un RET . Por lo tanto deacuerdo a la tabla 2-1 tiene complejidad Baja.

3.- Traducir los ILFs y EIFs a puntos función no ajustados utilizando la tabla de conversión apropiada para ILFs o EIFs.

- **Tabla de Conversión de ILFs**: Utilice la siguiente tabla para convertir los ILFs en puntos de función no ajustados.

Valorización Complejidad Funcional	Puntos Función no Ajustados
Baja	7
Media	10
Alta	15

Tabla 2-2: Conversión de ILFs en puntos función no ajustados (IFPUG: International Function Point User Group, 2009)

Para determinar los puntos de función no ajustados de la información de empleados (ILF) que tiene complejidad baja deacuerdo a la tabla 2-2 serían 7.

- **Tabla de Conversión de EIFs**: Utilice la siguiente tabla para convertir los EIFs en puntos de función no ajustados.

Valorización Complejidad Funcional	Puntos Función no Ajustados
Baja	5
Media	7
Alta	10

Tabla 2-3 : Conversión de EIFs en puntos función no ajustados (IFPUG: International Function Point User Group, 2009)

2.4.3 Funciones Transaccionales

2.4.3.1 Definiciones

Las funciones transaccionales representan la funcionalidad proporcionada al usuario para el procesamiento de los datos a través de una aplicación. Estas funciones son: Entradas Externas (Siglas en Inglés EIs), Salidas Externas (Siglas en Inglés EOs) y Consultas Externas (Siglas en Inglés EQs).

- **Entradas Externas (EI)**

Una entrada externa (EI) es un proceso elemental que procesa datos o información de control que provienen de fuera de los límites de la aplicación. El propósito principal de una EI es mantener uno o más ILFs y/o modificar el comportamiento del sistema.

En la figura 2-4 se muestra el proceso de introducir información de empleados en la Aplicación de Recursos Humanos.

- **Salidas Externas (EO)**

Una salida externa (EO) es un proceso elemental que envía datos o información de control fuera de los límites de la aplicación. El propósito principal de una salida externa es presentar información al usuario mediante una lógica de proceso diferente o adicional al de la recuperación de datos o información de control. La lógica de proceso debe contener al menos una fórmula matemática o un cálculo, crear datos derivados, mantener uno o más ILFs o alterar el comportamiento del sistema.
En la figura 2-4 se muestra el proceso de producir un informe con la lista de todos los empleados almacenados en la Aplicación de Recursos Humanos.

- **Consultas Externas (EQ)**

Una consulta externa (EQ) es un proceso elemental que envía datos o información de control fuera de los límites de la aplicación. El propósito principal de una consulta externa es presentar información al usuario a través de la recuperación de datos o información de control de un ILF o EIF. La lógica de proceso no contiene fórmulas matemáticas ni cálculos y no crea datos derivados. No se mantiene ningún ILF durante el proceso ni se altera el comportamiento del sistema.

En la figura 2-4 se muestra el proceso de solicitar información sobre los empleados (petición de entrada) y visualizar dicha información cuando aparece en pantalla (recuperación de salida)

- **Definición de Tipo de Fichero Referenciado (FTR)**

Un tipo de fichero referenciado es un fichero lógico interno (ILF) leído o mantenido por una función transaccional o un fichero de interfaz externo (EIF) leído por una función transaccional.

- **Definición de DET**

Un tipo de dato elemental (DET) es un campo único, no repetido, identificable por el usuario.

- **Definición de Proceso Elemental**

Un proceso elemental es la actividad más pequeña que es significativa para el (los) usuario(s). Debe ser autosuficiente y dejar la aplicación que está siendo contada, desde el punto de vista del negocio en un estado consistente.

2.4.3.2 Procedimientos de Medición de EI/EO/EQ

1. Identificar los Procesos Elementales
2. Clasificar los Procesos Elementales en EI/EO/EQ

Para los procesos elementales en los que el propósito principal es el mantenimiento de un ILF o la alteración del comportamiento del sistema:

3. Determinar la Complejidad de una EI.
4. Determinar la Contribución de una EI.

Para mayor información sobre las Reglas de FTR y DETS para EI/EO/EQ véase el Manual de prácticas de medición de puntos de función (IFPUG: International Function Point User Group, 2009) :

2.4.3.3 Procedimiento de Complejidad

Identificar y contar el número de FTRs y DETs, se valora la complejidad de EI/EO/EQ, utilizando la siguiente matriz de complejidad:

- **Entradas Externas (EI) - Matriz de Complejidad**

	1 a 4 DET	5 a 15 DET	16 o más DET
0 a 1 FTR	Baja	Baja	Media
2 FTRs	Baja	Media	Alta
3 o más FTRs	Media	Alta	Alta

Tabla 2-4 : Matriz de Complejidad para EI (IFPUG: International Function Point User Group, 2009)

- **Salidas Externas y Consultas Externas (EO y EQ) - Matriz de Complejidad**

	1 a 5 DET	6 a 19 DET	20 o más DET
0 a 1 FTR	Baja	Baja	Media
2 a 3 FTRs	Baja	Media	Alta
4 o más FTRs	Media	Alta	Alta

Tabla 2-5 : Matriz de Complejidad para EO y EQ (IFPUG: International Function Point User Group, 2009)

2.4.3.4 Procedimiento de Contribución

Convertir la complejidad de EI/EO/EQ en puntos de función no ajustados de acuerdo a las siguientes tablas:

- **Entradas Externas y Consultas Externas (EI y EQ)**

Valorización Complejidad Funcional	Puntos Función no Ajustados
Baja	3
Media	4
Alta	6

Tabla 2-6 : Conversión de la complejidad de EI y EQ en puntos de función no ajustados (IFPUG: International Function Point User Group, 2009)

- **Salidas Externas (EO)**

Valorización Complejidad Funcional	Puntos Función no Ajustados
Baja	4
Media	5
Alta	7

Tabla 2-7: Conversión de la complejidad de EO en puntos de función no ajustados (IFPUG: International Function Point User Group, 2009)

2.5 Costo de Estimación de Software

Según (Bozhikova, 2010), la estimación de costos de desarrollo de software es una de las tareas más desafiantes en la gestión de proyectos, que está asociada con la estimación fiable del tamaño del producto software y los recursos necesarios para producirlo. Asimismo, proporciona al administrador del proyecto información necesaria para desarrollar el proyecto de software en relación con la programación, el presupuesto, la asignación del personal y recursos.

Ningún método es necesariamente mejor o peor que el otro, sus puntos fuertes y débiles suelen ser complementarios entre sí. Hay un gran número de factores subjetivos que influyen en la objetividad de los resultados de estimación de software: la volatilidad de los requisitos del sistema, la rotación de los estimadores de costos, su experiencia tanto en el desarrollo de software y de los métodos y prácticas de estimación de costos (Bozhikova, 2010).

El método objetivo o método algorítmico como se clasifica en (Leung, 2002) , se basa en modelos matemáticos que estiman el costo del software en función de un número de variables que se consideran los principales factores de coste. El modelo COCOMO es un ejemplo de modelo algorítmico. Es un método fiable y preciso que evita en la mayoría de casos diferencias en los resultados de la estimación de costos relacionados con los factores subjetivos, ya que no permite al usuario del método manipular fácilmente los resultados(Bozhikova, 2010).

Las etapas básicas necesarias para llevar a cabo la estimación de software son:

- Identificar los objetivos del proyecto y los requisitos,
- Estimar el tamaño y la complejidad del producto,

- Estimación del esfuerzo aplicado, duración del proyecto y los recursos necesarios.

2.5.1 El modelo de costo constructivo (COCOMO)

El modelo del costo constructivo (COCOMO) es un método de estimación bien conocido, desarrollado originalmente por Barry Boehm (Boehm B. R., 2000) en la década de 1970. Este modelo determina el esfuerzo, el tiempo y calendario que se necesita en la planificación de un proyecto de desarrollo de software. Esta estimación se basa en el tamaño del proyecto que se expresa en el número de líneas de código estimados para desarrollar un producto de software.

Debido a la complejidad de los proyectos de software, el modelo original de COCOMO se modificó obteniendo COCOMO II, el cual ha sido desarrollado para ser utilizado por los proyectos empleando los procesos en cascada o en espiral (Center for Software Engineering at the University of Southern California (USC), 1995).

Este nuevo modelo puede estimar el esfuerzo, tiempo y horario necesario, utilizando tres diferentes sub modelos: Composición de Aplicaciones, Diseño temprano y modelos de Post arquitectura (Pow-Sang J. A., 2006).

2.5.2 Tamaño

El Tamaño de software puede definirse como un conjunto de atributos internos: longitud, funcionalidad y complejidad, que se puede medir estáticamente sin ejecutar el sistema. Reutilización mide cuanto un producto ha sido copiado o modificado, y también puede ser considerado como un aspecto de tamaño. La longitud es el tamaño físico del producto y se puede medir por la especificación, el diseño y el código. Funcionalidad mide las funciones vistos por el usuario (Ribu, 2001).

Una buena estimación del tamaño es muy importante para una buena estimación del modelo (Center for Software Engineering at the University of Southern California (USC), 1995).

En COCOMO 2.0, existen tres herramientas para medir el tamaño: Puntos de objetos, puntos de función sin ajustar, y líneas de código fuente.

En el Manual de Definición del Modelo de COCOMO II (Center for Software Engineering at the University of Southern California (USC), 1995) , el tamaño se

expresa como miles de líneas de código fuente (SLOC) o como puntos de función sin ajustar (UFP).

2.5.2.1 Contando Puntos de función no ajustados (UFP)

Los puntos de función son estimadores útiles ya que se basan en la información que está disponible al principio del ciclo de vida del proyecto. Miden un proyecto de software cuantificando la funcionalidad de procesamiento de la información asociada con los principales datos externos o de control de entrada, salida o tipos de archivo.

Para más detalle véase el Manual de Definición del Modelo de COCOMO II (Center for Software Engineering at the University of Southern California (USC), 1995).

2.5.2.2 Recuento de Líneas de Código Fuente (Siglas en inglés SLOC)

El tamaño del código se expresa en miles de líneas de código fuente (Siglas en inglés KSLOC). La definición de una línea de código es difícil debido a las diferencias conceptuales que intervienen en la contabilización de las sentencias ejecutables y las declaraciones de datos en diferentes lenguajes. Las dificultades surgen cuando se trata de definir las medidas consistentes en diferentes lenguajes de programación. En COCOMO II, la instrucción lógica de origen ha sido elegida como el estándar de la línea de código.

2.5.3 Estimación del Esfuerzo

En el Manual de Definición del Modelo de COCOMO II (Center for Software Engineering at the University of Southern California (USC), 1995) se presenta dos modelos: Post Arquitectura y los modelos iniciales de diseño.

El *Post-Arquitectura* es un modelo detallado que se utiliza una vez que el proyecto está listo para desarrollar y mantener un sistema desplegado. El sistema debe tener un paquete de arquitectura del ciclo de vida, que proporciona información detallada sobre las acciones del conductor de costos *(cost drivers),* y permite estimaciones de costos más precisos.

El *modelo inicial de diseño* es un modelo de alto nivel que se utiliza para explorar las alternativas arquitectónicas o estrategias de desarrollo incremental. Este nivel de detalle es consistente con el nivel general de la información disponible y del nivel general de precisión de estimación necesario.

2.5.3.1 Estimación del Esfuerzo Aplicado

En COCOMO II el esfuerzo se expresa en persona-mes (Siglas en inglés PM). Una persona-mes es la cantidad de tiempo que una persona pasa trabajando en el proyecto de desarrollo de software durante un mes.

COCOMO II trata el número de persona-hora por persona-mes, PH / PM, como un factor ajustable con un valor nominal de 152 horas por persona-mes. Este número no incluye el tiempo normalmente dedicado a feriados, vacaciones y el tiempo de fin de semana.

La ecuación general para calcular el esfuerzo necesario (PM) para el desarrollo del tamaño de un proyecto dado, expresado como persona-mes se da a continuación:

$$PM = A\ x\ Size^{E} x \prod_{i=1}^{n} EM_i \qquad \text{..... (1)}$$

$$where\ A = 2.94\ (\ for\ COCOMOII.2000)$$

La ecuación 1 lo conforman el Tamaño de desarrollo de software *Size*, una constante, *A*, un exponente, *E*, y un número de valores llamados multiplicadores de esfuerzo (*EM*).

- → El tamaño son miles de líneas de código fuente (KSLOC), puede ser también estimado a partir de los Puntos de Función Sin Ajustar (Siglas en inglés UFP), convertidos a líneas de código fuente (SLOC), luego dividido por mil.
- → Los conductores de costos (*cost drivers*) se utilizan para capturar las características del desarrollo de software que afectan al esfuerzo para completar el proyecto. Todos los conductores de costos de COCOMO II tienen niveles cualitativos de clasificación que expresan el impacto del conductor en el esfuerzo de desarrollo. Estos valores pueden variar de Extra Baja a Extra Alta. Cada nivel de calificación de cada factor de costo multiplicativo tiene un valor, denominado un Multiplicador Esfuerzo (Siglas en inglés EM), asociado con él. El valor de EM puede ser mayor o menor que 1.00, dependiendo del esfuerzo en el desarrollo del software.
- → El *E* exponente en la ecuación, es una agrupación de cinco factores de escala (SF) que dan cuenta de las economías relativas o deseconomías de escala, las cuales son factores que causan que las grandes empresas produzcan bienes y servicios con un incremento en el coste por unidad de cada producto, se encuentran los proyectos de software de diferentes tamaños.

Si E <1.0, el proyecto presenta economías de escala.

Si E = 1.0, las economías y deseconomías de escala están en equilibrio. Este l se utiliza a menudo para la estimación de costos de los proyectos pequeños.

Si E> 1.0, el proyecto presenta deseconomías de escala

Por otro lado (Bozhikova, 2010) nos muestra la siguiente ecuación para calcular el esfuerzo necesario para el tamaño de un proyecto de desarrollo expresado en persona-mes (PM):

$$PM = EF\ x\ EAF\ x\ KSLOC^{ee}\left[person-months\right]$$

$$Where:$$

$$PM = \begin{cases} PM_{nom}\ if\ EAF = 1; \\ PM_{real}\ if\ Effort\ Adjustment\ Factor\ is\ calculated. \end{cases} \quad \text{.....(2)}$$

En la ecuación 2 si el factor de ajuste del esfuerzo (EAF) es 1 (es el valor predeterminado) PM se interpreta como el esfuerzo nominal PMнoм necesario para un desarrollo de proyectos de tamaño dado, expresado en meses-persona (PM). Los valores del coeficiente de EF y el ee exponente en este caso se basan en el modelo COCOMO intermedio (Bozhikova, 2010).

2.5.3.2 Estimación del Factor de ajuste de esfuerzo

Según (Bozhikova, 2010), si el factor de Ajuste de Esfuerzo (EAF) es 1 (valor predeterminado) PM se interpreta como la PMnom esfuerzo nominal necesario para el tamaño del proyecto de desarrollo dado, expresado en persona-mes (PM).

El cálculo del factor de Ajuste de Esfuerzo (Siglas en inglés EAF) está relacionada con el cálculo de la PMreal esfuerzo ajustado. La estimación factor de ajuste de esfuerzo (EAF) podría basarse en los dieciocho (18) conductores de costos *(cost drivers)*.

Cada uno de los conductores de costos *(cost drivers)*: Capacidad del Analista , Aplicaciones de experiencia, Capacidad del programador, Utilización de herramientas de software, Desarrollo Multisite, Desarrollo del Cronograma requerido, Fiabilidad del software requerido, Tamaño de la base de datos, Complejidad del producto, Experiencia Personal, Idioma y Experiencia de herramientas, Continuidad de Personal, Ejecución limitaciones de tiempo, Restricciones de almacenamiento principal, Volatilidad de la Plataforma, Reutilización Requerido, Documentación y Usuario

(ACAP, APEX, PCAP, TOOL, SITE, SCED, RELY, DATA, CPLX, PLEX, LTEX, PCON, TIME, STOR, PVOL, RUSE, DOCU and USER) tiene (rating (i), i = 1 ... 18) en una escala de 6 puntos que va de "muy bajo" a "extra alto ". En la ecuación (3), el factor de ajuste de esfuerzo (EAF) para un proyecto dado se calcula como el producto de los equivalentes numéricos de los 18 niveles de clasificación EM (i). Los valores típicos para EAF rango desde 0.9 hasta 1.4.

$$EAF = \prod_{1}^{CDN} EM\,(1)$$

Donde:

$$CDN = \begin{cases} 18 \text{ Si COCOMO II es usado;} \\ 15 \text{ Si COCOMO Intermedio COCOMO es usado} \end{cases} \quad \text{…….. (3)}$$

En la tabla 2-8 se muestra los conductores de costos (*cost drivers)* en el Diseño temprano y Post-Arquitectura.

Conductores de costo Diseño temprano	**Conductores de costo Post-Arquitectura**
Fiabilidad y complejidad del producto (RCPX).	Fiabilidad requerida del software (RELY), Tamaño de la base de datos (DATA), Complejidad del producto (CPLX), Documentación de acuerdo a las necesidades del ciclo de vida (DOCU).
Reusabilidad requerida (RUSE).	Reusabilidad requerida (RUSE).
Dificultad de la plataforma (PDIF).	Restricción de tiempo de restricción (TIME), Restricción de almacenamiento principal (STOR), Volatilidad de la plataforma (PVOL) .
Capacidad del personal (PERS).	Capacidad de analistas (ACAP), Capacidad de programadores (PCAP), Continuidad del personal (PCON).
Experiencia del Personal (PREX).	Experiencia en aplicaciones (AEXP), Experiencia de plataforma (PEXP), Experiencia de lenguajes y herramientas (LTEX).
Facilidades (FCIL).	Uso de herramientas de software (TOOL), Desarrollo en múltiples lugares (SITE).
Restricciones de calendario impuestas al equipo de desarrollo (SCED).	Restricciones de calendario impuestas al equipo de desarrollo (SCED).

Tabla 2-8 : Conductores de costo *(Cost drivers)* en el diseño temprano y Post-Arquitectura (Boehm, Clark, Horowitz, Westland, Madachy, & Selby, 1995)

2.6 Indicadores de Medición

2.6.1 Fiabilidad y Validez

La medición es crucial para la ingeniería de software empírica. En el nivel más general, los procedimientos de medición deben poseer dos propiedades fundamentales: fiabilidad y validez. Se puede examinar la fiabilidad de un indicador en el grado en que un experimento, prueba, o cualquier procedimiento de medición arroja los mismos resultados en pruebas repetidas. Mientras más consistentes sean los resultados proporcionados en las mediciones repetidas, mayor será la fiabilidad del procedimiento de medida. Sin embargo, un indicador no sólo debe ser fiable, también debe ser válido. Un indicador de un concepto abstracto es válido en la medida en que mide lo que pretende medir (Carmines, 1979).

El procedimiento de medición en estudios comparativos de modelos de predicción de software con frecuencia es en general de la siguiente manera: Los modelos de predicción son validadas en una sola muestra (conjunto de datos) mediante el cálculo de un indicador de precisión utilizando la validación cruzada.

El modelo que obtiene la más alta precisión (es decir, un valor bajo del indicador de precisión) se considera "mejor" (En estudios más recientes, se agregan las pruebas de significación) (Ingunn, 2005).

2.6.2 Indicadores de Fiabilidad

La media de la magnitud del error relativo (MMRE) y el recuento del número de predicciones dentro de m% de los datos reales pred (m), son las dos estadísticas de precisión más utilizados (Kitchenham, 2001).

2.6.2.1 Margen de Error

Es importante entender los márgenes de error que son aceptables y no aceptables. Dado que el contexto es predecir esfuerzo de desarrollo temprano en el ciclo de vida del software, el margen de error siguiente, *e*, puede ser utilizado para fines de planificación (Hastings, April 2001):

$e \leq +20$ Porcentaje se considera predictivo y puede ser utilizado con confianza temprano en el ciclo de vida.

$+20 < e \leq \pm 50$ Porcentaje que se considera aceptable pero se debe utilizar con precaución.

$e > \pm 50$ Porcentaje no es fiable para propósitos de planificación.

El margen de error se calcula como: *e = (estimado – actual) / actual.*

2.6.2.2 Magnitud del error relativo (MRE)

Se utiliza para evaluar la eficiencia de las estimaciones (Heričko, 2008).

La MRE es una medida normalizada de la discrepancia entre los valores reales y predichos, que proporciona la base para los cálculos de MMRE, y puede ser definido como (Awan, 2010):

$$MRE = \frac{|E_{ai} - E_{pi}|}{E_{ai}}$$

Donde E_{ai} representa el esfuerzo real y E_{pi} representa esfuerzo previsto para la iteración i^{th}.

2.6.2.2.1 La Media de la Magnitud del error relativo (MMRE)

La media de la magnitud del error relativo (MMRE), estadística de predicción de la precisión, es el indicador más utilizado en los últimos años, sobre todo cuando se evalúa el rendimiento de los modelos de estimación de esfuerzo del software (Kitchenham, 2001).

Conte en (Conte, 1986) considera la MMRE <= 25 como aceptable para los modelos de predicción de esfuerzo.

$$\frac{1}{n}\sum_{i=1}^{i=n}\left(\frac{|x_i - x|}{x_i}\right)$$

Donde xi, es el valor actual y x es el valor estimado de una variable de interés.

2.6.2.3 Pred(m)

Otro indicador de calidad de predicción utilizado es pred (m), que es simplemente el porcentaje de estimaciones que están dentro de m% del valor real. Típicamente m se establece en 25 para que el indicador revela cuál es la proporción de las estimaciones se encuentran dentro de una tolerancia de un 25% (Kitchenham, 2001).

En el presente trabajo se utilizará como indicador de predicción el margen de error relativo (MRE) y en base al resultado se analizará si fiable para propósitos de planificación comparándola con la escala del margen de error señalada en el punto 2.7.2.1.

CAPITULO III: ESTADO DEL ARTE

En este capítulo se establecen los parámetros para la búsqueda y selección de los artículos e investigaciones académicas que soportan la sustentación de lo siguiente:

1. ¿Qué medidas/métricas se han definido para determinar la confiabilidad de una técnica que calcula el esfuerzo en proyectos de software?

2. ¿Qué técnicas basadas en puntos de función y prioridad en la construcción de casos de uso han sido utilizadas para estimar el esfuerzo en proyectos de software con ciclo de vida incremental y paradigma orientado a objetos?

3.1 Estrategia de Búsqueda y selección

Se utiliza PICOC (Población, Intervención, Comparación, Salidas y Contextos) para enmarcar las preguntas de investigación a definir. Ver tabla 3-1.

Criterio	Descripción
Población)	Desarrollo de software orientado a objetos que empleen el modelo de ciclo de vida incremental
Intervención	Métodos, técnicas basadas en Puntos de función (PF), UML y prioridad en la construcción de casos de uso para la estimación y planificación.
Comparación	Análisis de Puntos de Prueba (TPA), Análisis de Puntos de Casos de Uso (UCP) y COCOMO.
Salidas	Precisión o exactitud en la predicción, Margen de Error Relativo (MRE) confiable para estimación
Contexto	Académico, industria de software. Incluirá cualquier tipo de estudio empírico: observaciones, cuestionarios, encuestas, experimentos controlados, casos de estudio.

Tabla 3-1: Aplicación del criterio PICOP

Luego de establecer los criterios del PICOC. Se procede con la búsqueda en Scopus, Science Direct, IEEE Xplore y la Biblioteca Digital ACM. La búsqueda secundaria consiste en revisar las referencias y citaciones de los artículos obtenidos en la búsqueda primaria con el fin de encontrar artículos relevantes al tema.

Asimismo, se utilizó el motor de búsqueda de Google Académico, para determinar las citaciones a los artículos seleccionados.

A continuación se detallan algunos parámetros de búsqueda:

Parámetro	Resultados
(((Title:"Function Point"AND"Iterative") or (Title:"FPA"AND "Incremental"))or ((Title:"ifpug" AND "incremental")))	01
(((((Title:"Iterative development"AND"function Points Analysis Method") or (Title:"Estimating Methods"AND "Function Point counting")or (Title:"Effort estimates" AND "object-oriented size estimation")))))	01
((Title:"Functional size measurement"AND"function Point") or (Title:"software size measurement"AND "FPA"))or ((Title:"Functional size measurement" AND "Function Point Analysis"))	06
(((((Title:"Cost Estimation"AND"Effort estimation") or (Title:"Software development"AND "Effort estimation"))or ((Title:"Software Planning" AND "Effort estimation")))))	55
(((((Title:"Software Metrics"AND"acurracy indicators") or (Title:"Cost estimation"AND "accuracy stadistics"))or ((Title:"Software metrics" AND "Reliability and validity")))))	01
((((((Title:"Accuracy"AND"Effort estimation") or (Title:"Software development"AND "Effort estimation"))or ((Title:"metrics" AND "reliability and validity"))))))	85

Tabla 3-2: Búsqueda de artículos en ACM

Los siguientes artículos fueron seleccionados:

- ✓ Model-based functional size measurement (Lavazza, 2008).
- ✓ Some issues on scheduling estimation model for object-oriented software projects (Mishra, 2009).
- ✓ A novel fuzzy based approach for effort estimation in software development (Sinhal, 2013)
- ✓ An approach for effort estimation in incremental software development using cosmic function points (Paz, 2014).

Parámetro	**Resultados**
(("Document Title":"Function Point") OR ("Document Title":"Function Points") OR ("Document Title":"ifpug"))	86
(("Document Title":"Function Poitn") OR ("Document Title":"FPA") OR ("Document Title":"ifpug"))	57
((("Document Title":UML AND "Document Title":incremental) OR ("Document Title":"Unified Modelling language" AND "Document Title":iterative) OR ("Document Title":"Unified Modelling language" AND "Document Title":incremental)))	04
(("Document Title":"Effort Estimation") OR ("Document Title":"Tupuy")))	197
(((("Document Title":Software validity AND "Document Title":Software reliability) OR ("Document Title":"Margin error" AND "Document Title":software metrics) OR ("Document Title":"Accuracy" AND "Document Title":software indicators)))))	02
(((("Document Title":Effort estimation AND "Document Title":Software development projects) OR ("Document Title":"Function points" AND "Document Title":software planning) OR ("Document Title":"incremental" AND "Document Title":object oriented)))	16

Tabla 3-3: Búsqueda de artículos en IEEE XPLORE

Los siguientes artículos de la base de datos IEEE XPLORE fueron seleccionados:

- ✓ Model-Based Dynamic Cost Estimation and Tracking Method for Agile Software Development (Kang, 2010).
- ✓ Extending function point analysis to object-oriented requirements specifications (Harput, 2005).
- ✓ An Approach of a Technique for Effort Estimation of Iterations in Software Projects (Pow-Sang J. A., 2006).
- ✓ A vector-based approach to software size measurement and effort estimation (Hastings, April 2001).
- ✓ Reliability and validity in comparative studies of software prediction models (Ingunn, 2005).
- ✓ Predicting software test effort in iterative development using a dynamic bayesian network (Awan, 2010).

Parámetro	Resultados
title("Function Point") or title("Function Points") or title("ifpug")	26

Tabla 3-4: Búsqueda de artículos en SCIENCEDIRECT

Los siguientes artículos de la base de datos SCIENCEDIRECT fueron seleccionados:

- ✓ The size and effort estimates in iterative development (Heričko, 2008).
- ✓ Function point counting: one program's experience (Orr, 2000).

Por otro lado, se seleccionaron artículos de las siguientes bases de datos y repositorios digitales:

De la base de datos de la Australasian Journal of Information:

- ✓ Assessing software cost estimation models: criteria for accuracy, consistency and regression (Lo, 2007).

De la base de datos Springer:

- ✓ Effort Estimation in Incremental Software Development Projects Using Function Point (Pow-Sang J. I., 2012).

Del Archivo Digital de la Universidad Politécnica de Madrid UPM

- ✓ Tesis Doctoral: Técnicas para la Estimación y Planificación de Proyectos de Software con Ciclos de Vida Incremental y Paradigma Orientado a Objetos (Pow Sang Portillo, 2012).

Del Repositorio Digital de Tesis Pontificia Universidad Católica del Perú PUCP:

- ✓ Validación de Técnicas de Estimación de Esfuerzo en Proyectos de Software con Ciclos de Vida Incremental y Paradigma Orientado a Objetos (Villanueva Bendezú, 2013).

3.2 Estudios preliminares de literatura

Los estudios de la literatura que se han cubierto son: Técnicas basadas en Puntos de función para Modelos Orientado a Objetos, Técnicas para evaluar la confiabilidad de modelos de estimación del esfuerzo de software y Técnicas para calcular el esfuerzo requerido por cada incremento.

Con relación a ***Técnicas basadas en Puntos de función para modelos orientado a objetos*** podemos señalar los siguientes:

El Método de Puntos de Función Orientado a Objetos (Siglas en inglés OOMFP) se presenta después de los pasos de un modelo de proceso para la medición del software. Utilizando este modelo de proceso se presenta el diseño del método de medición, su aplicación en un estudio de caso, y el análisis de los diferentes tipos de evaluación que se pueden llevar a cabo para validar el método y verificar su aplicación y resultados (Abrahão, 2006).

En este artículo se proponen reglas de medición de análisis de punto de función detallados con las especificaciones de diseño basado en el Lenguaje Unificado de Modelado y se describe una herramienta de medición de puntos de función, cuyas entradas son las especificaciones de diseño desarrollados en Rational Rose. Asimismo, en este artículo se presenta el trabajo de validación de herramientas de software que participan en la evolución del software en una organización en la que se aplicó la herramienta para las especificaciones de diseño actuales y se examinó las diferencias entre los valores de los puntos de función obtenidos por la herramienta y los de una medición de un especialista con experiencia en punto de función en la organización (Uemura, 2001).

Este artículo tiene como objetivo evaluar una metodología para la medición de punto de función sobre la base de la representación del sistema a través de modelos de UML. Esta metodología tiene como objetivo proporcionar una definición precisa del

objeto de la medición, así como el procedimiento de medición y las reglas (del Bianco, 2008).

En este artículo se propone un mapeo unificado de modelos UML en puntos de función. El mapeo se describe formalmente para permitir la automatización del procedimiento de recuento. Tres niveles de estimación [estimación básica, estimación comparativa, estimación final] están definidos que corresponden a los diferentes niveles de abstracción del sistema de software. El nivel de abstracción influye en la precisión de una estimación. Esta investigación basada en un pequeño conjunto de datos, demostró que la precisión aumenta con cada subsiguiente nivel de abstracción (Živkovič, 2005).

En este artículo se definen reglas que especifican una transformación semi-automática de un modelo de requisitos orientado a objetos a un modelo de FPA. Debido a que en el método más ampliamente utilizado para la estimación del tamaño, Análisis de Puntos de Función (FPA), no está clara la cantidad de puntos de función que se pueden razonablemente contar de especificaciones de requisitos orientados a objetos. Se encontró que esto no puede hacerse de forma totalmente automática, ya que varios constructos de tal representación se pueden interpretar de varias maneras en el espíritu del FPA, dependiendo del contexto (Harput, 2005).

En este artículo se describe que a pesar de que el Análisis de Puntos de Función (FPA) es el método más ampliamente utilizado para medir el tamaño de los requisitos de software, se ve afectada por varios inconvenientes: debe ser realizada por personal específicamente calificado, es caro, y las medidas resultantes están sujetas a una gran variabilidad. Con el fin de resolver, al menos parcialmente estos problemas, los investigadores han propuesto basar el conteo de FP con modelos UML. Por lo tanto, en este documento se ilustra una técnica para la construcción de modelos FPA orientadas en UML que no necesitan incluir más información que por lo general requiere el proceso de desarrollo, y son fáciles de medir. Como resultado, los FPA se pueden realizar de una manera transparente, dando resultados fiables. La técnica propuesta fue validada por medio de un experimento controlado y un conjunto de aplicaciones piloto, que también se describen brevemente en el artículo (Lavazza, 2008).

En este artículo, se presenta un estudio empírico que evalúa el Método O-O Puntos de Función (OOmFP), un procedimiento de medición de tamaño funcional de sistemas orientados a objetos que se especifican utilizando el método O-O. Para ello se realizó un experimento de laboratorio con los alumnos, con la finalidad comparar OOmFP con el IFPUG – Análisis de Puntos de Función (FPA) en una serie de variables, como la eficiencia, la reproducibilidad, la precisión, facilidad de uso percibida, utilidad percibida y la intención de uso. Los resultados demostraron que OOmFP consume más tiempo que FPA pero los resultados de la medición son más reproducibles y precisos. Los resultados también indican que OOmFP se percibe como más útil y más probable que se adopten en la práctica que FPA en el contexto del desarrollo de sistemas del Método O-O (Abraha˜o, 2007).

Finalmente, en este artículo se propone un modelo para la estimación de la programación mediante el uso de métricas disponibles OO y afectando factores, debido a que los proyectos de software orientado a objetos (OO) son cada vez más popular que los proyectos basados en la tecnología estructurados (funcional). La tecnología de objetos (OT) ofrece apoyo para entregar productos al mercado más rápidamente y proporcionar alta calidad con bajos costos de mantenimiento. La estimación es un importante campo de la ingeniería de software y hay necesidad de buenas métricas OO y modelos para el proceso y gestión de productos. Para reducir los costes de desarrollo y cumplir con los plazos ajustados en los proyectos de software cortos de personal, es esencial el plan de los administradores y programar sus proyectos de una manera óptima (Mishra, 2009).

Con relación a la ***Evaluación de confiabilidad de modelos de estimación del esfuerzo de software*** podemos señalar lo siguiente:

En este artículo se propone la Medida Tamaño Vector (VSM) que incorpora la funcionalidad y la complejidad de un problema de una manera equilibrada y ortogonal. VSM se utiliza como la entrada a un modelo de predicción vectorial (VPM) que puede ser utilizado para estimar el esfuerzo de desarrollo temprano en el ciclo de vida del software. Se validó teóricamente el enfoque sobre un marco formal. También se validó empíricamente el enfoque con un estudio piloto. Los resultados indican que el enfoque proporciona un mecanismo para medir el tamaño de los sistemas de software, clasificar los sistemas de software, y estimar el esfuerzo de desarrollo temprano en el ciclo de vida del software dentro de + / -20 por ciento a través de una gama de tipos de aplicaciones (Hastings, April 2001).

En este artículo se examina un procedimiento de investigación utilizado con frecuencia, comprende tres ingredientes principales: una sola muestra de datos, un indicador de la precisión y la validación cruzada. Se utilizó la simulación y comparó un aprendizaje automático y un modelo de regresión. Los resultados sugieren que es el propio procedimiento de investigación lo que hace poco fiable. Esta falta de fiabilidad puede contribuir fuertemente a la falta de convergencia. Así, los hallazgos ponen en duda las conclusiones de cualquier estudio de los modelos de predicción de software que utilizan este procedimiento de investigación como base de la comparación de los modelos de la competencia. Por lo tanto, la necesidad de desarrollar procedimientos de investigación más fiables antes de que puedan tener confianza en las conclusiones de los estudios comparativos de los modelos de predicción de software (Ingunn, 2005).

En este artículo se examina las medidas actuales de estimación de la precisión y la coherencia, y propone dos nuevos: el promedio ponderado de los cuartiles de los errores relativos (WMQ) como una medida de la precisión y la desviación estándar de los coeficientes de la estimación de la observación real (SDR) como un medida de la consistencia. Además, se propone un nuevo criterio de regresión para determinar los parámetros del modelo. Las medidas y criterios propuestos fueron probados con un conjunto de datos a partir de proyectos reales de software del mundo. Los resultados obtenidos muestran que estas nuevas medidas y criterios superan muchas de las dificultades de los ya existentes (Lo, 2007).

En este artículo se valida una red bayesiana dinámica para predecir esfuerzo de las pruebas en el desarrollo de software iterativo. Se evalúa el marco propuesto en varias maneras: En primer lugar, el comportamiento del marco se observa al considerar diferentes parámetros y la validación inicial. Luego en segundo lugar, el marco es validado mediante la incorporación de los datos a partir de dos proyectos industriales. La exactitud de los resultados ha sido verificada a través de diferentes mediciones de precisión de la predicción y pruebas estadísticas. Los resultados de la verificación confirman que el marco tiene la capacidad de predecir esfuerzo de las pruebas en proyectos iterativos con precisión (Awan, 2010).

Finalmente en (Villanueva Bendezú, 2013) se realiza 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 Portillo, 2012) con alumnos de Ingeniería

Informática de la Pontificia Universidad Católica de Valparaíso, obteniendo resultados similares a los obtenidos en experimentaciones previas.

Con relación a ***Técnicas para calcular el esfuerzo requerido por cada incremento*** podemos señalar lo siguiente:

En este artículo se aborda el problema de la estimación del tamaño en el desarrollo iterativo. Se presenta un nuevo método que permite la temprana estimación del tamaño utilizando artefactos del Lenguaje Unificado de Modelado (UML). El método incluye pasos que aumentan la precisión de la estimación en iteraciones posteriores. También se presenta la prueba de su aplicabilidad y resultados de la investigación. Los resultados anticipan la posibilidad de una mejora significativa en las estimaciones del tamaño y el esfuerzo por la aplicación del enfoque que aquí se presenta (Heričko, 2008),

En el artículo se presenta un modelo de estimación de costos de software para el desarrollo ágil. Los métodos de estimación de costos existentes de desarrollos ágiles utilizan el punto de historia [story point]. Debido a que es un valor relativo, los resultados de la estimación tienden a ser fácilmente fluctuados por la pequeña variación de la línea de base del punto de historia. Para el seguimiento del progreso del proyecto, la velocidad fue medida para comprobar el progreso y se utiliza para hacer el plan de la iteración y las emisiones del proyecto. El método propuesto en este artículo proporciona la estimación sistemática y metodología de seguimiento dinámico de proyectos ágiles. Para estimar el esfuerzo de desarrollo de un proyecto, los puntos de función se utilizan además de punto de historia. Los puntos de función se determinan sobre la base de los casos de uso de las características deseadas del producto. Asimismo, se adopta el algoritmo de filtro Kalman para el seguimiento de los avances del proyecto. Los restantes puntos de función en un cierto punto durante el proyecto se modelan como el modelo de espacio de estado para el filtro de Kalman. La variación diaria de punto de función es observada y se introduce en el filtro de Kalman para proporcionar concreta estimación y la velocidad (Kang, 2010).

En el artículo se documenta las variaciones observadas y especula sobre cómo estas variaciones pueden medir las diferencias intrínsecas en el proceso de desarrollo en vez de las diferencias en la capacidad de conteo, realizadas en un programa de desarrollo de software implementado como una serie de incrementos en un nuevo

entorno. El método de conteo de puntos de función fue elegida como el método básico de la estimación sobre la mayor parte del proyecto donde se aplicó de manera uniforme en un entorno de desarrollo estable durante unos dos años. A pesar de la estabilidad en el medio ambiente y en el personal de desarrollo, se observaron variaciones interesantes en los recuentos de puntos de función a través del tiempo. Se propone un marco basado en esta variación como una posible manera de medir objetivamente las variaciones de desarrollo en el diseño y la eficiencia de la aplicación (Orr, 2000).

Este artículo tiene como objetivo estimar el esfuerzo de una manera eficiente utilizando una técnica Fuzzy. Para este propósito, el conjunto de datos COCOMO81 y el Sistema de Inferencia Fuzzy (FIS) de MATLAB se utilizan para la implementación. Finalmente, los resultados se comparan con los métodos tradicionales utilizando parámetros como la magnitud media del error relativo (MMRE) y Pred (25) (Sinhal, 2013).

En el artículo se presenta un enfoque denominado Incremental CFP, que permite estimar el esfuerzo por cada incremento usando Puntos de Función Cósmicos (CFP) y el factor de ajuste del esfuerzo de COCOMO (EAF). Con la finalidad de validar este nuevo enfoque, se empleó datos de dos proyectos diferentes en las que los estudiantes de pregrado tenían que trabajar inicialmente con puntos de función IFPUG. Los resultados mostraron que la diferencia entre el esfuerzo estimado y el esfuerzo real era inferior a 30% para el segundo incremento, lo que concluye que la técnica propuesta es aceptable y predictivo (Paz, 2014).

Finalmente en la tesis del Dr. Pow Sang en (Pow Sang Portillo, 2012) se propuso un enfoque denominado Tupuy, el cual 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. A continuación se define la técnica Tupuy así como un caso de ejemplo.

3.3 TUPUY

3.3.1 Definición

Tupuy, que significa en lengua quechua “medir” o “pesar”, está conformado por tres técnicas: UML2FP, Diagrama de precedencias de casos de uso (UCPD) e Incremental-FP, 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 (Pow Sang Portillo, 2012).

En la figura 3-1 se muestra el esquema general de la técnica Tupuy, conformada por tres técnicas: La Técnica UML2FP emplea el Diagrama de clases de análisis para realizar el cálculo de los puntos de función sin ajustar, La Técnica UCPD emplea las especificaciones de casos de uso para definir la secuencia de construcción de los casos de uso y Finalmente la Técnica Incremental FP utiliza los resultados obtenidos de emplear las técnicas UCPD y UML2FP definiendo los incrementos a construir y la determinación de qué casos de uso se va a desarrollar en cada incremento así como estimar el esfuerzo que se requiere para concluir cada incremento.

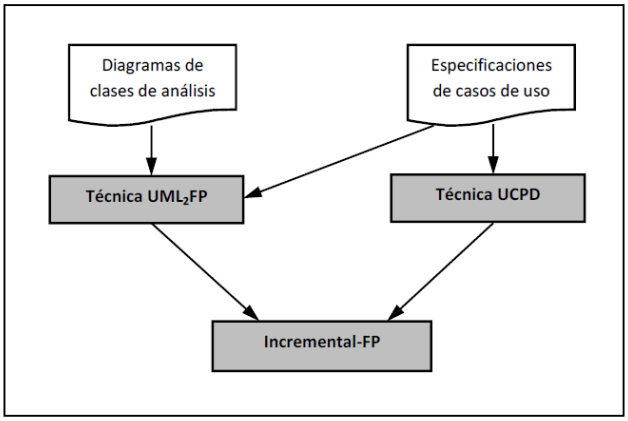

Figura 3-1: Esquema General de Tupuy (Pow Sang Portillo, 2012)

En el presente trabajo se utilizó como herramienta de apoyo a Tupux, el cual es un componente adicional de StarUML que facilita la planificación y estimación, utilizando puntos de función, en los proyectos de software que adoptan el modelo incremental (Balbin, 2009).

3.3.2 Descripción de las técnicas

- **UML2FP**

UML2FP es una técnica que permite el cálculo de los puntos de función basado en modelos orientados a objetos, utilizando para ello el diagrama de clases de análisis (Balbin, 2009) (Pow Sang Portillo, 2012).

En la figura 3-2 se muestra un ejemplo del diagrama de clases elaborado en la herramienta StarUML, en la cual se representa a los ficheros Producto, Línea Factura y Factura, así como la relación entre ellas.

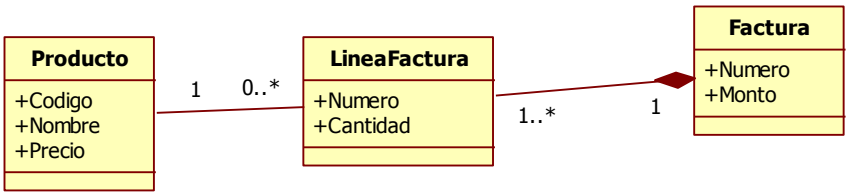

Figura 3-2: Ejemplo de diagrama de clases

- **Diagrama de precedencias de casos de uso (UCPD)**

El diagrama de precedencia ayuda al programador a definir visualmente el orden de construcción de los casos de uso y, por tanto inferir cuáles de ellos deben ser incluidos en cada incremento (Balbin, 2009).

En la figura 3-3 se muestra un ejemplo del diagrama de precedencias de casos de uso, los casos de uso que están en el lado izquierdo del diagrama (caso de uso A) se llevará a cabo antes de los que están en el lado derecho (casos de usos B, C y D) (Balbin, 2009).

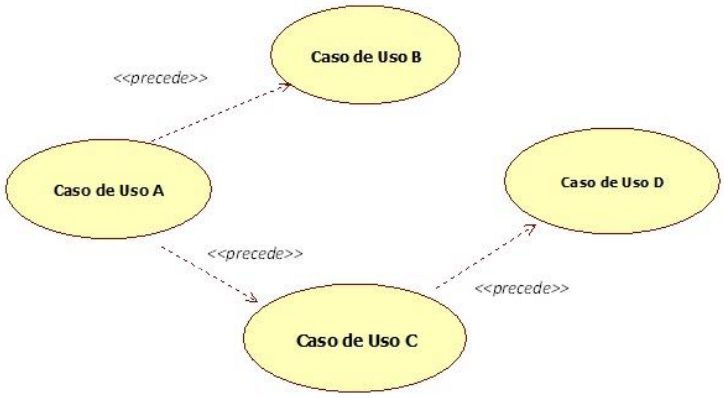

Figura 3-3: Ejemplo traducido de UCPD (Pow Sang Portillo, 2012)

- **Incremental –FP**

Con los resultados obtenidos al emplear las técnicas UML2FP y UCPD, se aplica finalmente la técnica Incremental–FP, la cual comprende la ejecución de cuatro actividades que permitirán definir los incrementos a construir, qué casos de uso se van a desarrollar en cada incremento y estimar el esfuerzo que se requiere para concluir cada incremento (Pow Sang Portillo, 2012).

En la figura 3-4 se muestra que con los resultados obtenidos de aplicar la técnica UML2FP (Actividad 1) y la técnica UCPD (Actividad 2) se aplica la técnica Incremental-FP conformada por las actividades 3 al 6 para cada incremento.

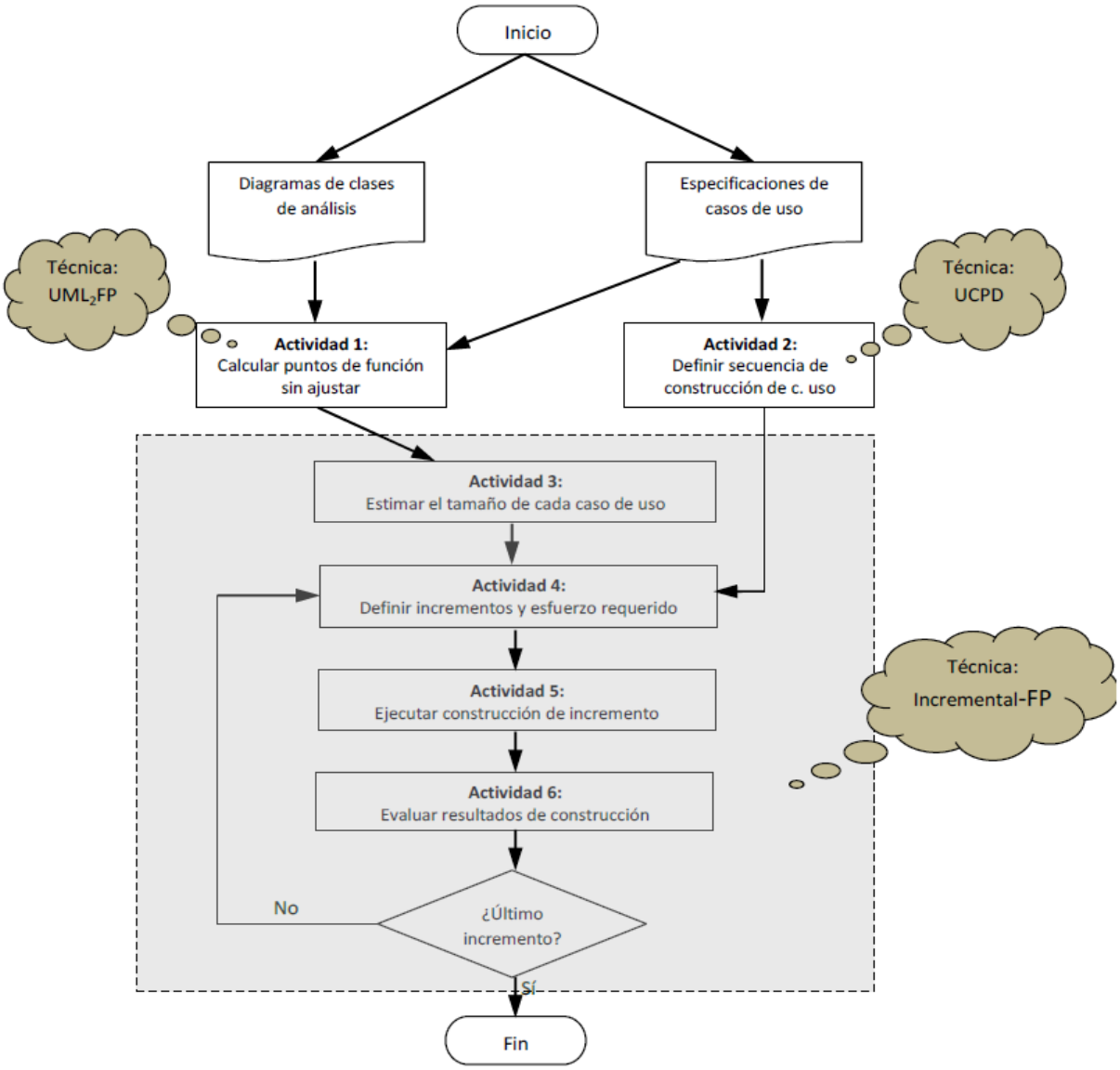

Figura 3-4: Actividades de Tupuy y técnicas que incluye (Pow Sang Portillo, 2012)

3.3.3 Caso Ejemplo

El ejemplo consiste en un sistema de información para el área de ventas de una empresa.

- **Aplicación de UML2FP**

Lo primero que especifica ésta técnica, es determinar los ficheros con sus DETs y RETs. Para ello, se deberá emplear el diagrama de clases de análisis. Ver figura 3-5.

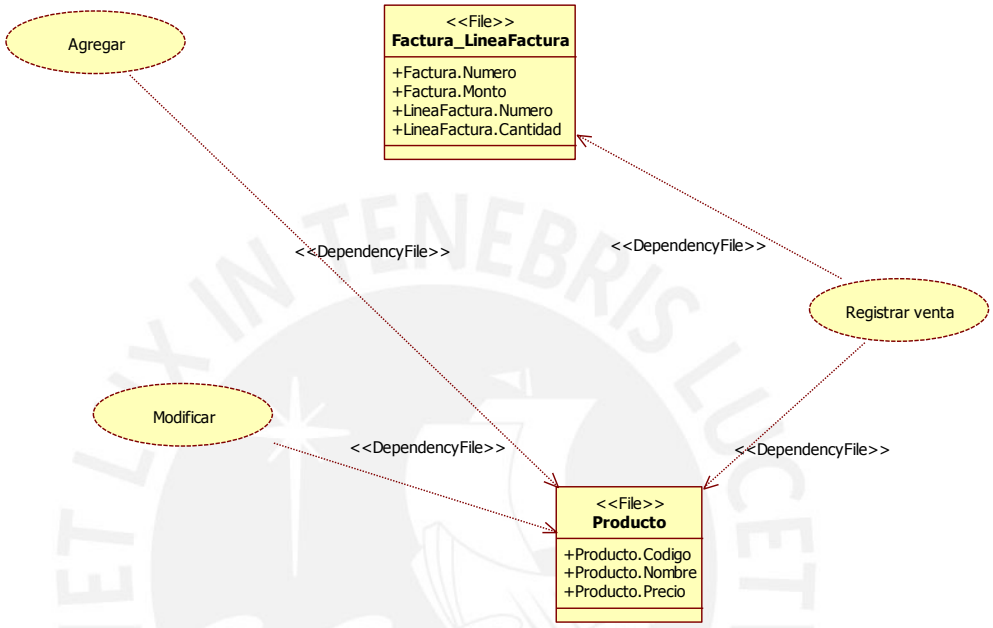

Figura 3-5: Diagrama de clases

Teniendo como base el diagrama de clases Figura 3-5, se procederá a determinar los Puntos de función sin ajustar por fichero y transacción, para ello se debe seguir las reglas definidas por (IFPUG: International Function Point User Group, 2009).

Tupux es un componente adicional de StarUML que permite calcular los puntos de función tomando como base diagramas orientados a objetos. En la Figura 3-6, se muestra una pantalla de StarUML con Tupux, en él se representa a los ficheros Factura_Línea Factura y Producto como estereotipos de clases, y las transacciones Registrar venta, Agregar y modificar como colaboración. Gráficamente, la relación entre los ficheros y la transacción se realiza mediante una dependencia "DependencyFile".

- **Cálculo de Puntos de función sin ajustar (PFSA) por Fichero:** Factura_Línea Factura, Producto.

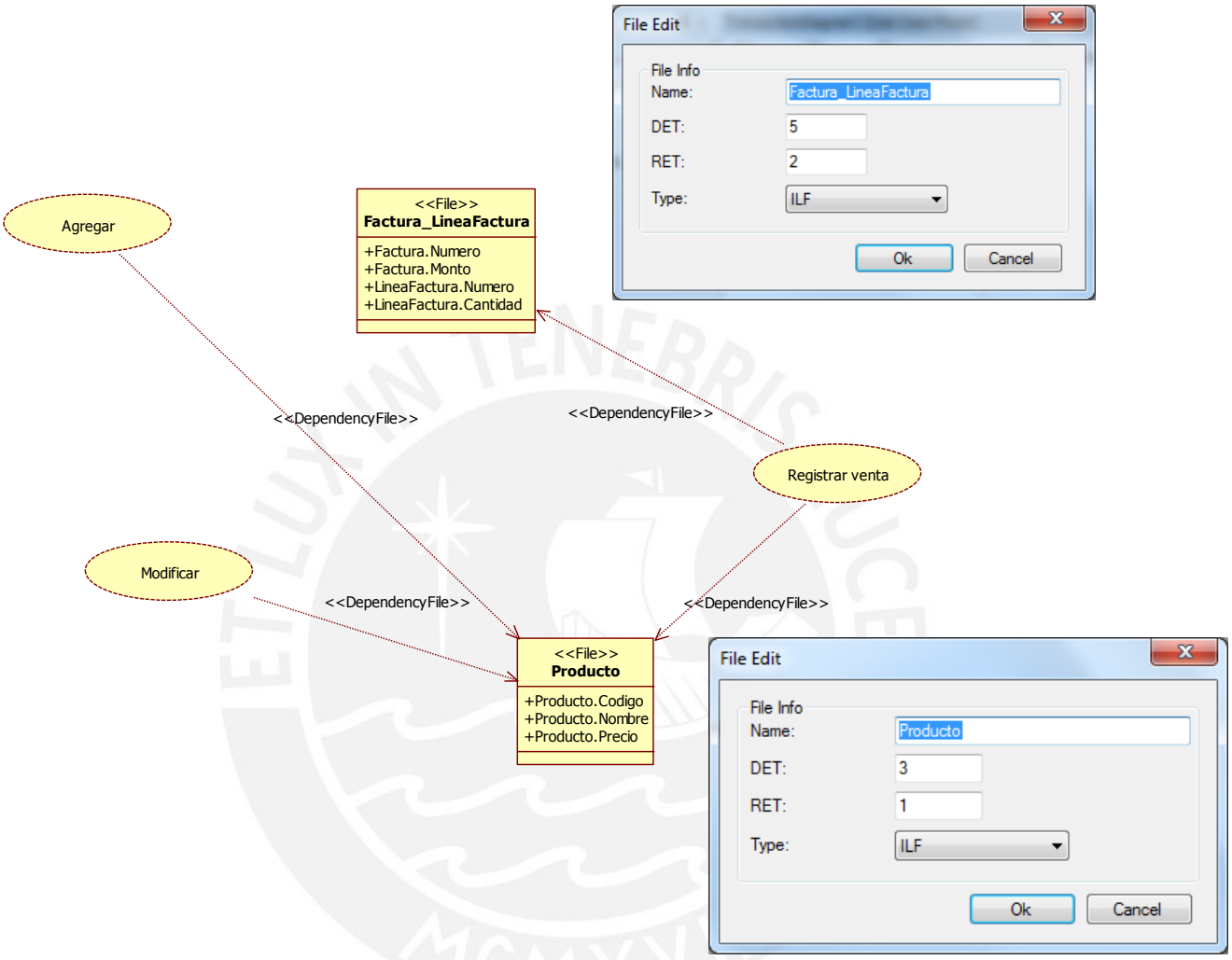

Figura 3-6: Cálculo de Puntos de Función de Ficheros con Tupux

En la figura 3-6 se observa los DETs y RETs de los ficheros ILF Factura_LíneaFactura y Producto determinados con Tupux.

Nombre de Fichero	Tipo de Fichero	DET	RET	Complejidad	PFSA
Factura_Línea Factura	ILF	5	2	Baja	7
Producto	ILF	3	1	Baja	7

Tabla 3-5: PFSA por Fichero del ejemplo

En la tabla 3-5 se determina la complejidad utilizando la tabla 2-1 y el conteo de los puntos de función sin ajustar utilizando la tabla 2-2 para cada fichero.

- **Cálculo de Puntos de función sin ajustar (PFSA) por Transacción:** Registrar venta, Agregar y modificar.

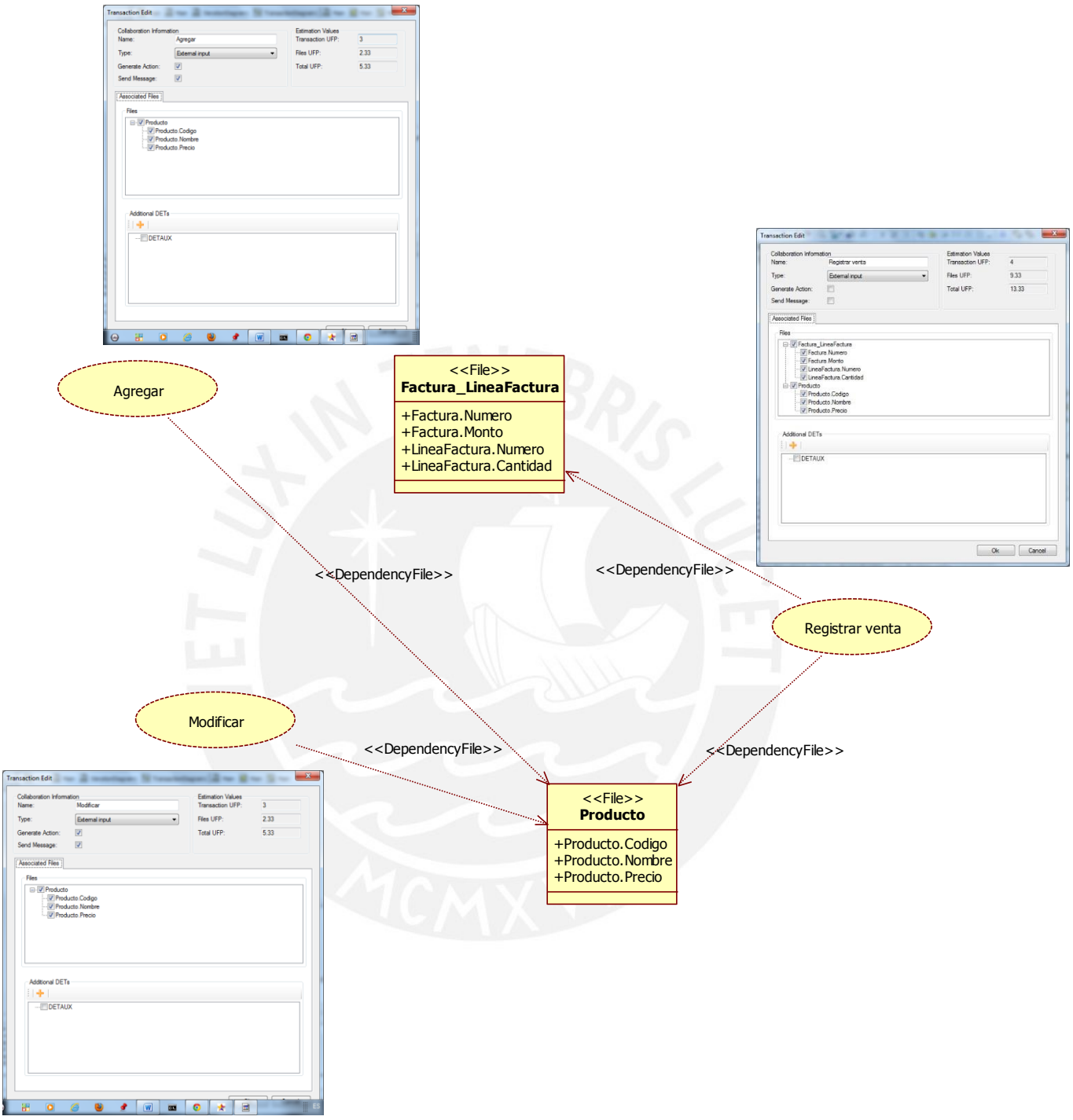

Figura 3-7: Cálculo de Puntos de Función de las transacciones con Tupux

En la figura 3-7 se observa los DETs y RETs de las transacciones agregar, registrar venta y modificar determinados con Tupux.

Caso de Uso	Transacción	Tipo	FTR	DET	Complejidad	PFSA
Actualizar Producto	Agregar	EI	1	5	Baja	3
	Modificar	EI	1	5	Baja	3
Registrar Venta	Registrar Venta	EI	2	7	Media	4

Tabla 3-6: PFSA por transacción de cada caso de uso del ejemplo

En la tabla 3-6 se muestra como tipo de función transaccional entrada externa (EI), se determina la complejidad utilizando la tabla 2-4 y el conteo de los puntos de función sin ajustar utilizando la tabla 2-6.

- **Aplicación de Diagrama de precedencias de casos de uso (UCPD)**

El diagrama de precedencias de casos de uso se utiliza para seleccionar que caso de uso se debe desarrollar en cada incremento. Ver Tabla 3-7.

Se debe determinar los Puntos de Función sin ajustar (UFP) para cada incremento.

Incremento	Caso de Uso
1	Actualizar Producto
2	Registrar Venta

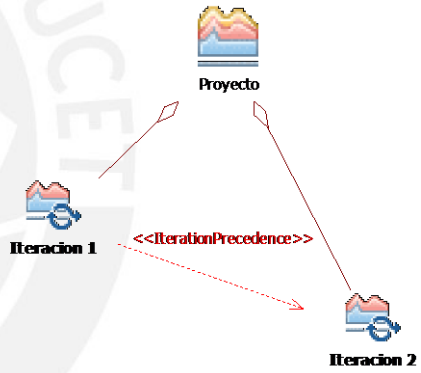

Tabla 3-7: Diagrama de precedencias de casos de uso del ejemplo

- **Aplicación de Incremental –FP**

Incremental-FP indica que se deben realizar tres actividades: estimar el tamaño de cada caso de uso (actividad 3), definir incrementos y esfuerzo requerido (actividad 4) y evaluar resultados de la construcción del incremento (actividad 6).

Estimación del tamaño de cada caso de uso (actividad 3)

1. Los Puntos de Función de todo el proyecto es igual a la suma de los puntos de función sin ajustar contado separadamente para cada incremento (Pow-Sang J. I., 2012).

$$PFSA\_Total = \sum_{i=1}^{n} PFSA(i)$$

(1)

2. Se debe determinar los Puntos de Función sin ajustar de los Ficheros Lógicos Internos (ILF) y Ficheros de Interfaz Externos (EIF) para cada caso de uso.

El número de Puntos de Función sin ajustar de los Ficheros Lógicos Internos (ILF) y Ficheros de Interfaz Externos (EIF) se debe distribuir proporcionalmente a los Puntos de Función de las Transacciones (EI, EO y EQ) (Pow-Sang J. I., 2012).

$$PFSA\_FichTr(j) = \sum_{i=1}^{n} \left[\frac{1}{TTrPF(i)} x\ PFSA\_Fich(i) \right] \quad (2)$$

En la Tabla 3-8 la primera columna muestra el nombre de los casos de uso; la segunda columna, el nombre de la transacción asociada a los casos de uso; la tercera columna, los puntos de función sin ajustar por transacción calculados en la tabla 3-6; la cuarta y quinta columna, muestra con una "x" que fichero es empleado por qué transacción; la sexta columna, presenta los puntos de función sin ajustar por fichero utilizando la fórmula 2; y la última columna, muestra los puntos de función sin ajustar para cada caso de uso que se calcula sumando los puntos de función sin ajustar por transacción más los puntos de función por fichero de cada transacción.

$$PFSA\_CU(k) = \sum_{j=1}^{n} [PFSA\_Tr(j) + PFSA\_FichTr(j)] \quad (3)$$

Caso de Uso	Transacción	PFSA por Transacción	Fichero ILF		PFSA por Fichero	FPSA por Caso de Uso
			Factura_Línea Factura	Producto		
Actualizar Producto	Agregar	3		X	2.33	5.33
	Modificar	3		X	2.33	5.33
Registrar Venta	Registrar Venta	4	X	X	9.33	13.33
			Total			**23.99**

Tabla 3-8: PFSA debido a ficheros por cada transacción del ejemplo

Definición de incrementos y esfuerzo requerido (actividad 4)

Para calcular el esfuerzo estimado para el primer incremento, se debe calcular en primer lugar la productividad estimada. La productividad es calculada utilizando la productividad histórica de proyectos anteriores, la cual es medida en horas-persona/UFP y el EAF de COCOMO (Pow-Sang J. I., 2012).

Cabe mencionar que se utilizará como factor de ajuste, el factor de ajuste del esfuerzo (EAF) de COCOMO y no el valor del factor de ajuste (VAF) de los Puntos de función, debido a que éste último no considera los cambios en el contexto de un proyecto, tales como la experiencia en el desarrollo de aplicaciones o experiencia en la herramienta de programación y del lenguaje, factores que son considerados por los multiplicadores de esfuerzo de COCOMO (Pow-Sang J. I., 2012).

Para el ejemplo se va a suponer que la productividad histórica es de 3.50 horas-persona/PFSA para un EAF de 1.00; entonces se podrá estimar el esfuerzo que se necesitará para el primer incremento utilizando la fórmula 4. Por otro lado, considerando que el equipo de desarrollo no tiene mucha experiencia en el lenguaje ni el desarrollo de aplicaciones; por lo tanto, el EAF sería de 1.90. Con estos datos, se podrá calcular la productividad estimada para el primer incremento, la cual sería de 6.65 horas-persona/PFSA. Asimismo, como se piensa implementar el caso de uso Actualizar producto y éste tiene 10.66 PFSA, entonces el esfuerzo estimado sería de 70.89 horas-persona, como se muestra en la tabla 3-9.

$$Productividad\ estimada\_Incr1 = \frac{EAF\_Incr1\ x\ Productividad\ Histórica}{EAF\_Historico} \quad (4)$$

Incremento	PFSA	EAF	Productividad Estimada (h-p/PFSA)	Esfuerzo Estimado (h-p)	Productividad Real (h-p/PFSA)	Esfuerzo Real (h-p)
Histórico		1.00 (z)			3.50 (y)	
1	10.66 (a)	1.90 (x)	6.65(x*y/z)=(b)	70.89(a*b)		

Tabla 3-9: Cálculo del esfuerzo estimado para el Primer incremento

Suponiendo que el esfuerzo real empleado en el primer incremento fue de 85 horas-persona, se puede determinar que la productividad real fue de 7.97 horas-persona/PFSA, como esta información es propia del mismo equipo, permitirá estimar con más precisión el esfuerzo necesario para el siguiente incremento.

Considerando que en el primer incremento el equipo ganó experiencia en el desarrollo de aplicaciones y de la herramienta, entonces el EAF cambiaría a 1.20. Con esta información se podrá determinar el esfuerzo requerido para el segundo incremento

Para los siguientes incrementos, utilizando información obtenida en los incrementos anteriores, se utiliza la siguiente fórmula.

$$Productividad\_Estimada\_Incr(i) = \frac{EAF(i)x\ ProductividadReal\_Incr(i-1)}{EAF(i-1)} \quad (5)$$

Incremento	PFSA	EAF	Productividad Estimada (h-p/PFSA)	Esfuerzo Estimado (h-p)	Productividad Real (h-p/PFSA)	Esfuerzo Real (h-p)
Histórico		1.00			3.50	
1	10.66 (a)	1.90 (x)	6.65	70.89	7.97 (c/a)=d	85.00 (c)
2	13.33	1.20 (w)	5.04 (w*d/x)	67.13		

Tabla 3-10: Cálculo del esfuerzo estimado para el segundo incremento

Asimismo, se puede utilizar el promedio de las productividades reales obtenidas en los anteriores incrementos, como se muestra en la siguiente fórmula.

$$ProductividadEstimada\_Incr(i) = EAF(i)x\frac{\sum_{j=1}^{i=1}\frac{ProductividadReal\_Incr(j)}{EAF(j)}}{i=1} \quad (6)$$

$$PFSA\_Incr(i) = \sum_{k=1}^{n} PFSA\_CU(k) \quad (7)$$

$$EsfuerzoEstimado\_Incr(i) = PFSA\_Incr(i)\ x\ Productividad\ Estimada\_Incr(i) \quad (8)$$

Evaluación de resultados de la construcción del incremento (actividad 6)

Incremento	PFSA	EAF	Productividad Estimada (h-p/PFSA)	Esfuerzo Estimado (h-p)	Productividad Real (h-p/PFSA)	Esfuerzo Real (h-p)	MRE
Histórico		1.00			3.50		
1	10.66	1.90	6.65	70.89	7.97	85.00	16.60
2	13.33	1.20	5.04	67.13	6.15	82.00	18.13

Tabla 3-11: Cálculo del MRE para cada incremento

Para determinar el MRE, se aplica la siguiente fórmula (Conte, 1986):

$$MRE = \frac{|y - \hat{y}|}{y}$$

(y=esfuerzo real, yˆ =esfuerzo estimado)

Analizando el resultado del MRE (Magnitud del error relativo) se debe considerar lo siguiente (Hastings, April 2001):

- MRE ≤ 20% indica que la técnica puede ser considerada predictiva y puede ser empleada con confianza para la planificación;
- 20% < MRE ≤ 50% indica que es aceptable, pero debe ser empleada con precaución;
- MRE > 50% indica que no es confiable para propósitos de planificación.

Si la diferencia entre esfuerzo estimado y real sea mayor a 20%, se debe evaluar el porqué de esta diferencia. Algunas causas podrían ser un cambio en el contexto de la construcción del incremento, por ejemplo cambios de personal, que no fueron considerados al inicio del incremento. Para estos casos, se deberá tomar en cuenta el EAF actualizado que considere estos cambios para así poder estimar el esfuerzo requerido para construir el siguiente incremento (Pow Sang Portillo, 2012).

Puede producirse diferencias significativas entre el esfuerzo real y estimado del primer incremento, debido a que la información tomada podría corresponder a un proyecto en el que ha participado otro equipo de proyecto. No obstante, las estimaciones de los siguientes incrementos deberán ser muy cercanas a los esfuerzos reales, porque es información propia del mismo proyecto (Pow Sang Portillo, 2012).

En la tabla 3-11 el MRE entre el esfuerzo estimado y esfuerzo real es menor al 20%, lo que significa que Tupuy puede ser considerado como una técnica de estimación, que puede ser utilizado con confidencialidad para propósitos de planeamiento.

3.4 Conclusiones del estado del arte

La estimación de costos de software es un problema complejo, que sigue atrayendo considerable atención a los investigadores (Bozhikova, 2010), siendo el esfuerzo de software su componente principal. De la revisión de los artículos presentados se

puede observar diversos estudios dirigidos a desarrollar métodos y herramientas de estimación de costos de software; así como indicadores de medición de los mismos.

En las técnicas presentadas en (Heričko, 2008) y (Kang, 2010) se definen reglas para estimar el esfuerzo entre los incrementos o iteraciones. Sin embargo, ninguna de las técnicas define la forma de dividir el número de puntos de función entre iteraciones o incrementos, o considera que la productividad del equipo puede cambiar debido a varios factores, tales como la rotación, la experiencia de desarrollo de software, el conocimiento de las herramientas y lenguajes a través de todo el proyecto, y por último no define como priorizar la construcción de los casos o los requisitos de uso (Pow-Sang J. I., 2012). Es por ello, que el Dr. Pow Sang propuso un enfoque denominado Tupuy, el cual 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. Esta propuesta está conformada por tres técnicas: UML2FP, Diagrama de Precedencia de Casos de uso (UCPD) e Incremental-FP (Pow Sang Portillo, 2012).

Por otro lado, es importante contar con estimaciones precisas de proyectos de software para maximizar su calidad y minimizar los costos en la gestión de proyectos de software iterativos. En (Hastings T. S., 2001) se muestra los márgenes de error que son aceptables y no aceptables para predecir el esfuerzo de desarrollo temprano en el ciclo de vida del software para propósitos de planificación de + / -20 por ciento a través de una gama de tipos de aplicaciones.

Finalmente, para evaluar la confiabilidad de la Técnica Tupuy, se aplicará en forma práctica en proyectos de software con ciclo de vida incremental y paradigma orientados a objetos, para examinar si arroja los mismos resultados al compararla con pruebas previas.

CAPITULO IV: APLICACIÓN PRÁCTICA TUPUY

La aplicación práctica de la técnica Tupuy se realizó sobre base histórica obtenida de los alumnos de pregrado de la especialidad de Ingeniería Informática, matriculados en el curso de Ingeniería de Software en el año 2009 en Pontificia Universidad Católica del Perú (PUCP). En este curso los alumnos desarrollaron un proyecto de software de un sistema de información para una cadena de hoteles que recién iniciaba su funcionamiento. La duración del proyecto fue de 14 semanas que es la duración de un semestre académico de la PUCP.

Los alumnos desarrollaron los siguientes sistemas de hoteles: Hotel 1, Hotel 2 y Hotel 3, y en base a la información elaborada por ellos (Especificaciones de requisitos de software, Diagrama de clases, Diagrama de secuencia, etc) se aplicó la técnica Tupuy, con la finalidad de realizar el cálculo de Puntos de función empleando modelos orientados a objetos (UML2FP), la priorización de casos de uso para su construcción (UCPD) y el esfuerzo requerido en cada incremento para un modelo de ciclo de vida incremental (Incremental–FP). Seguidamente, se aplicó la Magnitud del error relativo (MRE) y finalmente, se analizaron los resultados para determinar si la técnica es confiable para propósitos de planificación.

4.1 Los Equipos de trabajo

Los alumnos fueron divididos en grupos de 11 o 12 personas. Fue necesario que cada integrante del proyecto se capacite en las siguientes actividades:

- Análisis y diseño de lenguaje orientados a objetos.
- Introducción a RUP
- Lenguaje de programación Visual Basic.Net 2008
- MS Visio (Afianzar)
- MS SQL Server 2005.

4.2 Características del proyecto

Se desarrolló un software cliente-servidor con 3 capas: Capa de Presentación, Capa de Negocio y Capa de Acceso a Datos, con el siguiente alcance preliminar:

- Administrar hoteles (número de pisos, servicios ofrecidos, tipo y número de habitaciones, etc.)

- Administrar reservas de habitaciones
- Administrar pagos
- Administrar servicios al huésped (alimentación, llamadas telefónicas, lavandería, traslados aeropuerto, etc.)
- Administrar personal y turnos de trabajo
- Administrar compras
- Administrar promociones
- Administrar eventos (conferencias, reuniones, etc.)

Respecto al ambiente de desarrollo se consideró:

- SO Cliente: Windows XP.
- Entorno de programación: Visual Basic. Net.
- Manejador de Bases de Datos: MS SQL Server.

El desarrollo de software se definió por un modelo incremental en iteraciones, cuyas fases se mencionan a continuación en las tablas 4-1 y 4-2:

Fase	Iteraciones	Inicio	Fin
Concepción	1	Semana 1	Semana 5
Elaboración	1	Semana 6	Semana 10
Construcción	3	Semana 11	Semana 15

Tabla 4-1: Semanas por fases del proyecto

Fase	Iteraciones	Fecha Inicio	Fecha Fin
Concepción	1	23 Marzo	16 Abril
Elaboración	1	16 Abril	7 Mayo
Construcción	3	7 Mayo	25 Junio

Tabla 4-2: Cronograma por fases del proyecto

Todos los equipos realizaron tres iteraciones de la fase de construcción, es decir; tres incrementos, cada uno de los cuales tuvieron una duración aproximadamente de dos semanas.

4.3 Recopilación de datos

A los alumnos se les proporcionó un formato de hoja de cálculo donde registraron las horas utilizadas para realizar cada actividad por día de la semana y el grado de avance en porcentaje de los casos de uso.

Sólo se ha considerado las horas correspondientes al trabajo efectivo realizado para la construcción del software (diseño, programación y pruebas).

	Ingrese el esfuerzo en horas que ha utilizado para realizar cada una de las actividades en cada día de la semana							
		Semana						1
	Actividad	16/03/2008	17/03/2008	18/03/2008	19/03/2008	20/03/2008	21/03/2008	22/03/2008
1	Reuniones de coordinación (internas)	0	0	0	0	1	2	0
2	Reuniones de control de proyecto (con JP y/o profesor)	0	0	0	0	1	1	0
3	Elaboración de informes (no incluye ningún tipo de plan)	0	0	0	0	0	0	0
4	Actividades de gestión de proyectos (edt, riesgos, gantt, plan del proyecto, plan de iteración).	0	0	0	0	0	0	0
5	Determinación de requisitos (catálogo de requerimientos, ERS).	0	0	0	0	0	0	0
6	Determinación de la arquitectura del sistema	0	0	0	0	0	0	0
7	Preparación del plan de pruebas.	0	0	0	0	0	0	0
8	Diseño de interfaz de usuario	0	0	0	0	0	0	0
9	Diseño (no interfaz de usuario), programación y pruebas unitarias	0	0	0	0	0	0	0
10	Integración y pruebas de integración del software	0	0	0	0	0	0	0
11	Elaboración de manual de usuario	0	0	0	0	0	0	0
	TOTAL	0	0	0	0	2	3	0
		Acum. Semanal						5

Tabla 4-3: Hoja de cálculo para registrar horas trabajadas.

El cronograma de entregas del proyecto para la fase de construcción fue el siguiente:

Fecha	Tema
Jue 7 May.	Entregable 4: (Elaboración) **Plan de proyecto** **Documento de estimación,** **Documento de arquitectura del software,** **Prototipo de arquitectura de software.**
Jue 28 May.	Entregable 5: (Construcción-Iteración 1) **Diseño y Plan de Pruebas**
Jue 04 Jun.	Entregable 6: (Construcción-Iteración 1) **Programación**
Jue 11 Jun.	Entregable 7: (Construcción-Iteración 2) **Diseño y Plan de Pruebas**
Jue 18 Jun.	Entregable 8: (Construcción-Iteración 2) **Programación**
Jue 25 Jun.	Entregable 9: (Construcción-Iteración 3) **Diseño y Plan de Pruebas**
Jue 25 Jun.	Entregable 10: (Construcción-Iteración 3) **Producto Final (instaladores, fuentes, versiones de todos los artefactos) en CD.** **Revisión del avance de programación y pruebas**

Tabla 4-4: Cronograma de entregas del proyecto en la fase de construcción

4.4 Caso de Estudio

El presente caso de estudio propone aplicar la técnica Tupuy para los siguientes sistemas de hoteles: Hotel 1, Hotel 2 y Hotel 3, desarrollado por alumnos de pregrado de la especialidad de Ingeniería de informática, matriculados en el curso de Ingeniería de Software en el año 2009 en Pontificia Universidad Católica del Perú (PUCP).

Antes de comenzar la fase de construcción, cada equipo elaboró el documento de especificación de Requerimientos de Software (ERS) con casos de uso., después los estudiantes utilizaron el Diagrama de Precedencias de Casos de Uso (UCPD) para determinar las secuencias de construcción de los mismos para cada incremento.

Para el modelamiento de los casos de uso, los estudiantes utilizaron como herramienta StarUML, y en base al diagrama de clases de análisis de UML se aplicó las reglas para determinar los puntos de función sin ajustar (PFSA) utilizando el componente adicional Tupux. Asimismo; cada equipo elaboró los siguientes documentos: Plan de proyecto, Documento ERS, Documento de Visión y Documento de Arquitectura. En función de los cuales se realizaron las siguientes actividades:

- ✓ Se determinó los casos de uso del Hotel 1 por incremento, debido a que en el archivo (StarUML) que se recibió estaba modelado por paquete (Administración, servicios y Marketing).

- ✓ Se determinó el cálculo de las horas por persona para cada incremento en los tres proyectos.

- ✓ Se determinó los Puntos de Función sin ajustar para el cálculo de la estimación, obtenidos al aplicar la herramienta Tupux.

- ✓ Se determinó el valor del EAF para los tres proyectos, en base al contexto de los mismos, los cuales están sujetas a cambios de una iteración a otra (el conocimiento de la plataforma de desarrollo, integración del equipo de desarrollo, entre otros).

- ✓ Se determinó el esfuerzo estimado por cada incremento utilizando Incremental – FP mediante Puntos de función sin ajustar (PFSA) y el factor de ajuste del esfuerzo de COCOMO (EAF).

- ✓ Se determinó el MRE (margen del error relativo) para cada incremento.

- ✓ Finalmente se analizaron los resultados obtenidos del cálculo del esfuerzo estimado para Hotel 1 y Hotel 2, según las tablas 4.6 y 4.8.

A continuación se realizó el análisis de cada uno de los proyectos:

HOTEL 1

1. El punto de partida para la aplicación práctica, consistió en determinar los PFSA de Hotel 1 por incremento, debido a que en el archivo (StarUML) que entregaron los alumnos estaba por paquete (Administración, servicios y Marketing), tal como se muestra en la figura 4-1.

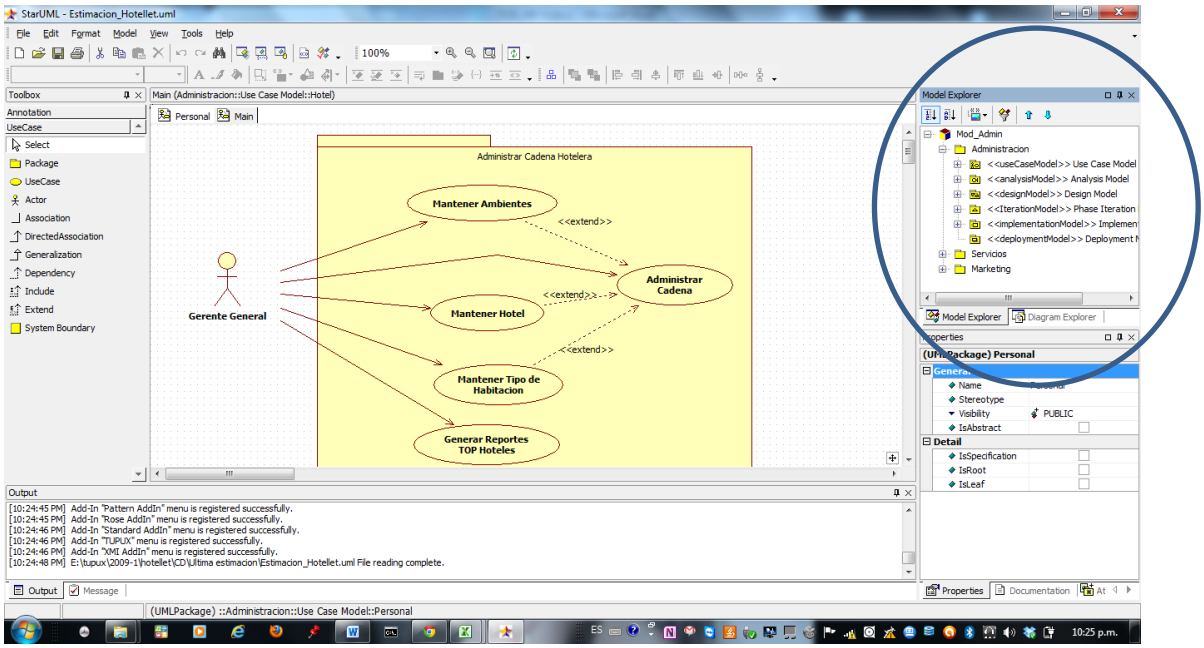

Figura 4-1: Casos de Uso por Paquete Hotel 1 – Elaborado por los alumnos

En base al plan del proyecto y el documento de ERS (Especificación de Requisitos de Software), elaborado por los alumnos, y considerando los entregables del 6 al 9 mencionados en la tabla 4-4, se determinaron los casos de uso para cada uno de los tres incrementos como se muestra en la figura 4-2:

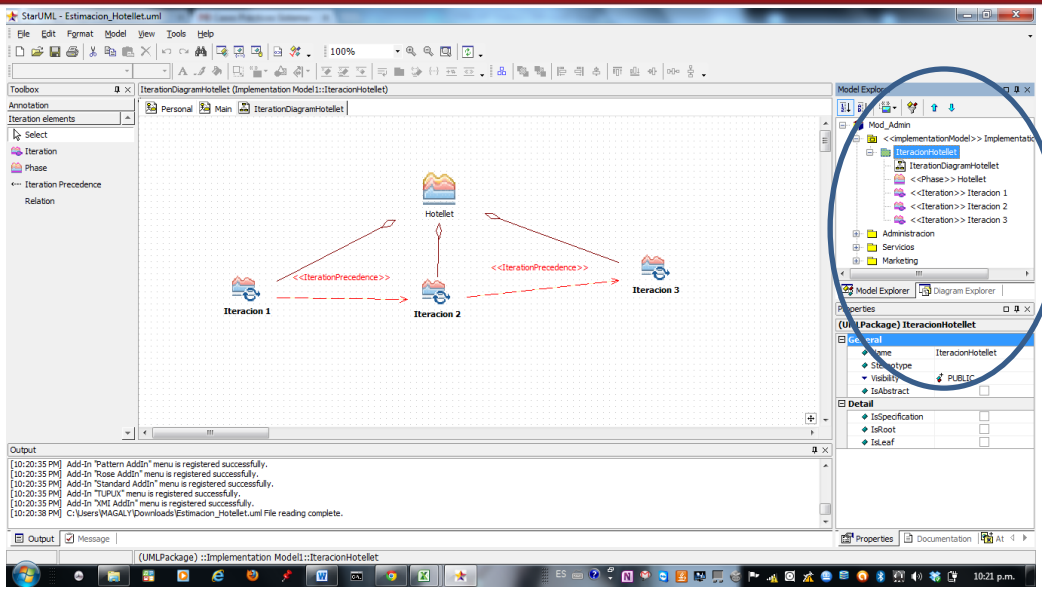

Figura 4-2: Casos de Uso por Incremento Hotel 1

2. Se determinó las horas para cada incremento:

En base al formato de hoja de cálculo preparado por los alumnos según tabla 4-3, se registraron las horas que utilizaron para realizar cada actividad por día de la semana y grado de avance en porcentaje de los casos de uso. Según la Tabla 4-1: Semana por fases del proyecto, sólo se consideró las semanas comprendidas desde la semana 11 hasta la 15.

En la tabla 4-5 se muestra a continuación el resumen de las horas utilizadas por los alumnos por cada incremento.

NOMBRE	Horas- Persona 1 era Iteración (18-May a 28 May)	Horas-Persona 2 da Iteración (29-May a 11-Jun)	Horas-Persona 3 era Iteración (12-Jun a 25-Jun)	Horas-Persona TOTAL
Persona 1	13.00	41.00	43.00	97.00
Persona 2	9.00	19.00	35.00	63.00
Persona 3	5.00	26.00	14.00	45.00
Persona 4	27.00	39.00	33.00	99.00
Persona 5	44.00	38.00	30.00	112.00
Persona 6	23.00	42.00	46.00	111.00
Persona 7	8.00	24.00	16.00	48.00
Persona 8	27.00	34.00	13.00	74.00
TOTAL	**156.00**	**263.00**	**230.00**	**649.00**

Tabla 4-5: Horas por persona para cada incremento Hotel 1

Una vez determinados los casos de uso y el cálculo de las horas por persona para cada incremento, se aplicó la técnica Tupuy, conformada por las técnicas: UML2FP, UCPD y por último la técnica Incremental–FP.

- **Aplicación de UML2FP**

Utilizando la herramienta Tupux, el cual es un componente adicional de StarUML que permite calcular los Puntos de función sin ajustar (PFSA), se determinó los DETs y RETs por Fichero y por Transacción, utilizando el diagrama de clases de análisis.

En el documento de arquitectura, elaborado por los alumnos, se muestra el diagrama de clases de análisis del sistema de gestión del Hotel 1. Parar mayor facilidad y orden, los alumnos dividieron por cada módulo en los que se agrupan sus funcionalidades: Módulo de Administración, Módulo Marketing y Módulo Servicios.

- **Aplicación de Diagrama de precedencias de casos de uso (UCPD)**

Los alumnos utilizaron el Diagrama de precedencias de casos de uso para seleccionar que caso de uso se debe desarrollar en cada incremento.

Para determinar los Puntos de Función sin ajustar (PFSA) por cada incremento, se determinaron los casos de uso para cada incremento, debido a que el archivo (StarUML) que entregaron los alumnos estaba por paquete (Administración, servicios y Marketing).

- **Aplicación de Incremental –FP**

La Técnica Incremental-FP indica que se deben realizar tres actividades: estimar el tamaño de cada caso de uso (actividad 3), definir incrementos y esfuerzo requerido (actividad 4) y evaluar resultados de la construcción del incremento (actividad 6) señaladas en la figura 3-4.

Aplicando las fórmulas 1, 2 y 3 señaladas en la sección 3.3.3 para la estimación del tamaño de cada caso de uso (actividad 3) , y utilizando la herramienta Tupux, se calculó los Puntos de Función sin ajustar (UFP) para cada incremento tal como se muestra a continuación en la figura 4-3:

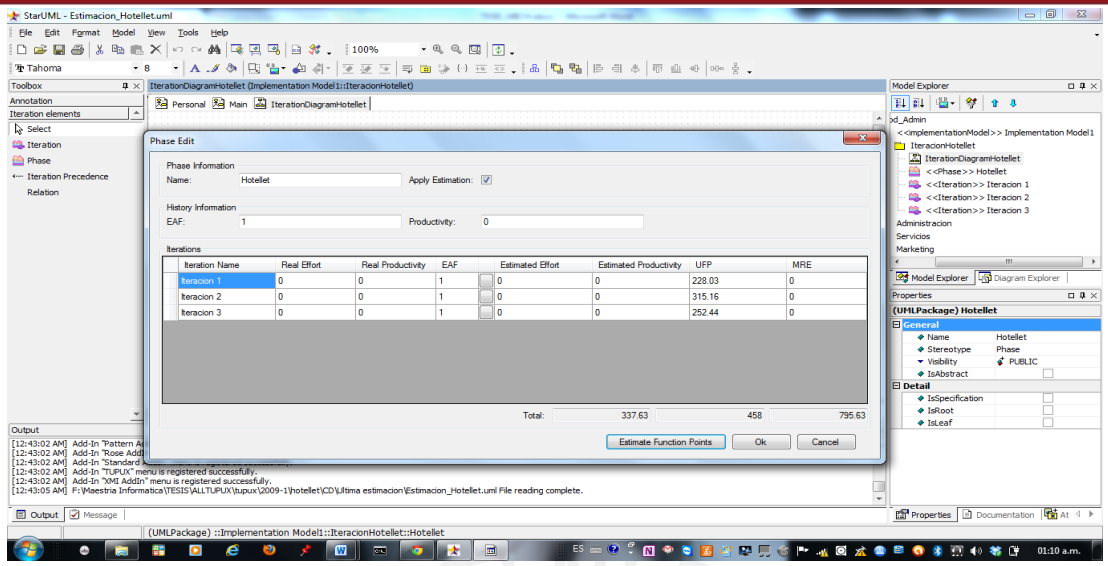

Figura 4-3: Cálculo de Puntos de función sin ajustar HOTEL 1 – Elaborado por los alumnos

Seguidamente, se aplicó las fórmulas 4 al 8 señaladas en la sección 3.3.3 para la definición de incrementos y esfuerzo requerido (actividad 4).

Finalmente se realizó la evaluación de resultados de la construcción del incremento (actividad 6) aplicando la fórmula del MER, señalada en la sección 3.3.3.

En la tabla 4-6 se muestra a continuación el cálculo de esfuerzo estimado para Hotel 1.

Incremento	PFSA	EAF	Productividad Estimada (h-p/PFSA)	Esfuerzo Estimado (h-p)	Productividad Real (h-p/PFSA)	Esfuerzo Real (h-p)	MRE
1	228.03	1.00			0.68	156.00	
2	315.16	1.00	0.68	215.61	0.83	263.00	18.02
3	252.44	1.00	0.76	191.68	0.91	230.00	16.66

Tabla 4-6: Esfuerzo Estimado HOTEL 1

HOTEL 2

En este proyecto se realizó las siguientes actividades en base a la información elaborada por los alumnos:

1. Al igual que en el proyecto anterior se determinó las horas para cada incremento en base al formato de hoja de cálculo mostrado en la tabla 4-3.

En la tabla 4-7 se muestra a continuación el resumen de las horas utilizadas por los alumnos por cada incremento.

NOMBRE	Horas- Persona 1 era Iteración (07-May a 28 May)	Horas-Persona 2 da Iteración (29-May a 11-Jun)	Horas-Persona 3 era Iteración (12-Jun a 25-Jun)	Horas-Persona TOTAL
Persona 1	20.00	10.00	27.00	57.00
Persona 2	13.00	19.00	17.00	49.00
Persona 3	11.00	18.00	14.00	43.00
Persona 4	14.00	46.00	50.00	110.00
Persona 5	11.00	25.00	48.00	84.00
Persona 6	24.00	38.00	46.00	108.00
Persona 7	14.00	46.00	45.00	105.00
Persona 8	30.00	38.00	18.00	86.00
Persona 9	17.00	35.00	36.00	88.00
Persona 10	26.00	26.00	35.00	87.00
TOTAL	**180.00**	**301.00**	**336.00**	**817.00**

Tabla 4-7: Horas por persona para cada incremento Hotel 2

Seguidamente se procedió aplicar la técnica Tupuy, conformada por las técnicas: UML2FP, UCPD y la técnica Incremental–FP.

- **Aplicación de UML2FP**

Utilizando la herramienta Tupux, se determinó los DETs y RETs por Fichero y por Transacción, utilizando el diagrama de clases de análisis, tal como se observa en la figura 4-4.

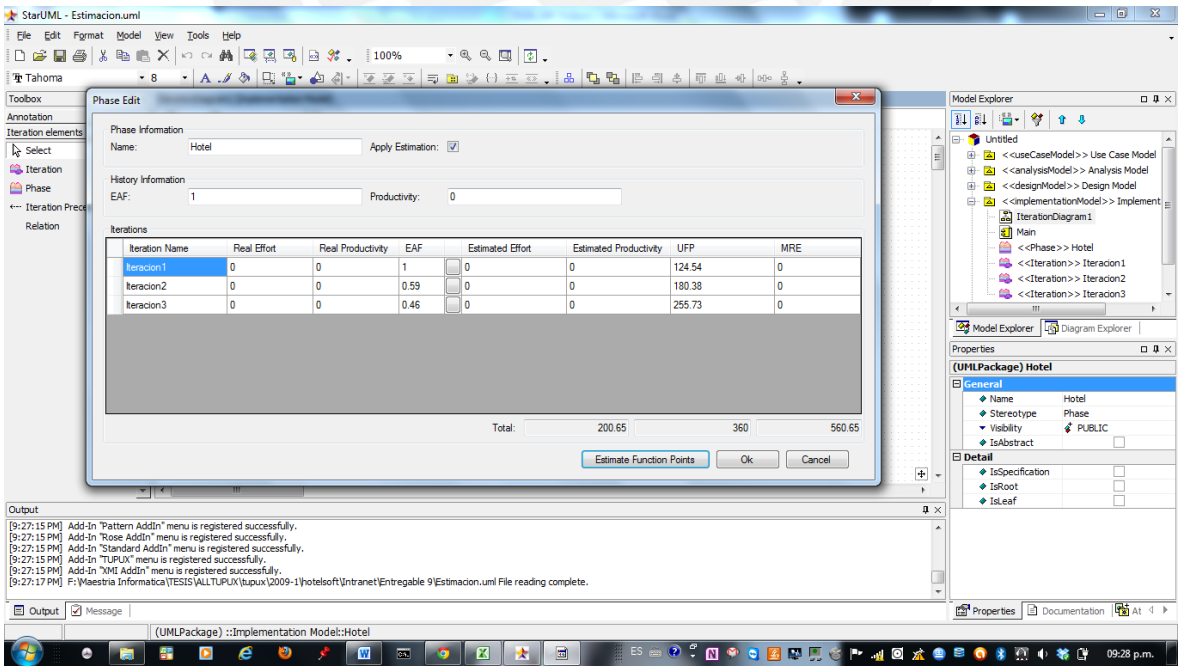

Figura 4-4: Cálculo de Puntos de función sin ajustar HOTEL 2 – Elaborado por los alumnos

- **Aplicación de Diagrama de precedencias de casos de uso (UCPD)**

Los alumnos utilizaron el Diagrama de precedencias de casos de uso para seleccionar que caso de uso se debe desarrollar en cada incremento. Esta información se encontró en el Documento de diseño – Diagrama de Precedencias, elaborada por ellos, como se muestra en la figura 4-5.

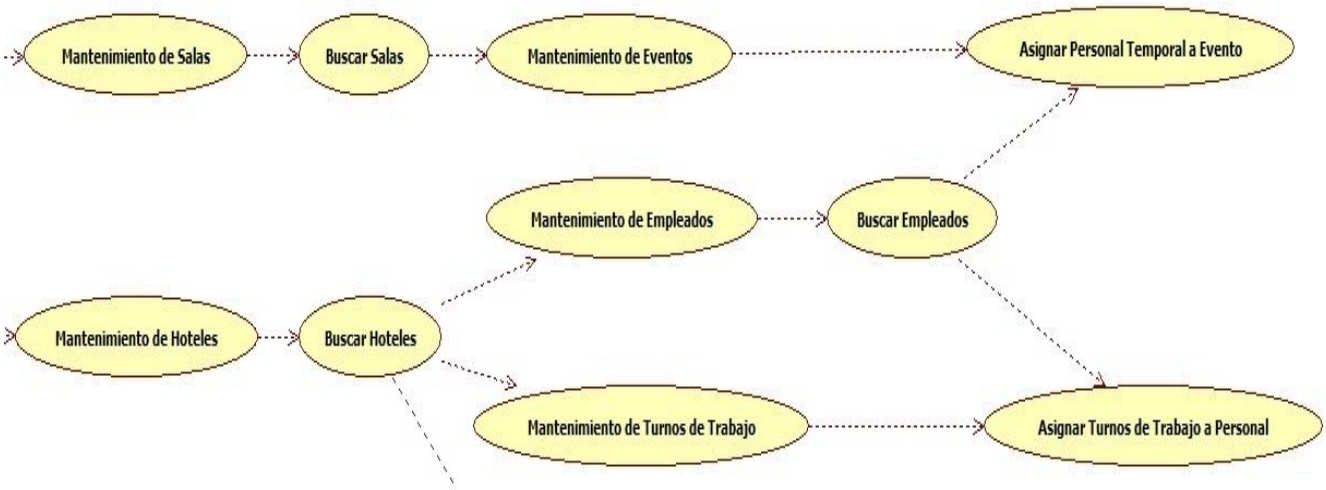

Figura 4-5: Diagrama de Precedencias HOTEL 2 – Elaborado por los alumnos

- **Aplicación de Incremental –FP**

De la misma manera que en el proyecto anterior se aplicó la técnica Incremental-FP conformada por tres actividades: estimar el tamaño de cada caso de uso (actividad 3), definir incrementos y esfuerzo requerido (actividad 4) y evaluar resultados de la construcción del incremento (actividad 6) señaladas en la figura 3-4.

Aplicando las fórmulas 1, 2 y 3 señaladas en la sección 3.3.3 para la estimación del tamaño de cada caso de uso (actividad 3) y utilizando la herramienta Tupux, se calculó los Puntos de Función sin ajustar (UFP) para cada incremento.

Seguidamente, se aplicó las fórmulas 4 al 8 señaladas en la sección 3.3.3 para la Definición de incrementos y esfuerzo requerido (actividad 4).

Finalmente se realizó la evaluación de resultados de la construcción del incremento (actividad 6) aplicando la fórmula del MER, señaladas en la sección 3.3.3

En la tabla 4-8 se muestra a continuación el cálculo de esfuerzo estimado para Hotel 2.

Incremento	PFSA	EAF	Productividad Estimada (h-p/PFSA)	Esfuerzo Estimado (h-p)	Productividad Real (h-p/PFSA)	Esfuerzo Real (h-p)	MRE
1	124.54	1			1.45	180.00	
2	180.38	1	1.45	260.71	1.67	301.00	13.39
3	255.73	1	1.56	398.17	1.31	336.00	18.50

Tabla 4-8: Esfuerzo Estimado HOTEL 2

HOTEL 3

Finalmente se analizó el presente proyecto realizando las siguientes actividades:

1. Se determinó las horas para cada incremento en base al formato de hoja de cálculo tal como se muestra en la tabla 4-9. Cabe mencionar que la información recibida de las horas estaba incompleta para el tercer incremento.

NOMBRE	Horas- Persona 1 era Iteración (18-May a 28 May)	Horas-Persona 2 da Iteración (29-May a 11-Jun)	Horas-Persona 3 era Iteración (12-Jun a 24-Jun)	Horas-Persona TOTAL
Persona 1	5.00	21.00	22.00	48.00
Persona 2	8.00	30.00	27.00	65.00
Persona 3	8.00	28.00	18.00	54.00
Persona 4	8.00	27.00	10.00	45.00
Persona 5	14.00	30.00	24.00	68.00
Persona 6	10.00	26.00	12.00	48.00
Persona 7	8.00	30.00	14.00	52.00
Persona 8	14.00	27.00	33.00	74.00
Persona 9	7.00	25.00	51.00	83.00
TOTAL	**82.00**	**244.00**	**211.00**	**537.00**

Tabla 4-9: Horas por persona para cada incremento HOTEL 3

Por otro lado al revisar la información del StarUML en el diagrama de clases de análisis y la documentación del sistema de gestión de Hotel 3. Se pudo observar lo siguiente:

1. No se consideró en el conteo de los PFSA para el tercer incremento las siguientes transacciones :

- Reporte Ocurrencia.
- Reporte de Servicios por Cliente.
- Registrar Pago.

- Reporte de Ingresos y Egresos.
- Registrar Pedido.
- Registrar Orden de Compra.
- Registrar Entradas y Salidas de Almacén.
- Generar reporte de eventos.

2. Por otro lado, en la documentación proporcionada por los alumnos se encontraron reportes que deberían ser considerados para el conteo del Puntos de función sin ajustar:

- Modelo_reporte_eventos
- Prototipo Reporte de Servicios por Cliente
- ReporteIngresosEgresos

3. Se ha completado los DETs de las transacciones del ILF (Evento). Ver Figura 4-6.

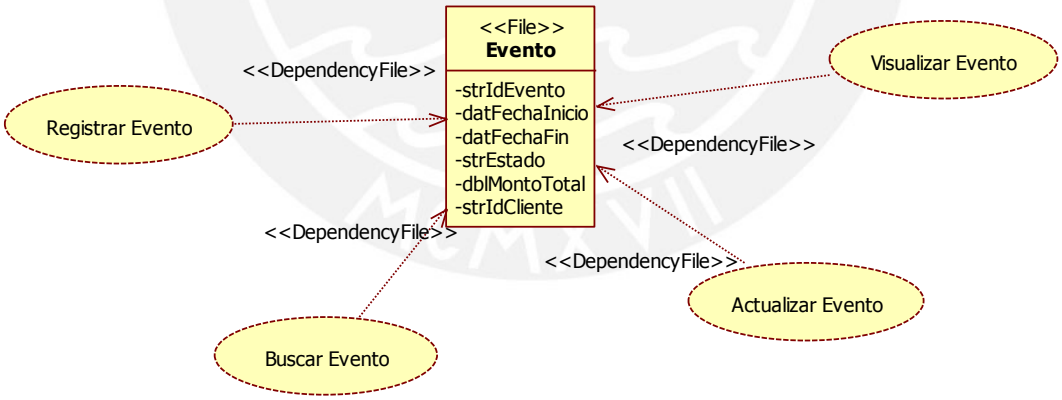

Figura 4-6: Diagrama de Transacciones HOTEL 3 – Elaborado por los alumnos

En la Figura 4-7 se muestra el cálculo de los puntos de función sin ajustar por cada incremento. Cabe mencionar que los puntos de función sin ajustar no son confiables debido a que no se consideró reportes y transacciones en el conteo de los puntos de función sin ajustar.

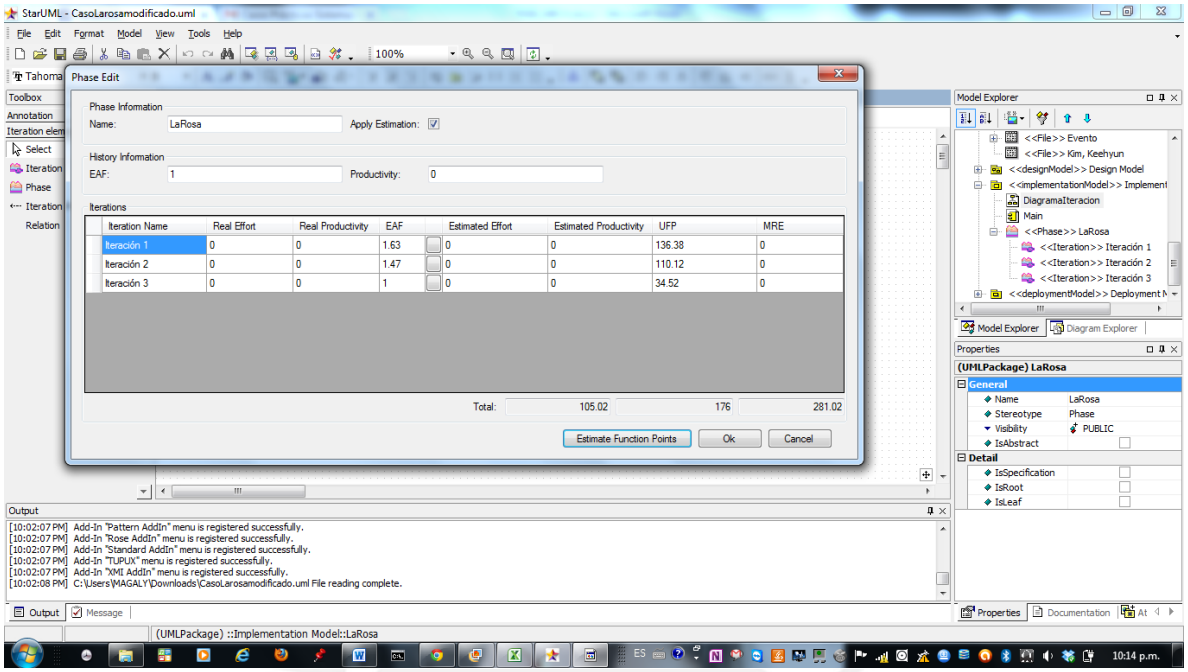

Figura 4-7: Cálculo de Puntos de función sin ajustar HOTEL 3 – Elaborado por los alumnos

4.5 Discusión

El propósito principal de este trabajo es determinar la confiabilidad de la Técnica Tupuy, al aplicarlo en forma práctica en Proyectos de software con ciclo de vida incremental y paradigma orientado a objetos. Los resultados se presentan a continuación:

Se puede observar que de la información entregada por los equipos para los proyectos Hotel 1, Hotel 2 y Hotel 3, se consideró apropiado sólo realizar el cálculo de la estimación para los proyectos Hotel 1 y Hotel 2, debido a que Hotel 3 contaba con información incompleta (horas por persona) necesaria para nuestro cálculo.

Se calculó la estimación del esfuerzo a partir del segundo incremento según tablas 4.6 y 4.8, porque la estimación para el primer incremento no sería ni confiable ni precisa, debido a que se emplearía información de semestres anteriores y esta es información no es propia de los equipos, por lo que la diferencia entre los valores estimados y reales podría ser alta.

Los estudiantes no habían utilizado la herramienta de programación antes de la construcción del primer incremento. Al inicio del segundo incremento, los estudiantes sabían cómo programar con la herramienta (Visual Basic.Net 2008).

Para asignar un valor al EAF, necesario para calcular el esfuerzo estimado, se consideró que la experiencia del lenguaje y la experiencia de la aplicación no varían en el contexto de cada iteración, siendo su valor nominal o promedio de 1.00 para los tres incrementos.

En la tabla 4-10 se puede observar que en el segundo incremento el MRE es 18.02 para Hotel 1 y 13.39 para Hotel 2, lo que significa que el esfuerzo estimado se acerca más a la realidad para el caso del Hotel 2, caso contrario ocurre en el tercer incremento donde el MRE es 16.66 para Hotel 1 y 18.50 para Hotel 2, siendo el esfuerzo estimado que se acerca más a la realidad para el caso del Hotel 1.

Para los resultados mostrados en la tabla 4-10, se utilizó la Magnitud del Error Relativo (MRE) aplicando la siguiente fórmula (Conte, 1986):

$$MRE = \frac{|y - \hat{y}|}{y}$$

(y=esfuerzo real, yˆ =esfuerzo estimado)

	Esfuerzo Estimado (h-p)		Horas-Persona Reales		MRE (Margen de error relativo)	
Incremento	**Hotel 1**	**Hotel 2**	**Hotel 1**	**Hotel 2**	**Hotel 1**	**Hotel 2**
2	215.61	260.71	263	301	18.02	13.39
3	191.68	398.17	230	336	16.66	18.50

Tabla 4-10: Cuadro resumen de esfuerzo estimado, esfuerzo real y MRE-HOTEL 1 y HOTEL 2

Aplicando lo que señala (Hastings, April 2001) relacionado la Magnitud del Error Relativo (MRE) se debe considerar lo siguiente:

Si el MRE ≤ 20%, indica que la técnica puede ser considerada predictiva y puede ser empleada con confianza para la planificación.

Por otro lado, (Carmines, 1979) señala que los procedimientos de medición deben poseer dos propiedades fundamentales: fiabilidad y validez. "En primer lugar, se puede examinar la fiabilidad de un indicador. Fundamentalmente, las preocupaciones de confiabilidad el grado en que un experimento, prueba, o cualquier procedimiento de medición arroja los mismos resultados en pruebas repetidas [...] el más consistente de los resultados proporcionados por las mediciones repetidas, mayor será la fiabilidad del procedimiento de medida.

Se confirma el resultado presentado en (Pow-Sang J. I., 2012) donde se muestra el resultado mediante la aplicación del enfoque denominado Incremental-FP, que permite estimar el esfuerzo para cada incremento utilizando puntos de función sin ajustar y el EAF de Cocomo, en el proyecto realizado con los estudiantes de pregrado, obteniendo buenos resultados, ya que la diferencia entre el esfuerzo estimado y verdadero esfuerzo fue menor del 20% para el segundo incremento.

Finalmente de los resultados obtenidos del análisis de las tablas 4-6, 4-8 y 4-10, se observa que la diferencia entre el esfuerzo estimado y real medido con el MRE para todas las iteraciones son menores al 20% en los proyectos Hotel 1 y Hotel 2, lo que significa que la técnica puede ser utilizada con toda confianza para fines de planificación.

CONCLUSIONES Y TRABAJOS FUTUROS

En el presente trabajo se confirma lo propuesto en la tesis del Dr. Pow Sang, al comparar los resultados obtenidos con pruebas previas. En la tabla 4-10 se observa que la diferencia entre el esfuerzo estimado y real de los proyectos hotel 1 y hotel 2, medido con la Magnitud del Error Relativo (MRE) para todas las iteraciones, fueron menores al 20%. Por lo tanto, se puede utilizar la técnica Tupuy con toda confianza para fines de planificación. En el caso de estudio presentado se aplicó la técnica Tupuy de la siguiente manera:

- Primero se utilizó la técnica UML2FP para determinar los DETs y RETs por fichero y transacciones en los diagrama de clases de análisis utilizando el componente adicional Tupux para los hoteles 1 y 2. No se aplicó esta técnica para el caso del hotel 3 debido a que los alumnos no habían calculado los DETs el ILF (Evento) así como otras transacciones y reportes necesarios para el conteo de los puntos de función sin ajustar.

- Segundo se utilizó la técnica UCPD. Para el caso del hotel 1 se determinó los casos de uso para cada incremento, debido a que en el archivo (StarUML) que entregaron los alumnos estaba por paquete (Administración, servicios y Marketing). En el caso de los hoteles 2 y 3 los diagramas de precedencias fueron elaboradas por los alumnos.

- Finalmente se utilizó la técnica Incremental F-P. Tal como se observa en las tablas 4-6 y 4-8 se ha calculado la estimación del esfuerzo a partir del segundo incremento, porque la estimación para el primer incremento no sería ni confiable ni precisa, debido a que se emplearía información de semestres anteriores que no es propia de los equipos. Para el caso del hotel 3 no se utilizó esta técnica debido a que la información de las horas reales (esfuerzo real) estaba incompleta para la tercera iteración.

Aplicando lo que señala (Hastings, April 2001) relacionado a la Magnitud del error relativo (MRE), si el MRE ≤ 20%, indica que la técnica puede ser considerada predictiva y puede ser empleada con confianza para la planificación. De los resultados obtenidos en los casos Hotel 1 y Hotel 2 aplicando la técnica Tupuy, en proyectos con alumnos de pregrado de la especialidad de Ingeniería de informática, se puede observar que el MRE fue menor a 20%.

Algunos trabajos futuros que podrían basarse en los resultados de esta aplicación sería que la técnica puede ser validada en proyectos industriales, el impacto del número de iteraciones en el rendimiento de estimación y un análisis del factor EAF.

BIBLIOGRAFÍA

Ahmed, A. (2011). *Software Project Management: A Process-Driven Approach.* CRC Press.

The USC Center for Software Engineering. (s.f.). *Center for Systems and Software Engineering*. Recuperado el 28 de Junio de 2013

http://csse.usc.edu/csse/research/COCOMOII/cocomo_main.html

Abraha˜o, S. G. (2007). Experimental evaluation of an object-oriented function point measurement procedure. *Information and Software Technology*, 366-380.

Abrahão, S. P. (2006). A functional size measurement method for object-oriented conceptual schemas: design and evaluation issues. *Software \& Systems Modeling*, 48-71.

Albrecht, A. (1979). Measuring Applicaton Development Productivity. *IBM Corporation*, 83-92.

Awan, N. M. (2010). Predicting software test effort in iterative development using a dynamic Bayesian network. *School of Engineering, Blekinge Institute of Technology, MSc Thesis*.

Balbin, D. M.-S. (2009). TUPUX: An Estimation Tool for Incremental Software Development Projects. Washington, DC, USA: IEEE Computer Society.

Boehm, B. R. (2000). *Software cost estimation with Cocomo II.* Prentice Hall PTR.

Boehm, B., Clark, B., Horowitz, E., Westland, C., Madachy, R., & Selby, R. (1995). Cost models for future software life cycle processes: COCOMO 2.0. *Annals of software engineering*, 57-94.

Bozhikova, V. S. (2010). An approach for software cost estimation. *Proceedings of the 11th International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing on International Conference on Computer Systems and Technologies*, 119-124.

Carmines, E. G. (1979). *Reliability and validity assessment.* Vol. 17.

Center for Software Engineering at the University of Southern California (USC). (1995). COCOMO II Model Definition Manual- Version 2.1.

Chemuturi, M. C. (2010). *Mastering Software Project Management: Best Practices, Tools and Techniques.* J. Ross Publishing.

Cockburn, A. (2008). Using both incremental and iterative development. *Software Engineering Technology*.

Conte, S. D. (1986). *Software engineering metrics and models.* Redwood City, CA, USA: Benjamin-Cummings Publishing Co., Inc.

del Bianco, V. G. (2008). An evaluation of function point counting based on measurement-oriented models. *Evaluation and Assessment in Software Engineering--EASE*, 26-27.

Dimes, T. (2015). *Conceptos Básicos de Scrum: Desarrollo de Software Agile y Manejo de Proyectos Agile.* Babelcube Inc.

Fenton, N. E., & Pfleeger, S. L. (1998). *Software metrics: a rigorous and practical approach.* Boston, MA, USA: PWS Publishing Co.

Fetcke, T., Abran, A., & Nguyen, T.-H. (1997). Mapping the OO-Jacobson approach into function point analysis. *Technology of Object-Oriented Languages and Systems, 1997. TOOLS 23. Proceedings*, 192-202.

Garcia, C. A., & Hirata, C. M. (2008). Integrating functional metrics, COCOMO II and earned value analysis for software projects using PMBoK. *Proceedings of the 2008 ACM symposium on Applied computing*, 820-825.

Gutwenger, C. J. (2003). A new approach for visualizing UML class diagrams. *ACM*, 179-188.

Harput, V. K. (2005). Extending function point analysis to object-oriented requirements specifications. *Software Metrics, 2005. 11th IEEE International Symposium*, 10-pp.

Hastings, T. E. (April 2001). A Vector-Based Approach to Software Size Measurement and Effort Estimation. *IEEE Transactions on Software Engineering*, 337-350.

Heričko, M. (2008). The size and effort estimates in iterative development. *Information and Software Technology*, 772-781.

IFPUG: International Function Point User Group. (2009). *Manual de prácticas de medición de puntos de función (versión 4.2.1).* USA.

Ingunn, M. S. (2005). Reliability and validity in comparative studies of software prediction models. *Software Engineering, IEEE Transactions on 31.5*, 380-391.

Kang, S. C. (2010). Model-Based Dynamic Cost Estimation and Tracking Method for Agile Software Development. *IEEE*, 743-748.

Karkukly, W. (2012). *Managing the Pmo Lifecycle: A Step-by-Step Guide to PMO Set-Up, Build-Out, and Sustainability.* Trafford Publishing.

Keele, S. (2007). Guidelines for performing systematic literature reviews in software engineering. *EBSE Technical Report EBSE-2007-01*.

Kitchenham, B. A. (2001). What accuracy statistics really measure. *In Software, IEE Proceedings- (Vol. 148, No. 3, pp. 81-85). IET.*, 81-85.

Knoernschild, K. (2002). *Java Tm Design: Objects, UML and Process.* ADDISON WESLEY Publishing Company Incorporated.

Kurt Bittner, I. S. (2003). *Use Case Modeling.* EEUU: Addison-wesley .

Lavazza, L. A. (2008). Model-based functional size measurement. *Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement*, 100-109.

Leung, H. F. (2002). Software cost estimation. *Handbook of Software Engineering*.

Lo, B. G. (2007). Assessing software cost estimation models: criteria for accuracy, consistency and regression. *Australasian Journal of Information Systems*, vol. 5, no 1.

Magnus, C. I. (1992). *Object-Oriented Software Engineering.* Addison-Wesley.

Mishra, A. M. (2009). Some issues on scheduling estimation model for object-oriented software projects. *ACM SIGSOFT Software Engineering Notes*, 1-4.

Mishra, J. (2011). *Software Engineering.* Pearson Education India.

Mohagheghi, P., Bente, A., & Reidar, C. (2005). Effort estimation of use cases for incremental large-scale software development. *Software Engineering, 2005. ICSE 2005. Proceedings. 27th International Conference on. IEEE*, 303-311.

OMG Unified Modeling Language. (2008). *Object Management Group*. Recuperado el 24 de Enero de 2013, de Object Management Group: http://www.uml.org

Orr, G. R. (2000). Function point counting: one program's experience. *Journal of Systems and Software*, 239-244.

Paz, F. Z.-S. (2014). An Approach for Effort Estimation in Incremental Software Development Using Cosmic Function Points. *ACM*, 44:1--44:4.

Pender, T. (2003). *UML Bible.* John Wiley & Sons,.

Petticrew, M. R. (2008). *Systematic reviews in the social sciences: A practical guide.* John Wiley & Sons.

Pfleeger, S. L. (2002). *Ingeniería del software: teoría y práctica.* Buenos Aires,Argentina: Prentice Hall.

Pow Sang Portillo, J. A. (Marzo de 2012). Tesis Doctoral : Técnicas para la Estimación y Planificación de Proyectos de Software con Ciclos de Vida Incremental y Paradigma Orientado a Objetos. *Técnicas para la Estimación y Planificación de Proyectos de Software con Ciclos de Vida Incremental y Paradigma Orientado a Objetos.* Madrid.

Pow Sang, J. A. (2006). An Approach of a Technique for Effort Estimation of Iterations in Software Projects. *IEEE*, 367-376.

Pow-Sang, J. I. (2012). Effort Estimation in Incremental Software Development Projects Using Function Point. *Computer Applications for Software Engineering, Disaster Recovery, and Business Continuity*, 458-465.

Ribu, K. (2001). Estimating Object-Oriented Software Projects with Use Cases. *Master of Science Thesis*.

Roebuck, K. (2012). *Software Development Life Cycle (SDLC): High-impact Strategies - What You Need to Know: Definitions, Adoptions, Impact, Benefits, Maturity, Vendors.* Emereo Publishing.

Rolandostudios. (2012). Obtenido de http://www.youtube.com/user/rolandostudios

Royce, W. W. (1970). Managing the development of large software systems. *proceedings of IEEE WESCON*.

Ruhe, G. C. (2014). *Software Project Management in a Changing World.* Springer.

Ruparelia, N. B. (2010). Software development lifecycle models. *SIGSOFT Softw. Eng. Notes*, 8-13.

Sater Carstens, D. L. (2013). *Project Management Tools and Techniques: A Practical Guide.* CRC Press.

Sinhal, A. B. (2013). A Novel Fuzzy Based Approach for Effort Estimation in Software Development. *ACM*, 1--6.

Sommerville, I. (2005). *Ingeniría del Software.* Madrid: Pearson Educación.

Tan, T. L. (2009). Productivity trends in incremental and iterative software development. *Empirical Software Engineering and Measurement, 2009. ESEM 2009. 3rd International Symposium on*, 1 -10.

Trendowicz, A. (2014). *Software Project Effort Estimation: Foundations and Best Practice Guidelines for Success.* Springer.

Uemura, T. K. (2001). Function-point analysis using design specifications based on the Unified Modelling Language. *Journal of software maintenance and evolution: Research and practice*, 223-243.

Villanueva Bendezú, D. V. (2013). Validación de técnicas de estimación de esfuerzo en proyectos de software con ciclos de vida incremental y paradigma orientado a objetos. *Pontificia Universidad Católica del Perú, Escuela de Posgrado. Mención: Ingeniería de Software*.

Zhang, F., Ma, Z. M., Cheng, J., & Meng, X. (2009). Fuzzy semantic web ontology learning from fuzzy UML model. *ACM*, 1007-1016.

Živkovič, A. I. (2005). Automated software size estimation based on function points using UML models. *Information and Software Technology*, 881--890.