UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

ESTIMACIÓN DE PROYECTOS DE TECNOLOGÍA DE LA INFORMACIÓN

T E S I S

QUE PARA OBTENER EL TÍTULO DE

INGENIERO EN COMPUTACIÓN

P R E S E N T A N

DAFNÉ CASTILLO BLANCO
EDGAR JIMÉNEZ CASILLAS

DIRECTOR DE TESIS:
M.C. MA. JAQUELINA LÓPEZ BARRIENTOS

INDICE

Prólogo

En la mayoría de las empresas donde se produce software para apoyar el negocio, se aplican técnicas de administración que utilizan débilmente la planificación. En empresas cuyo giro es el desarrollo de software, existe una creciente necesidad de formalizar los mecanismos de estimación de proyectos.

La realización de estimaciones adecuadas sobre el tamaño y esfuerzo requerido es una de las características fundamentales de un proyecto de desarrollo de software exitoso (de hecho, de cualquier proyecto). Por otro lado, las malas estimaciones o comúnmente las no estimaciones, son posiblemente una de las principales causas de los fracasos, por lo menos del incumplimiento de plazos y presupuestos en la entrega de aplicaciones de software.

En muchas organizaciones (o en la mayoría) la no existencia de un proceso estructurado para la preparación de estimaciones de tamaño, esfuerzo, costo y calidad dificulta la definición de equipos de trabajo de magnitud y perfil adecuado así como plazos de cumplimiento razonables. Además, la falta de registros de proyectos realizados en el pasado contribuye fuertemente a esa situación de descontrol en cuanto a la determinación del esfuerzo y plazo requerido para cumplir con un determinado proyecto.

La estimación de tiempos y personal requerido para el desarrollo de sistemas es el elemento esencial de las dificultades que se presentan en el control de los proyectos de software.

Una estimación es una predicción basada en un modelo probabilístico, no un modelo determinístico; es decir, la cantidad que se está estimando puede tomar no solamente un valor sino distintos valores, teniendo algunos de ellos mayor probabilidad de ser correctos.

Los pasos básicos en el proceso de estimación de proyectos de software son los siguientes:

- Estimación del tamaño del producto que se va a desarrollar o a implementar.
- Estimación del esfuerzo requerido para el desarrollo del proyecto en horas-hombre. Del esfuerzo surge comúnmente el costo, ya que típicamente el costo de la mano de obra será el costo principal en los proyectos de desarrollo.
- Estimación del plazo de realización del proyecto, normalmente en meses.

Actualmente, para la estimación del tamaño de un proyecto existen distintas metodologías, las más comunes son: Juicio Experto (Expert Judgement), Líneas de Código (Lines of Code), Top-Down, Bottom-Up y Análisis de Puntos de Función (Function Points Analysis, FPA).

Function Points Analysis es una técnica estructurada y objetiva reconocida ante el ISO, para medir el tamaño del software cuantificando la funcionalidad desde el punto de vista del usuario. Se basa en los requerimientos del usuario, en el diseño lógico y en cómo el usuario interactúa con el sistema.

Algunas de las ventajas de esta metodología son que ayuda a comparar entre herramientas, ambientes o lenguajes para determinar cuáles son los más productivos para implementar el proyecto y que facilita las negociaciones con el cliente y entre los miembros de la organización al hablar en términos de funcionalidad, costo y calidad de todo el proyecto y no de aspectos técnicos. Por otra parte requiere de mucho trabajo manual ya que el proceso de conteo no se puede automatizar y se necesita tener personal con cierto grado de experiencia en las reglas de conteo pues éstas podrían resultar complejas.

El presente trabajo de tesis tiene como objetivo principal demostrar que Function Points Analysis es la mejor metodología disponible en la actualidad para la estimación del tamaño de los proyectos de desarrollo de software.

Para lograr dicho objetivo, la tesis está divida en 2 grandes bloques:

El primero, compuesto por los capítulos 1 y 2, nos lleva desde la definición de conceptos básicos de administración de proyectos como:

medir, planear, métrica, etc. hasta el análisis de las metodologías más relevantes para la estimación del tamaño de proyectos de software.

El segundo, compuesto por los capítulos 3, 4 y 5, expone la metodología de Function Point Analysis, desde su historia, definición de conceptos básicos, usos y beneficios, las reglas que la definen hasta su aplicación en la medición de un sistema para el control de ingresos de la compañía Mexicana de Aviación.

Finalmente, en las conclusiones, cerramos el tratamiento del tema exponiendo a FPA como la mejor metodología de estimación de tamaño y como una vía hacia el aseguramiento de la calidad en los proyectos de software.

CAPÍTULO 1

INTRODUCCIÓN

1. INTRODUCCIÓN

1.1. Antecedentes

¿Qué es medir?

Es relacionar una magnitud con otras que se consideran patrones universalmente aceptados, estableciendo una comparación de igualdad, de orden y de número.

¿Qué es planear?

Es fijar el curso concreto de acción que ha de seguirse, estableciendo los principios que habrán de orientarlo, la secuencia de operaciones para realizarlo y las determinaciones de tiempo y números necesarios para su realización.

Una planeación debe basarse en hechos y no en emociones vagas y genéricas, para ello los pronósticos y la investigación realizados cuidadosamente son elementos clave.

La planeación busca prever el número y tipo de recurso (material y humano) que se necesitará en cada puesto y/o actividad.

1.1.1. Necesidad del hombre de planear sus actividades

El hombre desde que aparece sobre la superficie de la Tierra tiene a su favor la posición erecta, el desplazamiento bípedo, la oposición del dedo pulgar y el raciocinio, características que le permiten tener una visión de más de 180 grados, emplear las extremidades superiores y razonar para experimentar, construir y elaborar instrumentos que le ayudan a hacer la vida más plácida al aprovechar los elementos que le proporciona el medio en que vive.

Haciendo uso de estas cualidades que lo distinguen de cualquier especie descubrió el fuego y fue perfeccionando los utensilios de trabajo; observó los astros y su posición periódica, llevando registro por medio de dibujos primero y después empleando el lenguaje escrito; aplicó el conocimiento adquirido a la agricultura y empezó a medir el tiempo y el espacio.

Ya con el empleo del calendario y la selección del lugar idóneo para vivir, el producto de sus recolecciones, cosecha, caza y pesca se incrementó y con la abundancia se multiplicaron los grupos humanos cuyos excedentes de producción eran cambiados en trueque.

Inicialmente todo era de todos pero al evolucionar cambió también la organización social. Con el esclavismo surge la propiedad privada y con ella la posesión de bienes y su venta; se inicia el comercio y con la cuantificación de productos fue necesario acordar las medidas que emplearían en sus transacciones.

El primer intento del hombre por establecer una medida que fuera reconocida en todas partes fue el ocupar su propio cuerpo, pero esta práctica resultó deficiente y no equitativa ya que variaba de acuerdo a la persona que se tomara como modelo.

A partir del Feudalismo se experimenta la necesidad de marcar los límites territoriales de cada feudo, apareciendo la medición de longitudes y el empleo de la moneda como unidad de intercambio.

El comercio, antes local, se amplió, lo que obligó a los mercaderes a la adopción de unidades de medida equivalentes.

A partir de ese momento dejaron de intercambiar sólo el excedente y comenzaron a producir para obtener mayores ganancias. Bajo estas condiciones necesitaron modificar los métodos de producción, crear nuevas rutas comerciales más cortas y más seguras así como nuevos centros de almacenaje y comercio. Se estaban aplicando así los principios de administración y planeación comerciales.

Al analizar esta evolución histórica podemos afirmar que la planeación y la medición son actividades inherentes al hombre y van perfeccionándose junto con él.

El avance tecnológico de hoy en día le proporciona herramientas cada vez más eficaces para realizar mediciones más exactas, pudiendo no sólo planear para controlar su propia vida sino todo lo que le interese.

1.1.2. Necesidad de la industria de planear, medir y controlar.

La capacidad de una organización para conservar su poder competitivo y lograr tasas de crecimiento depende en gran parte de la planeación de sus actividades, del desarrollo de programas, nuevos productos y servicios y de la adopción de estrategias adecuadas.

La necesidad de planear, esencialmente se deriva del hecho de que toda organización opera en un medio que experimenta constantes cambios: tecnológicos, políticos, competitivos, en las actitudes o normas sociales y en la actividad económica, derivados del fenómeno de globalización.

En este complejo contexto ambiental y de incertidumbre surge la necesidad de dirigir organizaciones bajo ideas y conceptos de la planeación concebida como un proceso dinámico y sistemático basado en una actitud y una forma de vida que requiere dedicación para identificar oportunidades y peligros que podrían surgir con el objeto de tomar decisiones en el presente para aprovechar de la mejor manera las oportunidades y evitar los peligros.

En otras palabras, la planeación es la determinación racional de a dónde queremos ir y cómo llegar allá, es una relación entre fines y medios.

El propósito de la planeación es determinar lo que debe hacerse esta semana, mes o año, para estar en una situación satisfactoria la próxima semana, mes o año. No se relaciona con futuras decisiones sino con el impacto futuro de presentes decisiones. Debe trabajarse con el respaldo de objetivos y determinar lo que ha de realizarse para alcanzarlos en tiempo determinado.

La planeación no intenta eliminar el riesgo, pero asegura que los riesgos sean tomados en el tiempo correcto. Intenta asegurar el uso efectivo de los recursos disponibles que conduzcan al logro de los objetivos.

Los cambios registrados durante las últimas décadas hicieron crecer la complejidad de los problemas de la administración al mismo ritmo que la actividad económica e industrial, de tal suerte que los empresarios deben reflexionar sobre el conjunto de cambios acelerados que conforman su entorno, la administración de los mismos es un desafío y una oportunidad que debe ser afrontada con un método y ese método se llama planeación.

Una organización debe planear los esfuerzos que le permitan alcanzar los resultados deseados.

La industria necesita medir, planear y controlar porque el logro de objetivos es un resultado del orden, no puede venir del azar ni de la improvisación. Todo control es imposible si no se compara con un plan previo. Sin plan, se trabaja con los ojos vendados.

Ninguna empresa puede organizar, ejecutar y controlar con éxito a menos que antes haya planeado.

1.1.3 Importancia que ha adquirido el área de Sistemas en cualquier industria

Los sistemas de información no son nuevos. Mucho antes de la automatización de las computadoras las compañías reunían, almacenaban y actualizaban la información en lo que era el curso normal de hacer negocios.

En el pasado como ahora, los sistemas de información consistían en los procedimientos y reglas establecidas para entregar información a la gente dentro de una organización.

Diferentes personas requieren distinta información para realizar su trabajo y las reglas del sistema gobiernan qué información debería ser distribuida a cada persona, cuándo y en qué formato.

En grandes compañías, la automatización de las tareas de negocios impulsó el desarrollo de departamentos por separado para servir a los sistemas de información por computadora recién surgidos. Inicialmente, esos departamentos y la gente que trabajaba en ellos estaban aislados del resto de las operaciones de una compañía. Tales departamentos estaban creando sistemas que reunían datos del nivel de operaciones y la convertían en información para los administradores. El surgimiento de la PC y su aceptación casi universal cambió esos departamentos junto con los sistemas a los que servían.

Conforme otras personas que no eran administradores se convertían en trabajadores de la información, los departamentos de sistemas de información comenzaron a servir a organizaciones enteras y se convirtieron en parte integral de la operación de un negocio.

El tamaño del departamento de sistemas de información de una compañía se relaciona generalmente con el de la compañía a la que da apoyo. En compañías muy grandes estos departamentos pueden emplear cientos o incluso miles de personas.

Con frecuencia las grandes compañías contratan como consultores externos a trabajadores independientes o compañías para que les provean servicios especializados (como el desarrollo de un sistema).

Una empresa bien conformada basa su estructura organizacional en una buena planeación y distribución de sus recursos contando con áreas específicas que contribuyen en el proceso de elaboración del producto o prestación del servicio.

El área de sistemas ha cobrado una gran importancia en las empresas debido a que la computadora se ha convertido en una herramienta indispensable para la ejecución de cualquier tarea y también porque día a día se crean nuevos programas que automatizan actividades repetitivas o peligrosas para el hombre.

En empresas cuyo giro principal no es el de la tecnología de la información, el contar con un área de sistemas eficiente que le permita a la organización poner todo su esfuerzo en la elaboración y comercialización de su producto o servicio, es prioritario ya que le brindará a todos los

usuarios de la organización desde soporte técnico hasta el desarrollo de herramientas personalizadas que les resolverán necesidades específicas.

La tendencia actual de asignarle más recursos a las áreas que proporcionan mayor valor al giro principal de la empresa ha hecho que aquellas áreas fuera de esa razón de ser del negocio estén pasando a ser sustituidas por servicios de outsourcing.

1.1.4 Necesidad de medir el software

El negocio de desarrollo de software es un negocio relativamente joven y la falta de medición es simplemente un reflejo del nivel de madurez de la perfección del software. Cuando analizamos otras profesiones tales como contabilidad, leyes o medicina, las cuales han estado en el mercado por un período de tiempo más largo, vemos que han establecido medidas y estándares que se han convertido en parte del negocio.

Sin embargo, el negocio del desarrollo del software está madurando y la medición del software está siendo reconocida lentamente como parte del proceso; ejemplo de ello es el desarrollo de estándares internacionales tales como el ISO 9001.

Debido a que nos encontramos en las primeras etapas del proceso de maduración de esta profesión, el punto fundamental alrededor de la expansión y utilización de la medición del software como una herramienta viable en el proceso de desarrollo, es la necesidad de incrementar el nivel de conocimiento entre los desarrolladores, administradores de proyectos y otros.

Antes de que podamos explicar por qué se necesita medir el software debemos identificar quién necesita de esta medición.

Usualmente las actividades de medición del software se dividen en tres niveles, ver Tabla 1.1

Nivel	Audiencia	Parámetros a medir del Software				
3	Administración de alto nivel	Enfoque: Tiempo para el mercadeo		Satisfacción del Cliente		Ahorro de costos
2	Mandos medios	Enfoque: Productividad		Monitoreo de costos	Eficiencia	Desempeño
1	Administrador de proyecto	Enfoque: Puntos de función	Esfuerzo	Defectos	Calendario de actividades	Cumplimiento de acuerdos

Tabla 1.1 Categorías de las actividades de medición del software

En el primer nivel se tiene al administrador del proyecto, quien es responsable de asegurar las herramientas correctas y los métodos disponibles para administrar y controlar el ambiente.

Un administrador de proyectos debe tener la habilidad de estimar con precisión y controlar el software entregable, para lo cual requiere mediciones bien definidas, incluyendo el tamaño de lo que se va a entregar, calendarios de trabajo, defectos, costos relativos al desarrollo del proyecto (esfuerzo), cumplimiento de estándares y administración de peticiones de cambios.

Las predicciones del proyecto basadas en datos de mediciones que anticipan los retrasos en las fechas de entrega, gastos no programados y baja calidad, necesitan ser parte de la ecuación de valor agregado que un buen programa de medición de software provee.

En la actualidad, muchas organizaciones se limitan a capturar estos datos, analizarlos superficialmente y tomar decisiones. En cambio, un programa proactivo de medición captura estos datos y los guarda en un repositorio junto con otros datos esenciales del proyecto. El experto en estimación del software desarrolla modelos que son utilizados por el administrador del proyecto en futuros desarrollos para predecir resultados.

En resumen, la necesidad de medir el software a nivel del administrador del proyecto radica en la habilidad de manejar proactivamente los entregables usando una base de datos histórica de mediciones.

El segundo nivel incluye aquellos parámetros requeridos por mandos medios, líderes del negocio y responsables de los entregables, quienes comparten las necesidades del administrador del proyecto y cuya actividad está enfocada en el desempeño general más que en el avance diario.

Las mediciones que proporcionan valor a los mandos medios son: productividad, eficiencia y desempeño. Lo que usualmente no miden es si la aplicación realmente cubre las necesidades del usuario; más bien se preocupan de que esté lista para usarse, que la instalación sea fácil, que funcione correctamente y en la posibilidad de pasarle el control al área de soporte.

La administración de alto nivel (capa 3 de la Tabla 1.1) está compuesta principalmente por los siguientes puestos:

- Chief Information Officer (CIO).
  Gerente de Tecnología de Información (TI)

- Chief Executive Officer (CEO).
  Gerente Ejecutivo

- Chief Operating Officer (COO).
  Gerente de operaciones.

Los CEO’s y otros ejecutivos de alto nivel tienen un gran deseo por entender y cuantificar la contribución del área de TI al negocio.

Hoy más que nunca TI es reconocida como parte integral de la estrategia de negocio de cualquier empresa.

El desempeño de las actividades de medición en TI, que se han llevado a cabo en el pasado para mostrar la productividad y calidad de los resultados, deben evolucionar en mediciones más sofisticadas y orientadas

al negocio. A los CIO's les corresponde demostrar su contribución al negocio usando estas nuevas mediciones.

Los datos de las mediciones ayudan al administrador del proyecto a ser proactivo, es decir, no tiene que esperar a que sucedan hechos para actuar y tomar decisiones sino con la información que le proporcionan las mediciones puede predecir una posible desviación en el curso planeado del proyecto y evitar consumir más tiempo y recursos.

En casi cualquier industria madura se han desarrollado métodos que permiten evaluar el producto o servicio ofrecido: en la industria automotriz el número de autos producidos por mes, en la aviación el número de pasajeros transportados etc.

En la industria del software se ha carecido de un método que permita evaluar el tamaño de lo que producimos o a lo que le damos soporte. Por lo tanto la capacidad de planeación se ve mermada al no contar con datos históricos o estadísticos que ayuden al momento de dimensionar los costos y el tiempo de desarrollo del nuevo producto.

Como se muestra en la Tabla 1.1, Function Points es una herramienta utilizada a nivel de la administración del proyecto para determinar el tamaño del desarrollo y es parte del proceso general de medición del software.

Function Points viene a cubrir esa necesidad de lo que las industrias requieren sobre definir el tamaño de lo que se está produciendo.

CAPÍTULO 2

SITUACIÓN ACTUAL DE PRÁCTICAS DE ESTIMACIÓN DE PROYECTOS DE SOFTWARE

2. Situación actual de prácticas de estimación de proyectos de software

2.1 Métricas: conceptos, usos y beneficios

Tan sólo medir no es suficiente. El conjunto de datos de tamaño, esfuerzo, tiempo y defectos es sólo eso, un conjunto de datos. Para que una compañía de TI mejore, debe usar esta información para determinar las mejores practicas, mejoras a sus modelos de proceso, establecer benchmark, analizar tendencias, mejorar la estimación, educar a loc clientes y mantener informada a la organización, desde los administradores de proyectos hasta los desarrolladores.

Una ***medida*** es un número que asigna un valor relativo. Combinaciones de una o más medidas producen una ***métrica***, como por ejemplo promedio de entregas (horas / FP) o densidad de defectos (número de defectos / FP). Las métricas son estándares con las cuales un proceso o producto puede ser evaluado objetivamente.

Para maximizar el éxito de un programa de métricas, una organización debería seguir los siguientes lineamientos:

- Integrar las métricas dentro de los procesos existentes
- Hacer las métricas parte de la cultura de la empresa
- Recolectar información de cada proyecto y resumirla para conformar el archivo histórico de la organización
- Seleccionar solamente unas cuantas métricas para la implementación inicial
- Utilizar métricas estándares de la industria para facilitar las comparaciones con otras organizaciones
- Establecer un repositorio de métricas
- Usar un programa de medición simple y consistente
- Usar las métricas para apoyar la toma de decisiones, establecer los alcances y objetivos y modelar la mejora de procesos
- Comunicar los resultados rápida y apropiadamente
- Incrementar la información brindada a la administración de alto nivel

- Seleccionar al equipo del programa de medición basándose en sus habilidades y no en su disponibilidad
- Capacitar a todos aquellos involucrados en el programa de medición
- Evolucionar y cambiar las métricas junto con la organización
- Promover las métricas dentro de la organización

Las métricas pueden ser usadas para identificar las mejores prácticas[1], modelar mejoras en los procesos, establecer un punto de referencia con el cual determinar tendencias, asistir en el proceso de planeación y estimación del proyecto, administrar el presupuesto, identificar la cantidad y calidad de los productos entregados, comparar la efectividad y eficiencia de los procesos y herramientas actuales, proveer una base para comparar entre industrias y habilitar una mejor comunicación entre clientes y desarrolladores.

CRITERIO DE SELECCIÓN DE MÉTRICAS

Las métricas seleccionadas por la organización para su propio programa de medición deben: ser métricas estándares de la industria, cumplir con las metas del programa de medición, ser claramente definibles y de fácil compresión, deben poder ser útiles para todos los niveles, deben ser realistas, medibles y acordes a las necesidades del cliente y las necesidades de desarrollo de la organización.

2.1.1 CATEGORÍAS DE MÉTRICAS

Al comenzar con un programa de medición dentro de la organización, se debe utilizar pocas métricas que sean fáciles, precisas y consistentes de recolectar. Las siguientes categorías de métricas son recomendadas para el inicio de un programa:

Productividad

Productividad es la medida de la eficiencia de un proceso que consume entradas y produce salidas. En el ambiente de la TI, la productividad puede ser medida usando la siguiente fórmula:

[1] Las mejores prácticas son un conjunto de sugerencias que describen la forma óptima de realizar una actividad y que se ha comprobado da mejores resultados. Estas prácticas son dictadas por los expertos en la materia.

$$\text{Pr}\,oductividad = \frac{Esfuerzo\ en\ horas}{Tamaño\ en\ FP's}$$

El retrabajo es otra medida de productividad y puede ser calculada como sigue:

$$Porcentaje\ de\ retrabajo = \frac{Horas\ de\ retrabajo}{Horas\ totales\ del\ proyecto}$$

Una organización debe proveer una definición exacta de retrabajo, de tal manera que aquellas horas sean capturadas de la misma forma en toda la organización. El retrabajo es el proceso de rehacer, corregir o descartar un entregable de un ciclo de vida completado previamente. No se considera retrabajo un cambio en los requerimientos del cliente.

Calidad

Es una medida del proceso del software durante el ciclo de vida del desarrollo y el producto del software entregado.

El proceso del software puede ser determinado por la siguiente fórmula:

$$Eficiencia\ de\ remoción\ de\ defectos = \frac{\begin{array}{c}Núm.\ total\ de\ defectos\ encontrados\\ antes\ de\ la\ entrega\end{array}}{\begin{array}{c}Núm.\ total\ de\ defectos\ descubiertos\\ (antes\ y\ después\ de\ la\ entrega)\end{array}}$$

El producto de software[2] puede ser medido por la siguiente fórmula:

$$Defectos\ de\ producción = \frac{Total\ de\ defectos\ de\ producción}{FP's\ entregados}$$

Existen múltiples métricas de calidad pero las dos anteriores se recomiendan para un programa inicial debido a que se pueden colectar

[2] Los defectos del producto de software deben ser colectados durante un período finito después de la instalación del producto, teniendo en cuenta la duración de los procesos que automatiza el sistema y considerando que cada proceso debe ser analizado al menos una vez en producción.

fácilmente. Sin embargo, otra métrica de calidad puede ilustrar un problema que tiene un impacto significativo no sólo en la calidad del producto sino también en el costo y tiempo en el mercado: Modificación del alcance.

La métrica para las modificaciones en el alcance del proyecto mide la cantidad en FP de revisiones de la especificación funcional que ocurren durante el proceso de desarrollo, refleja la funcionalidad agregada, cambiada o eliminada y es monitoreada durante el ciclo de vida de desarrollo.

Las modificaciones al alcance deben ser capturadas en cada requerimiento de cambio después de entregados los requerimientos iniciales y se calculan como sigue:

$$Modificaciones\ al\ alcance = \frac{\sum (FP's\ agregados + cambiados + \text{eliminados})}{FP's\ contados\ originalemte}$$

Costo

Es uno de los factores más importantes al decidir seguir con un proyecto. Los costos que sobrepasan el presupuesto inicial pueden causar que un proyecto se detenga antes de ser terminado o en el último de los casos resulta en una insatisfacción del cliente.

La eficiencia del costo se puede medir con la siguiente fórmula

$$Eficiencia\ del\ \text{cos}to = \frac{Costo\ actual}{FP's\ entregados}$$

Tiempo

Es usado para desarrollar calendarios y para determinar los recursos necesarios para terminar un proyecto en la fecha concertada.

El tiempo puede afectar la calidad del producto. Las modificaciones al alcance pueden afectar negativamente el tiempo estimado para el proyecto. Una organización puede monitorear el tiempo y utilizarlo para

evaluar el impacto de varios factores en la fecha de entrega actual del proyecto o para asistir en estimar futuros proyectos.

La duración de un proyecto en meses se puede medir como sigue:

Fecha de Fin del Proyecto – *Fecha de Inicio del* Proyecto

2.1.2 MEDIDAS CLAVE

FP's (tamaño), horas (esfuerzo), defectos, meses - calendario (duración), dólares (costo) son los elementos clave para un programa de medición inicial. Pueden ser combinados de varias formas para producir las métricas descritas en la sección anterior.

Una organización debe definir estas medidas claramente y capturarlas efectivamente. Si estas medidas son defectuosas las métricas resultantes también lo serán.

Function Points

El análisis por FP's es un método estandarizado para medir la funcionalidad del software desde el punto de vista del usuario. Se inclina por el diseño lógico más que por el diseño físico y así puede ser usado independientemente de la tecnología utilizada para la implementación.

Los beneficios de usar FP's para medir un proyecto van desde su independencia de los lenguajes, plataformas y otras características físicas, usarlos durante las diferentes fases del ciclo del vida del proyecto, incorporarlos con otras medidas hasta usarlos para hacer comparaciones entre industrias (benchmarking).

Información más detallada de los usos y beneficios de FP's pueden encontrarse en el capítulo 3 de esta tesis.

Horas

Es crítico para cada proyecto dar seguimiento preciso al consumo de tiempo, incluyendo los tiempos extra.

El tiempo deberá ser monitoreado cuidadosamente por horas gastadas en el trabajo original y por horas gastadas en el retrabajo. Adicionalmente los miembros del equipo de trabajo no deben incluir en sus planes de trabajo cualquier tiempo gastado en tareas de mantenimiento

Defectos

Un defecto es una diferencia entre el resultado esperado y el resultado obtenido. Una manera de medir la calidad es registrar el número de defectos.

Una organización puede determinar el impacto de detecciones tardías de costo tiempo y productividad monitoreando los defectos a través del ciclo de vida del proyecto (en ambas fases: cuando el defecto fue introducido y cuando el defecto fue encontrado).

La meta de toda organización debe ser identificar y corregir defectos en las primeras etapas del proyecto.

Meses - Calendario

La duración del proyecto se determina por el número de meses consumidos desarrollando un proyecto. Sólo los meses activos deben ser incluidos. La organización debe determinar la definición de las fechas de inicio y terminación del proyecto.

Típicamente la fecha de inicio de un proyecto es cuando comienza la fase de requerimientos y la fecha de terminación de un proyecto es cuando se instala en producción.

Dólares

El dinero gastado en un proyecto incluye: salarios del equipo de desarrollo, hardware o software adquirido para el proyecto y cualquier gasto externo como consultorías y capacitación.

Atributos del Proyecto

A diferencia de las medidas descritas anteriormente los atributos de un proyecto son “información suave” recabada para ayudar a definir las características del mismo. Como mínimo se necesita saber si se trató de una mejora o un desarrollo nuevo, si fue un producto comprado o hecho en casa y qué plataformas de desarrollo, producción y lenguajes de programación fueron usados.

Dentro de los atributos del proyecto podemos mencionar la metodología de desarrollo, experiencia del equipo de trabajo, uso de consultorías, técnicas de revisión e inspección, métodos de prueba. Los atributos ayudan a identificar los mejores procesos y aquellos que van en detrimento del éxito del proyecto. La organización necesita determinar los atributos que aplican dentro de su ambiente.

EL EQUIPO DE MEDICIÓN

Para tener éxito, cualquier programa de medición necesita tener el apoyo de los ejecutivos y directivos de la organización. Inicialmente se encontrará resistencia del equipo de trabajo pero esto puede ser superado con el respaldo de los directivos.

Es recomendable que durante el conteo se tenga la asesoría de un experto en FP’s, un experto en el negocio y un experto en desarrollo (estos tres roles pueden ser desempeñados por la misma persona), esto con el objetivo de tener un conteo más preciso.

Comúnmente el experto en FP’s es una persona certificada por la IFPUG. El experto en el negocio es el administrador del proyecto y el experto en desarrollo es el líder del equipo de codificación.

La tabla 2.1 resume las métricas básicas para comenzar un programa de mediciones.

Categoría	Nombre	Fórmula	Medida
Productividad	Productividad	$\text{Pr}\,oductividad = \frac{Esfuerzo\ en\ horas}{Tamaño\ en\ FP's}$	Hrs/FP
Productividad	% de retrabajo	$Porcentaje\ de\ retrabajo = \frac{Horas\ de\ retrabajo}{Horas\ totales\ del\ proyecto}$	%
Calidad	Eficiencia de remoción de defectos	$Eficiencia\ de\ remoción\ de\ defectos = \frac{Núm.\ total\ de\ defectos\ encontrados\ antes\ de\ la\ entrega}{Núm.\ total\ de\ defectos\ descubiertos\ (antes\ y\ después\ de\ la\ entrega)}$	NA
Calidad	Defectos de producción	$Defectos\ de\ producción = \frac{Total\ de\ defectos\ de\ producción}{FP's\ entregados}$	Defectos/ FP
Calidad	Modificaciones al alcance	$Modificaciones\ al\ alcance = \frac{\sum'(FP's\ agregados + cambiados + eliminados)}{FP's\ contados\ originalmente}$	NA
Costo	Eficiencia del costo	$Eficiencia\ del\ costo = \frac{Costo\ actual}{FP's\ entregados}$	USD/FP
Tiempo	Duración	$Fecha\ de\ Fin\ del\ Proyecto - Fecha\ de\ Inicio\ del\ Proyecto$	MESES

TABLA 2.1 Métricas Básicas del Software

2.2 Análisis de metodologías

Estimar el tiempo y el esfuerzo necesarios para desarrollar o modificar un sistema es una de las tareas más susceptibles de error en la ingeniería de software.

La estimación de software es compleja por varias razones, dentro de las cuales se encuentran las siguientes:

- Estimaciones prematuras.
  No podemos estimar lo que no entendemos.

- Falta de datos históricos relevantes.
  Todas las estimaciones son proyecciones del pasado al futuro.

- Fallas al calibrar.
  Todas las técnicas de estimación requieren ser calibradas a las condiciones locales.

- Falta de un proceso de estimación.
  Los procesos de estimación deben definirse y seguirse.

- Falla en el manejo de las estimaciones.
  Condiciones preescritas en una estimación no deben ser violadas si se pretende alcanzar el resultado de la predicción.

- Falla al actualizar estimaciones.
  Los proyectos de software no son entidades estáticas.

Debido a que el desarrollo de software y sus modificaciones son actividades orientadas a grupos con trabajo intelectual intenso, la mayoría de las técnicas de estimación se enfocan en estimar el esfuerzo requerido para desarrollar o modificar el software así como en establecer rangos de factibilidad para negociar recursos y tiempo. Este punto resulta sumamente importante debido a que los recursos y el tiempo no pueden ser arbitrariamente modificados en la ecuación de esfuerzo, donde:

Esfuerzo = Recursos x Tiempo

A partir de ésta podemos deducir por qué se requiere mayor cantidad de recursos (humanos y tecnológicos) para desarrollar un proyecto en menos tiempo.

Como regularmente sucede, lo que aumenta es la cantidad de trabajo (Esfuerzo) y se mantiene fijo el personal asignado para su desarrollo (Recursos), lo cual nos lleva a mover las fechas de entrega (Tiempo) para poder cubrir las nuevas necesidades, ya que el hecho de aumentar el personal no siempre es factible por cuestiones de costos.

En cualquier proyecto existen actividades críticas a las cuales no se les pueden modificar las fechas de entrega aún y cuando se asigne más personal para dicha tarea. Recordemos que "*un ser humano se gesta en nueve meses; es imposible que nueve mujeres puedan gestar al mismo ser en un mes*".

Las prácticas administrativas actuales parten del hecho de que cuentan con un número determinado y limitado de recursos y saben las fechas compromiso de entrega. Con estos datos se ajusta el esfuerzo que deberá realizar cada recurso para poder cumplir con los compromisos, lo que se traduce en excesivas cargas de trabajo que por lo regular derivan en productos sin calidad pero entregados a tiempo.

La estimación del software ayuda a los administradores a determinar las fechas de entrega, ya que se basan en la información del esfuerzo requerido (proporcionado por la estimación) y en los recursos con los que cuentan.

Con una buena metodología de estimación que proporcione información sobre el esfuerzo requerido y debido a que los administradores de proyectos conocen la cantidad y calidad de recursos con los que cuenta la empresa, se pueden determinar fechas reales de entrega de productos de calidad.

La estimación del software no sólo tiene que ver con la estimación total del tiempo y el esfuerzo sino también con especificar las actividades

laborales, los niveles de habilidad, atender las restricciones y calendarizar los recursos necesarios. Estas consideraciones están influenciadas por factores tales como el modelo del proceso, la metodología y el desarrollo de herramientas a utilizar.

Además de proveer estimaciones iniciales de esfuerzo y calendarizar los proyectos de software, la estimación de software también tiene que ver con estimaciones actuales de costos y fechas a completar; recabar y analizar datos históricos; desarrollo, calibración y ajuste de modelos de estimación; y con el análisis de las tendencias en la calidad de software, la productividad del programador y la predictibilidad de proyectos.

Los factores a considerarse cuando se prepara una estimación de software incluyen las características de producto tales como: tamaño, complejidad y confiabilidad requerida; niveles de habilidad del personal, disponibilidad y motivación; la adecuación de las herramientas de software y el desarrollo del ambiente; y las restricciones en la calendarización, presupuesto y características del producto.

Una técnica efectiva de estimación de software debe capturar las relaciones y las posibilidades de negociación entre el calendario, el presupuesto, el activo de la empresa, los objetivos y las restricciones.

La estimación del costo del software (ECS) sigue siendo un punto débil en la administración de un proyecto de software.

Es responsabilidad del líder del proyecto hacer estimaciones precisas del esfuerzo y el costo ya que a partir de éstas la administración siempre decide si debe proceder o no con el proyecto.

A este respecto es muy importante presentar propuestas competitivas pues una propuesta muy alta comparada con los competidores resultaría en la pérdida del contrato y una muy baja en pérdidas para la organización.

La industria tiene la necesidad de contar con estimaciones precisas del esfuerzo y el tamaño en etapas muy tempranas de un proyecto. Sin embargo, cuando se estima el costo del software al inicio del desarrollo, la estimación podría estarse basando en requerimientos erróneos o incompletos.

El proceso de estimación del software comprende el conjunto de técnicas y procedimientos que una organización utiliza para calcular el costo y el esfuerzo requeridos.

Es necesario conocer los pasos para establecer la estimación del Costo del Ciclo de Vida del software (CCV), después dar seguimiento y refinar las estimaciones a lo largo de la vida del proyecto ya que al establecer este proceso en etapas tempranas del ciclo de vida proporcionará mayor precisión, credibilidad y una mayor claridad de los factores que influyen en los costos del desarrollo.

La estimación del software es un proceso continuo que debe ser usado a lo largo del ciclo de vida de un proyecto y consiste en el siguiente procedimiento:

- Estimar tamaño
- Estimar costo y esfuerzo
- Estimar un calendario
- Estimar los recursos de cómputo críticos
- Establecer los riesgos
- Inspeccionar/aprobar las estimaciones
- Dar seguimiento y reporteo a las estimaciones
- Medir y mejorar el proceso

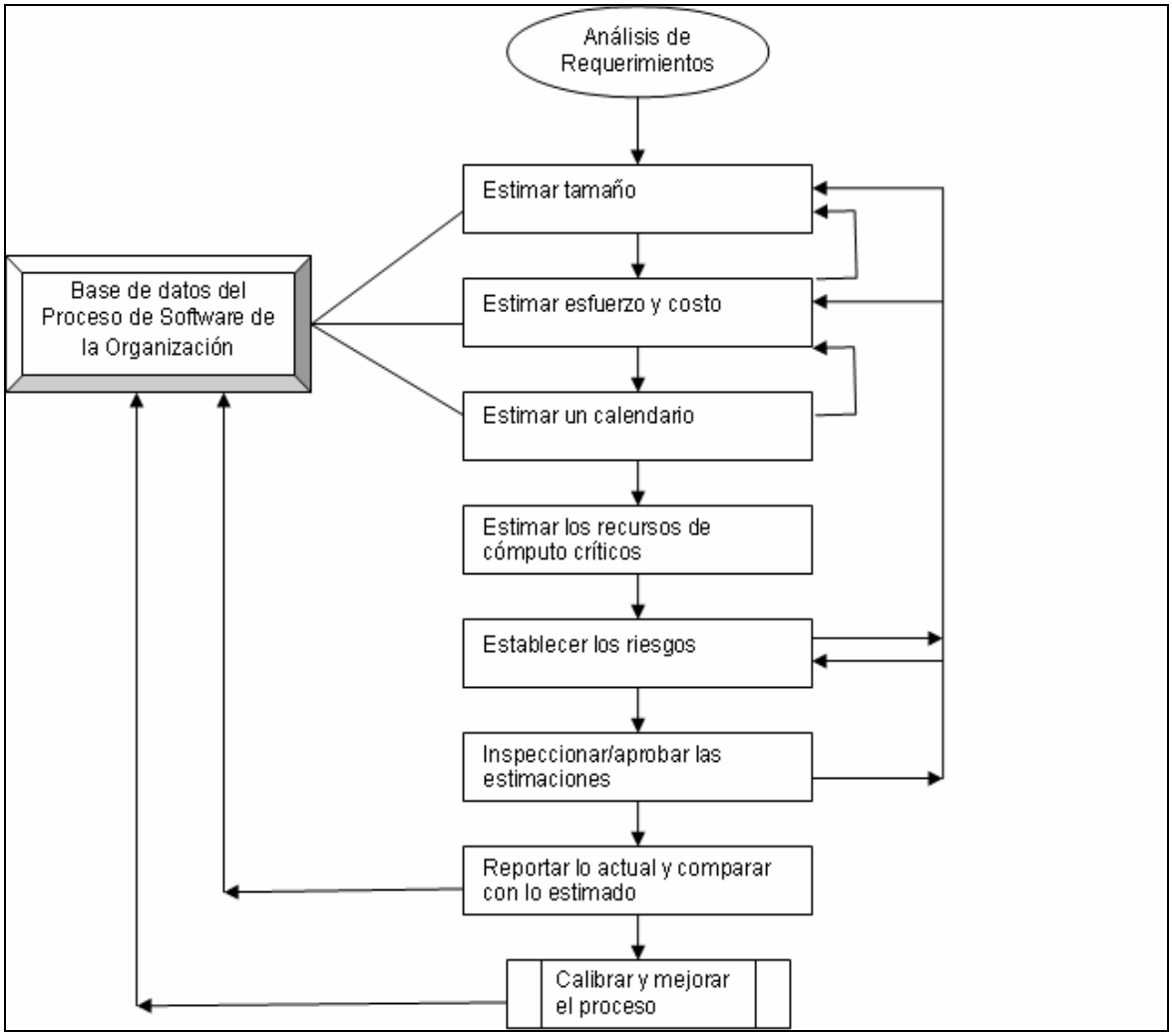


Figura 2.1 Procedimiento de estimación del Software

Las actividades del proceso para desarrollar el tamaño, el esfuerzo y el costo se muestran antes que la estimación de un calendario en la figura 2.1 porque ésta es la secuencia que siguen comúnmente los modelos de costos. Sin embargo, el desarrollo de un calendario se indica comúnmente antes de que se visualice el alcance del esfuerzo.

El establecimiento de una estructura de repartición del trabajo ayuda a dividir el esfuerzo dentro de los equipos a los que se les calendariza y da prioridad.

Como se muestra en la figura 2.2, el primer paso para estimar el costo del desarrollo de un proyecto dado es estimar el tamaño de la

aplicación a ser desarrollada. En otras palabras, qué tan grande será el esfuerzo.

Después se ingresan a la computadora coeficientes para generar el tamaño de un proyecto específico contra las curvas de costos para determinar el esfuerzo total y el calendario para dicho proyecto.

El siguiente paso es ajustar esta estimación a los factores de ambiente del proyecto, tales como: la capacidad del equipo, la complejidad del problema, el desarrollo del ambiente, etc. El resultado es una estimación del esfuerzo y tiempo, asumiendo que el proyecto se desarrollará en óptimas condiciones. En el mundo real es común para los negocios que el proyecto se desarrolle en condiciones aceleradas y no óptimas, lo cual genera un decremento en la eficiencia y un aumento en los costos.

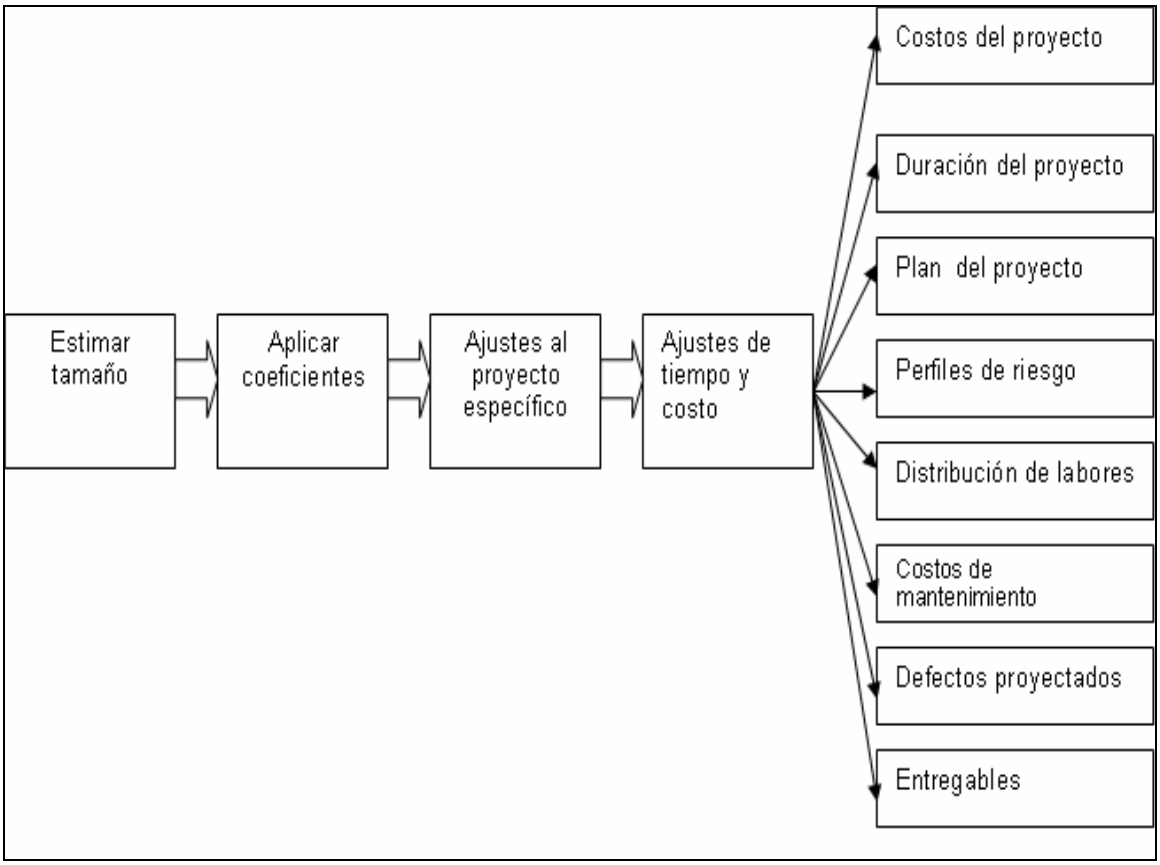


Figura 2.2 Ciclo de vida de la estimación de costo de software

Existen diversos métodos para estimar el tamaño del software, entre los cuales podemos encontrar los paramétricos y heurísticos.

La estimación **paramétrica** aproxima el tamaño del software entregado usando un pronosticador (**métrica**) que puede ser fácilmente determinado en las primeras etapas del ciclo de vida del software**.**

Líneas de Código (LOC) y Function Points (FP) son dos métricas de este tipo usadas comúnmente.

2.2.1 LINEAS DE CODIGO

LOC (Lines of Code) o SLOC (Source Lines of Code) es una métrica de software usada para medir la cantidad de código en un programa. LOC se utiliza comúnmente para estimar la cantidad de esfuerzo que se requerirá para desarrollar un programa, así como para estimar la productividad una vez que el software se ha producido. Medir el tamaño del software por número de líneas de código se practica desde la creación del software.

Existen 2 principales tipos de medición de LOC: LOC físicas y LOC lógicas.

La definición más común para LOC físicas es el conteo de líneas sin tomar en cuenta las líneas en blanco y los comentarios del código fuente del programa.

La medición de LOC lógicas consiste en contar el número de "sentencias", pero la definición de "sentencia" depende de cada lenguaje de programación. Por ejemplo, una medición de LOC lógicas en un programa en C se puede obtener del número de sentencias que terminan por punto y coma.

Es mucho más fácil crear herramientas para medir LOC físicas, debido a que la mecánica de conteo, como su nombre lo indica, se limita a contar la cantidad física de líneas sin tomar en cuenta la gramática del lenguaje de programación al que pertenece el código fuente. De esta forma una sentencia que ocupe 5 líneas, bajo esta modalidad de conteo puede ser contabilizada como 5 sentencias.

Existen muchos modelos de estimación del esfuerzo, costo y tiempo que utilizan LOC como un parámetro de entrada, incluyendo el mundialmente utilizado Constructive Cost Model (COCOMO).

VENTAJAS

1) Conteo automatizado

Debido a que LOC es una entidad física, el esfuerzo derivado del conteo manual puede ser fácilmente eliminado al automatizar el proceso a través de pequeñas utilerías que cuenten los programas. Sin embargo, una utilería desarrollada para un lenguaje específico no puede ser utilizada para otros lenguajes debido a las diferencias sintácticas y estructurales entre dichos lenguajes.

2) Es una métrica intuitiva

El conteo de LOC sirve como una métrica intuitiva para medir el tamaño del software debido al hecho de que “puede verse” y su efecto se puede visualizar. De esta manera, LOC facilita expresar el tamaño del software entre los programadores con poca experiencia.

DESVENTAJAS

1) Falta de confiabilidad

La medición de LOC sufre de problemas fundamentales, comenzando por la infortunada cuestión de tener que medir la productividad del desarrollo de un proyecto en base a los resultados de sólo una de sus fases: la de codificación, la cual usualmente representa del 30 al 35% del esfuerzo total.

2) Pobre relación entre LOC y funcionalidad

A pesar de que los experimentos han confirmado que el esfuerzo está altamente relacionado con LOC, la funcionalidad no. Esto se refiere a que la funcionalidad de un sistema no es directamente proporcional a la cantidad de líneas de código que lo compone. Por ejemplo, la cantidad

de LOC que se requerirían para desarrollar un sistema en lenguaje C variaría si el mismo sistema se programara en un lenguaje visual.

3) Dependencia de la experiencia del desarrollador

El número de líneas de código difiere de programador en programador. Esto se refiere a que desarrolladores expertos serían capaces de programar la misma funcionalidad con menos código, es decir, un programa con menos líneas de código podría exhibir mayor funcionalidad que otro programa similar. En particular LOC es una medida de productividad pobre para los individuos, partiendo del hecho que un desarrollador puede programar pocas líneas de código y ser más productivo que aquél que hace más líneas de código.

4) Popularidad de herramientas visuales

Con el incremente en el uso de herramientas y lenguajes visuales tales como Visual Basic, gran parte del trabajo del desarrollo se hace "arrastrando" componentes y con unos cuantos "clicks" al Mouse, disminuyendo considerablemente la cantidad de LOC desarrolladas por el programador. No es posible contar el código que se genera automáticamente. Esta diferencia genera grandes variaciones en la productividad y otras métricas con respecto a los diferentes lenguajes, haciendo de las LOC más irrelevantes en el contexto de herramientas visuales.

5) No aplica en desarrollo orientado a objetos

LOC no tienen ningún significado en los desarrollos orientados a objetos donde todo se refiere a objetos y clases.

6) Problemas con múltiples lenguajes

En el escenario del software de hoy en día, se requiere de más de un lenguaje de programación para el desarrollo de un sistema. Con frecuencia el número de lenguajes depende de la complejidad de los requerimientos. El seguimiento y reporteo de la productividad y rangos de defectos posee un serio problema en este caso debido a que los

defectos no pueden ser atribuidos a un lenguaje en específico después de la integración del sistema.

7) Falta de estándares de conteo.

No existe una definición estándar de lo que es un LOC, ¿cuentan los comentarios?, ¿se incluyen las declaraciones de variables?, ¿qué pasa si una sentencia se extiende varias líneas? Estas son algunas preguntas que podrían surgir. A pesar de que organizaciones como SEI y la IEEE, han publicado algunas guías en un intento por estandarizar el conteo, es difícil ponerlas en práctica, especialmente en los nuevos lenguajes de programación.

Existen tres **técnicas heurísticas** principales: top-down, bottom-up y juicio experto.

Al utilizar técnicas heurísticas no necesitan aplicarse coeficientes ni variables de ambiente.

2.2.2 BOTTOM-UP

En esta técnica primero se divide el programa en módulos; para cada módulo, se estima directamente el nivel de esfuerzo requerido en términos de horas/hombre. Esto indica el esfuerzo total estimado para la fase de codificación. Normalmente esta actividad recae sobre el equipo actual de programación.

Después se estima el esfuerzo requerido para las actividades restantes del ciclo de vida. Esta lista de actividades variará con el modelo de desarrollo que se utilice y el tipo de especificaciones del proyecto.

Finalmente, se comparan los porcentajes de esfuerzo estimados para cada actividad en el plan de trabajo contra los porcentajes estándar. Si los porcentajes estimados concuerdan con los estándares, entonces la estimación es confiable. Si se desvían considerablemente de los estándares, indicaría una cantidad de esfuerzo no razonable, muy alto o muy bajo, para una o más actividades.

Al término del análisis de cada módulo, las estimaciones individuales se suman para obtener la estimación total.

VENTAJAS

1) Se obtienen estimaciones precisas y estables gracias al nivel de detalle en la evaluación de las unidades de bajo nivel.

2) Se promueve la responsabilidad individual ya que los encargados de analizar las unidades son los responsables también del desarrollo de la unidad y por lo tanto de mantenerla dentro del costo estimado.

3) Soporta el seguimiento del proyecto porque sus estimaciones por lo regular direccionan cada actividad dentro de cada fase del ciclo de vida del desarrollo del software.

DESVENTAJAS

1) El costo de integración del sistema, el aseguramiento de la calidad del software y la administración de la configuración no son relevantes durante el proceso.

2) Este tipo de análisis consume mucho tiempo lo cual lo hace inapropiado cuando el personal disponible es limitado en número.

3) Es necesario un gran detalle en los requerimientos lo cual no siempre es posible en las primeras etapas de un proyecto. El uso de esta técnica puede no ser apropiada hasta tener completo un diseño detallado.

4) El proceso de estimación no considera el costo asociado con la integración de las unidades en componentes de más alto nivel. Dichos costos deberán ser estimados por separado.

2.2.3 TOP-DOWN

Esta técnica parte con una estimación del esfuerzo requerido del proyecto entero. Después, utiliza porcentajes estándar para cada actividad del ciclo de vida para calcular el esfuerzo necesario en cada una de esas actividades. Finalmente compara el esfuerzo estimado para cada actividad con los estándares para determinar si los números son razonables; si no, el esfuerzo total puede ser ajustado hacia arriba o hacia abajo según se necesite.

VENTAJAS

1) Para estimaciones en las primeras etapas de un proyecto cuando solamente se conocen propiedades globales.
2) Considera las actividades a nivel de sistema: integración, documentación, control de proyecto, administración de la configuración, etc.

3) Fácil y rápido de implementar ya que requiere un mínimo de detalles del proyecto.

DESVENTAJAS

1) Es menos preciso y tiende a pasar por alto componentes de alto nivel y posibles problemas técnicos.

2) Proporciona pocos detalles para justificar estimaciones y para la toma de decisiones.

2.2.4 JUICIO EXPERTO

Utiliza la experiencia y el conocimiento de expertos para estimar el costo de un proyecto de software. Es adecuado para evaluar las diferencias entre programas pasados y futuros.

VENTAJAS

1) Reuso de la experiencia de los expertos en proyectos pasados.

2) Cálculo del impacto en el proyecto de nuevas tecnologías, aplicaciones y lenguajes.

3) Útil para programas nuevos o únicos para los cuales no existen precedentes.

4) Al incorporar la experiencia de expertos en múltiples dominios se asegura que la base del conocimiento esta completa y comprendida.

DESVENTAJAS

1) Las tendencias o falta de conocimiento de los expertos pueden crear dificultades.

2) Es difícil documentar los factores tomados en cuenta por los expertos que contribuyen a la estimación.

3) Al tener múltiples expertos en la misma área se puede crear un "cuello de botella" en el momento de llegar a acuerdos sobre la estimación.

Para el caso de los modelos basados en líneas de código, en la actualidad las herramientas de desarrollo proveen la capacidad de disminuir substancialmente el esfuerzo de codificación, pues la tendencia actual ya no es codificar, sino generar código.

Hoy el esfuerzo se centra en la fase de diseño ya que la codificación se ve fuertemente asistida por herramientas automatizadas.

A lo largo de los siguientes capítulos mostraremos las características, usos y beneficios de esta técnica así como sus ventajas y desventajas para la estimación de proyectos.

CAPÍTULO 3

FUNCTION POINTS COMO MEDIDA DE SOFTWARE

3. Function Points como medida de software

La complejidad y el tamaño de los sistemas continúan en aumento, haciéndose cada vez más difíciles de entender.

Como las mejoras en las herramientas de codificación permiten a los desarrolladores producir grandes cantidades de software para alcanzar los requerimientos del usuario, es necesario usar un método para entender y comunicar el tamaño del software que se está desarrollando.

Entender el tamaño del software implica que a partir de una medición se pueda tener una idea de qué tan “grande” (en términos de funcionalidad) es el sistema a desarrollar, para poder comunicarlo después a todos los involucrados en el proyecto.

Los seres humanos resuelven sus problemas partiéndolos en piezas más pequeñas y fáciles de comprender. Problemas que al inicio pueden parecer difíciles resultan ser simples cuando se separan en componentes o clases.

Cuando los objetos a ser clasificados son el contenido de un sistema de software, se debe usar un conjunto de definiciones y reglas o esquemas y clasificaciones para poner estos objetos en una categoría apropiada.

Function Point Analysis es un método estructurado de solución de problemas que consiste en partir los sistemas en pequeños componentes para que éstos puedan ser analizados y comprendidos de una mejor forma.

3.1 Historia de los Function Points

A mediados de los 70’s IBM estaba usando más de una docena de lenguajes de programación (APL, ensamblador, COBOL, FORTRAN, PL/I, PL/S, RPG etc.). Allan Albrecht y sus colaboradores dentro de la División de Servicios de Procesamiento de Datos de IBM se daban a la tarea de desarrollar un método efectivo de estimación y medición que pudiera ser aplicado sobre cualquier lenguaje de programación sin sufrir desviaciones. La métrica de Function Points fue el resultado de esta investigación.

En octubre de 1978 IBM y dos de sus mayores asociados (SHARE y GUIDE) patrocinaron una conferencia sobre productividad en el desarrollo de software en Monterrey, California. Allan Albrecht presentó por primera vez en esta conferencia los Function Points ante la comunidad del desarrollo de software. A la vez, IBM hacía del dominio público la métrica de Function Points.

La primera publicación internacional acerca de FP fue en 1981 en el libro del autor “Programming Productivity, Issues for the Eighties”, que fue publicado por la IEEE Computer Society Press. El libro incluía la documentación completa de Albrecht sobre FP, con el permiso de IBM.

Cuando los grandes beneficios económicos de los FP se comenzaron a conocer, el uso de las métricas se expandió rápida y espontáneamente. Para 1984, el uso de los FP entre los clientes de IBM había crecido lo suficiente como para formar el núcleo de la actual International Function Point Users Group (IFPUG), que comenzó en Toronto, Canadá.

También en 1984 Albrecht e IBM entregaron la primera gran revisión de las reglas de conteo de FP. Desde entonces se han hecho revisiones periódicas de estas reglas cada dos años. La IFPUG es ahora quien se encarga de publicar estas reglas para los usuarios de EU y gran parte del mundo.

Para el tiempo en que se fundó la IFPUG, el uso de la métrica de FP se expandía rápidamente en el mundo. A principios de los 90’s, la métrica de FP se había convertido en una útil herramienta para los administradores de proyectos de software.

Los FP pueden ser cuantificados directamente de las especificaciones y requerimientos del proyecto, mucho antes de que la codificación comience. FP puede ser comprendido tanto por los clientes como por el equipo de codificación (después de una pequeña capacitación) y ser menos errático y ambiguo que la antigua métrica de líneas de código.

Sin embargo, la métrica de FP ha ido incrementando su complejidad: la versión actual de las reglas de conteo tiene al menos 100 páginas.

En 1986 más de 75 compañías estadounidenses estaban usando la métrica de FP por lo que fue necesario capacitar un mayor número de personas en el FPA; para asegurar esto, la IFPUG decidió crear un examen de certificación en FPA y un comité encargado de validar los materiales de capacitación.

Allan Albrecht, se retiró de IBM y se unió al Software Productivity Research (SPR); juntos, en 1986, desarrollaron el primer curso de certificación en FPA ante la IFPUG. Para el 2004 había quizá más de 30 compañías americanas y 50 a nivel mundial dedicadas a ofrecer capacitación en FPA.

Para el 2001 había cerca de 650 especialistas certificados en el conteo de FP en los EU. Este número se incrementa en promedio de 125 personas cada año. El número total de especialistas certificados en el mundo sobrepasa las 1000 personas y continúa creciendo, tal vez en unos 200 cada año. El gran total estimado de aplicaciones de software medidas en FP superó los 30,000 en el 2004.

3.2 Usos y Beneficios de FP

No solamente la métrica de Function Points se expandió en términos de número de usuarios sino que sus aplicaciones fueron más allá de las contempladas inicialmente.

A continuación se describen brevemente algunos de los usos de la métrica de FP:

Evaluación de Lenguajes de Programación

Allan Albrecht y sus colaboradores usaron al mismo tiempo los FP y la vieja métrica de líneas de código. Esta práctica permitió descubrir una conversión directa de LOC a FP en los inicios de 1979. Por ejemplo, se determinó que aplicaciones en COBOL requerían en promedio 105 líneas de código por FP y entre 250 y 350 en lenguaje ensamblador.

Para 1980, habían sido medidas suficientes aplicaciones con LOC y su conversión a FP y notaron que la métrica de FP podía ser usada para evaluar el nivel de los lenguajes de programación.

Antes de la aparición de la metodología de Function Point Analysis (FPA), la clasificación de los lenguajes de programación era muy ambigua. Ahora podemos establecer que la frase “lenguajes de alto nivel” puede ser aplicada a aquellos lenguajes que requieren menos de 50 líneas de código para codificar un FP. En contraste los “lenguajes de bajo nivel” requieren más de 100 líneas para codificar un FP.

Con más de 600 lenguajes de programación en uso, la habilidad para examinar y clasificar un lenguaje de programación en términos de FP ha creado una interesante rama en el análisis económico del software.

Estimación de costos del software

La primera herramienta comercial para estimar el software usando FP fue SPQR/20, liberada en marzo de 1985. Desde este punto en la historia, el uso de los FP se expandió rápidamente en el negocio de la estimación del software. En 1985 sólo 4 herramientas comerciales de estimación de software estaban disponibles en el mercado de Estados Unidos; sólo una estaba basada en FP.

Para 1995, más de 50 herramientas comerciales de estimación de software se estaban vendiendo en los EU., y por lo menos 30 de las más grandes soportaban el uso de FP; algunas de las más populares fueron: Before Your Leap (BYL), Bridge Modeler, CHECKPOINT, COCOMO II, Estimacs, GECOMO, KnowledgePlan, Price-S, ProQMS, SLIM y SPQR/20.

Aunque algunas viejas herramientas basadas en LOC siguen en uso, no han sido desarrolladas nuevas herramientas de este tipo. En el siglo 21 predominan las herramientas de estimación de software basadas en FP.

Calidad del software

La métrica de FP es la preferida para el análisis de la calidad del software ya que permite medir el volumen de defectos en los

requerimientos, especificaciones y otros aspectos no relacionados con la codificación.

Las normas actuales en Estados Unidos indican que para entregar un software de calidad éste deberá tener cerca de 5 errores o defectos por FP, lo cual se traduce en una eficiencia del 85% en la detección y corrección de errores durante el desarrollo del software.

Los FP proporcionan un panorama general de los defectos del software. Según la norma, los errores deberán distribuirse como se indica en la Tabla 3.1

Requerimientos	1.00
Especificaciones	1.25
Líneas de Código Fuente	1.75
Documentación de Usuario	0.60
Errores Secundarios	0.40
TOTAL	5.00

Tabla 3.1 Volumen Promedio de defectos por FP

Las mejores organizaciones en su clase pueden producir aplicaciones con menos de tres errores o defectos por FP y eliminar más del 95% de todos los defectos antes de la liberación al cliente.

Antes del desarrollo de los FP no se podían medir los defectos de los requerimientos y el diseño, sólo los de la codificación, usando la métrica KLOC (defectos por cada 1000 líneas de código fuente).

Debido a que los defectos en la codificación son menos de la mitad del volumen total de los defectos del software, la inhabilidad de medir los defectos no relacionados con la codificación antes de FP explica por qué la pobre calidad del software ha sido un problema significativo para los proyectos de desarrollo del software.

Benchmarking del software

Uno de los mayores usos de la métrica de FP es en los estudios de benchmarking o comparaciones de la calidad o productividad del software entre compañías, entre industrias e incluso entre países.

La métrica de FP se ha convertido en una herramienta básica para el benchmarking y es usada por una lista larga y creciente de consultorías en la materia; por ejemplo: Compass Group, David Consulting Group, Gartner Group, IBM, IFPUG y Software Productivity Research (SPR). En el 2004 al menos 15 consultorías internacionales estaban usando FP para estudios de benchmark del software[1].

Actualmente FP se utiliza en más de 30 países alrededor del mundo.

Análisis del portafolio de soluciones

Pronto la industria se dio cuenta de que la métrica de FP podía ser usada para analizar portafolios de soluciones así como para proyectos individuales. El análisis de soluciones basadas en FP se ha estado aplicando en cientos de compañías e incluso para industrias enteras como: bancos, compañías aseguradoras, fabricantes de computadoras y compañías de software.

Pequeñas compañías poseen menos de 250,000 FP en sus soluciones corporativas, pero grandes compañías a menudo exceden el 1,000,000 de FP en sus soluciones. Las 50 compañías más grandes del mundo, exceden cada una los 5,000,000 de FP en sus soluciones corporativas[2].

Análisis de Herramientas

Uno de los usos más interesantes de la métrica de FP es su aplicación en estudios de regresión múltiple para explorar el impacto de herramientas tales como: Sybase, Rational, Lotus, IBM, Microsoft, Texas Instruments y COGNOS, en la calidad del software que se produce.

[1] Fuente: www.ifpug.org
[2] Fuente: www.fortune.com/fortune/fortune500

Cuando los estudios de productividad incluyen tanto a las herramientas que se usan como a los datos de productividad basados en FP, es posible determinar el impacto de la creciente cantidad de herramientas de mantenimiento y desarrollo de software en los equipos de ingenieros que las utilizan. Por ejemplo, herramientas de diseño, pruebas, análisis de complejidad, control de calidad, ingeniería en reversa, estimación de costos, administración de proyectos etc, están siendo evaluadas vía estudios de FP benchmark.

Ahora se sabe que se requieren más de 50,000 FP para equipar completamente un grupo de ingeniería de software y más de 30,000 FP para los administradores de proyectos. Estudios más recientes demuestran que para el aseguramiento de la calidad del software (QA), grupos con más de 10,000 FP en herramientas tienen niveles de calidad mayores que grupos de QA que están poco equipados. Los administradores de proyectos con más de 25,000 FP en herramientas de administración pueden hacer mejor su trabajo de estimación y planeación que administradores con pocas herramientas[3].

La tabla 3.2 muestra el tipo de herramientas usadas por las compañías y el número aproximado de FP que contiene cada herramienta.

Herramienta	Compañías Promedio	Compañías Líderes
Planeación	1250	3000
Estimación de costos		3000
Análisis estadístico		3000
Administración de metodología	750	3000
Análisis del problema del año 2000		2000
Estimación de calidad		2000
Soporte en la Evaluación	500	2000
Medición de proyecto		1750
Análisis de portafolio		1500

[3] Fuente: www.ifpug.org

Análisis de riesgos		1500
Seguimiento de recursos	750	1500
Análisis de valor	350	1250
Reporteo de variación de costos	500	1000
Soporte de personal	500	750
Seguimiento de objetivos clave para la empresa	250	750
Soporte al presupuesto	250	750
Function Point Analysis	250	750
LOC a FP		750
FP subtotal	5350	30,250
Número de herramientas	10	18

Tabla 3.2 Tamaño en FP de Herramientas de Administración de Proyectos

Como se puede apreciar en la tabla anterior, las compañías promedio no invierten en ciertos tipos de herramientas porque consideran que no es relevante, que no da beneficio al negocio o bien porque no cuentan con los recursos para invertir en ellas.

Análisis de "hacer" contra comprar

Aplicaciones como hojas de cálculo pueden ser adquiridas por $0.25 USD por FP. Programas más especializados en el área Financiera o en Análisis de mercados pueden tener un costo de entre $10 USD y más de $300 USD por FP. Información de este tipo sirve como un nuevo medio para evaluar los costos entre hacer desarrollos nuevos o comprar herramientas existentes[4].

Los costos del desarrollo del software pueden estar de entre menos de $200 USD por FP en pequeñas aplicaciones para usuarios finales, hasta más de $5,000 USD en grandes aplicaciones para la milicia.

[4] Fuente: www.lrgl.uqam.ca

Comparaciones de Industrias Internacionales

Una vez que se vio que los FP podían ser usados para estudios de gran escala que implicaban soluciones completas, el siguiente paso lógico fue el uso de los FP para estudios a gran escala que involucraran industrias o países.

Para 1992 los niveles comparativos de productividad y calidad de varias industrias cuyos dispositivos requerían software para operar, tales como: la bancaria, seguros, telecomunicaciones, manufactura y defensa habían sido cuantificados usando la métrica de FP.

Aunque se requieren más de 1000 proyectos analizados para tener un verdadero estudio, algunos resultados preeliminares basados en muestras pequeñas revelan datos interesantes. Por ejemplo, los largos períodos vacacionales de los que gozan los trabajadores de Europa Occidental tienden a reducir la productividad comparada contra países como Japón y los EU.

Análisis de Outsourcing

La métrica de FP provee un nuevo medio para la evaluación de los contratos de outsourcing, de hecho, una gran cantidad de compañías que contratan servicios de outsourcing están incluyendo costos por FP como parte de sus contratos para diversos tópicos, entre ellos: proyectos de desarrollo muy básico, requerimientos no contemplados desde el inicio, mantenimiento y mejoras.

Además de la información de costos, muchas de estas compañías usan los FP para otros propósitos, como por ejemplo el establecimiento de tarifas adicionales como retribución a la mejora en: productividad, entregables, calidad y fechas de entrega.

El uso de los costos por FP como un método para los contratistas de software comienza a ser muy difundido en el mundo, llegando a países como Rusia, Ucrania y la República Checa.

La razón principal de la rápida adopción de los FP por los grupos de outsourcing globales es la significativa ventaja de marketing. El costo

promedio de la construcción de un FP en Europa Occidental es cercano a los $1500 USD, pero en Europa Oriental el costo promedio es menor de $350 USD[5].

Cuando se lleva a cabo un estudio de reingeniería de procesos de negocio (Business Process Reegineering - BPR), el uso de FP provee una nueva y poderosa herramienta para alinear las capacidades de software de la compañía con las unidades operativas nuevas o revisadas.

Análisis de Aplicaciones ERP

El rápido crecimiento de las aplicaciones ERP (Enterprise Resource Plannig) comercializadas por compañías tales como SAP, Oracle, Baan, J.D. Edwards y PeopleSoft han disparado la necesidad de estimar y medir la utilización de estos proyectos masivos, que a menudo son tan grandes como una solución completa para la industria. Por ejemplo SAP R3 mide más de 250,000 FP.

Los FP pueden ser usados parar estimar la implementación de ERP's (proyectos que a menudo toman varios años) y también para estimar el desarrollo de aplicaciones bajo el esquema de ERP.

Evaluación de Aplicaciones Basadas en WEB

El explosivo crecimiento de Internet y de la World Wide Web tomó a la industria por sorpresa. Miles de aplicaciones Web están siendo desarrolladas usando nuevas herramientas como JAVA, HTML, Rapid Application Development (RAD) etc.

La métrica de FP es usada para estimar, medir y evaluar la calidad de las aplicaciones Web. Hasta el 2003, había poca información para el benchmark sobre la materia porque muchas de las aplicaciones basadas en Web son tan pequeñas (menos de 500 FP) que las compañías no se preocupan por medirlas a pesar de que esta probado que se desarrolla más rápido en este tipo de plataformas (25 FP más que el promedio) y que los defectos tienden a ser detectados bastante rápido.

[5] Fuente: www.softwaremetrics.com

La rápida aparición y desaparición de compañías ".com" ha impedido el registro de información de medición de software, algunas de estas compañías no se quedan el tiempo suficiente en el mercado como para medir los volúmenes, productividad y calidad de su software.

La métrica de FP se está colocando rápidamente en estudios económicos de productividad y calidad del software por lo que podemos asegurar que FP comienza a ser la métrica dominante en el mundo del software.

3.3 Definición de Function Point

Function point es la unidad de medida ordinal para el software tal como una hora lo es para medir el tiempo, kilómetros para la distancia o grados Celsius para la temperatura. Es una medida estándar y mundial recientemente reconocida ante la organización ISO. Es una medida independiente de aspectos tecnológicos tales como el lenguaje de programación, la plataforma, etc. Refleja el punto de vista lógico del usuario. Se puede aplicar aún cuando los requerimientos no estén totalmente definidos. Se enfoca a contar el fin y no el medio.

Function Points mide objetivamente el tamaño del software y la funcionalidad a través de los requerimientos del usuario; en este sentido cabe decir que FP es objetivo, consistente y auditable, no mide a las personas directamente[6] por lo que se considera una herramienta "macro", no "micro"[7].

Las métricas del software incluyen a FP como un factor de comparación: FP conjuntamente con el tiempo produce la métrica de Productividad, en conjunto con los defectos da la métrica de Calidad, con el costo se obtienen las métricas de costo Unitario, Retorno de la Inversión y Eficiencia.

[6] Lo que se mide de las personas es su desempeño y para ello se utiliza la métrica de Productividad, la cual a su vez requiere del cálculo previo del tamaño de la aplicación en FP's. Ver Capítulo 2, métrica "Productividad".
[7] Mide el tamaño de toda la aplicación y a partir de esta medida se pueden obtener otras métricas más específicas. Ver Capítulo 2, "Categorías de métricas".

3.4 FP y los Requerimientos de Software

La estructura de la métrica de FP está íntimamente relacionada con los aspectos fundamentales que debe incluir un requerimiento de software y considera siete puntos fundamentales.

En orden cronológico, estos siete puntos fundamentales deben ser explorados como parte del proceso de levantamiento de requerimientos:

1) Las *salidas* que debe entregar la aplicación.
2) Las *entradas* que alimentarán a la aplicación.
3) Los *archivos lógicos* que deben ser mantenidos por la aplicación.
4) Las *entidades y relaciones* que habrán en los archivos lógicos de la aplicación.
5) Los tipos de *peticiones de información* que se le pueden hacer a la aplicación.
6) Las *interfases* entre la aplicación y otros sistemas.
7) Los *algoritmos clave* que deben estar presentes en la aplicación.

Existen otros puntos auxiliares que complementan a los anteriores y deberían ser contemplados durante la fase de levantamiento de los requerimientos:

1) El *calendario de trabajo* de la aplicación desde la fase de levantamiento de requerimientos hasta la entrega.
2) Los *niveles de calidad* en términos de los defectos, confiabilidad y facilidad de uso.
3) La *plataforma de hardware* sobre la cual operará la aplicación.
4) Las *plataformas de software* tales como los sistemas operativos y las bases de datos.
5) Las *políticas de seguridad* para la aplicación y sus bases de datos.
6) Los requerimientos de *desempeño* (si es que existen).
7) Los requerimientos de *capacitación* y manuales de usuario que se necesiten.
8) Los requerimientos de *instalación* para poner la aplicación en los servidores productivos.
9) Criterios de *re-uso* para la aplicación.
10) Casos de uso o las *tareas* que los usuarios están esperando poder realizar a través de la aplicación.

La importancia de un buen levantamiento de información durante la etapa de requerimientos radica en que los elementos básicos del análisis por FPA se corresponden con los 7 puntos básicos arriba mencionados. La precisión de la estimación de FP's dependerá de la información que se pueda explotar de los requerimientos.

En el siguiente capítulo, expondremos a detalle el proceso de conteo de FP's a partir de los requerimientos del usuario.

CAPÍTULO 4

ESTIMACIÓN USANDO FUNCTION POINTS ANÁLISIS

4. Estimación usando Function Points Analysis

Function Point Analysis mide el software cuantificando la funcionalidad proveída al usuario, basándose en el diseño lógico. En este caso se considerará que el usuario es alguien que entiende el sistema desde una perspectiva funcional, más que alguien que provee requerimientos o hace pruebas de aceptación.

Durante el proceso de conteo no se consideran factores tales como la plataforma tecnológica, las herramientas de desarrollo o las líneas de código generadas, simplemente se mide la funcionalidad que será entregada al usuario final.

El resultado del conteo de FP's por si mismo no refleja ninguna información del valor o costo del sistema. Después del conteo se deben considerar otros factores como las características de la aplicación (requerimientos de seguridad, desempeño, etc.) y atributos del proyecto (habilidades de los desarrolladores, lenguajes de programación a ser utilizados, metodología y tecnología a usarse, tareas a desempeñarse, etc.) para poder estimar el costo, valor o requerimientos de recursos para la adquisición, mantenimiento o desarrollo del software.

FPA se puede introducir en las primeras fases del proceso de estimación; el conteo deberá ser re-evaluado cada vez que se modifique el alcance del proyecto o que comience una nueva fase del desarrollo.

El proceso usado para determinar el número de FP's se puede resumir en siete pasos:

1. Determinar el tipo de conteo.
2. Identificar los alcances del conteo y la frontera de la aplicación.
3. Identificar todas las funciones de datos (archivos lógicos internos y archivos de interfaz externa) y su complejidad.
4. Identificar todas las funciones transaccionales (entradas externas, salidas externas, peticiones externas) y su complejidad.
5. Determinar el conteo de FP's "sin ajustar".
6. Determinar el valor del factor de ajuste.
7. Calcular el conteo de FP's ajustado.

4.1 Determinar el tipo de conteo

El conteo de FP's puede ser asociado con tres tipos de proyectos de aplicaciones de software: desarrollo, mejora y mantenimiento.

a) Conteo de FP's para proyectos de desarrollo

Este tipo de conteo se asocia con trabajos de desarrollo para aplicaciones nuevas y debe ser actualizado en las siguientes fases del proceso de desarrollo:

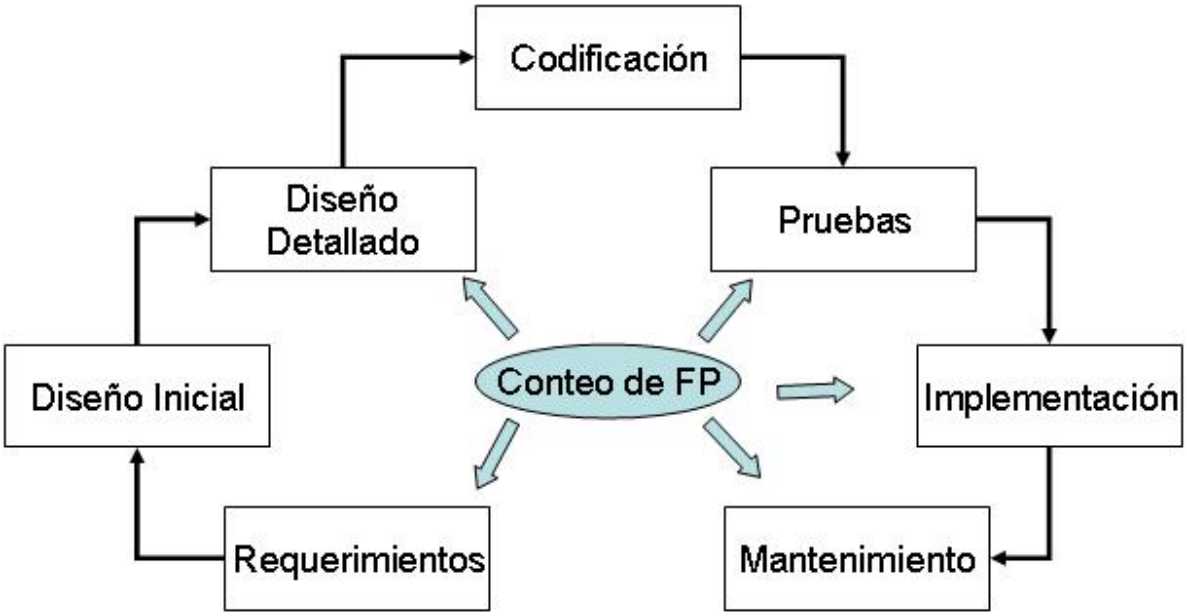


Figura 4.1 Conteo Durante el Desarrollo

A este conteo se le conoce como conteo base.

b) Conteo de FP's para proyectos de mejora

Este tipo de conteo se asocia con trabajos de mejora de sistemas existentes e incluye la funcionalidad combinada que se le proporciona al usuario a través de añadir nuevas funciones, borrar viejas funciones y cambiar funciones existentes.

Después de la mejora, el conteo de FP's de toda la aplicación debe actualizarse para reflejar la funcionalidad agregada por la mejora.

c) Conteo de FP's para proyectos de mantenimiento

Este conteo se realiza en aplicaciones que se encuentran en producción y que no fueron medidas inicialmente.

Adicionalmente, este tipo de conteo puede ayudar a las organizaciones enteras a determinar el tamaño del portafolio de aplicaciones con que cuenta, es decir, a levantar el "inventario" del tamaño de su software.

4.2 Identificar los alcances del conteo y la frontera de la aplicación.

El *alcance* del conteo lo determina su objetivo, es decir, depende de lo que se quiera medir de una aplicación; se puede medir la aplicación completa o bien uno de sus módulos. El conteo podría abarcar incluso: aplicaciones compradas, aplicaciones en outsourcing o sólo funciones dentro de la aplicación que desempeñen propósitos específicos, por ejemplo reportes.

Teniendo en cuenta que los sistemas interactúan con otros sistemas y con seres humanos, se hace necesario dibujar una frontera alrededor de cada sistema para que pueda ser medido antes de clasificar sus componentes.

La frontera separa la aplicación que se está midiendo de otras aplicaciones independientes y del dominio del usuario[1].

La IFPUG ha definido reglas específicas para identificar fronteras:

- *La frontera está basada en el punto de vista del usuario.*

  El usuario debe ser capaz de definir en su propio lenguaje el alcance de la aplicación y la funcionalidad de negocio.

[1] El dominio del usuario se refiere a las actividades que el sistema necesita que realice el usuario: capturar información, calendarización de procesos, mantener la red etc. para continuar con la operación.

- *La frontera entre aplicaciones relacionadas se determina a partir de la separación de la funcionalidad del negocio que cubre, más que en consideraciones técnicas.*

  Por ejemplo, Microsoft Office está compuesta de Word, Excel, Power Point y Access; cada una de ellas es una aplicación dentro de la suite de Microsoft Office. La frontera de Excel se delimita en base a las funciones que cumple como hoja de cálculo y se separa de la de procesador de textos que le corresponde a Word.

- *La frontera inicial de una aplicación no está influenciada por el alcance del conteo.*

  La frontera de una aplicación que está siendo mejorada permanece tal y como era al inicio excepto que la funcionalidad agregada pueda expandirla o bien que la funcionalidad borrada pueda reducirla.

  Una mejora no se puede convertir en su propia frontera de aplicación, pero sí se puede definir en el alcance del conteo, es decir, el componente mejorado forma parte de toda una aplicación por lo tanto, la frontera del proyecto de mejora debe ser la de la aplicación entera, aunque el conteo se lleve a cabo sólo sobre la mejora.

  Proyectos nuevos y de mejora comúnmente incluyen más de una aplicación. En estos casos, las fronteras de las múltiples aplicaciones se identifican dentro del alcance del conteo, pero se cuentan por separado.

  El establecimiento de la frontera para el conteo de FP se describe en siguiente figura:

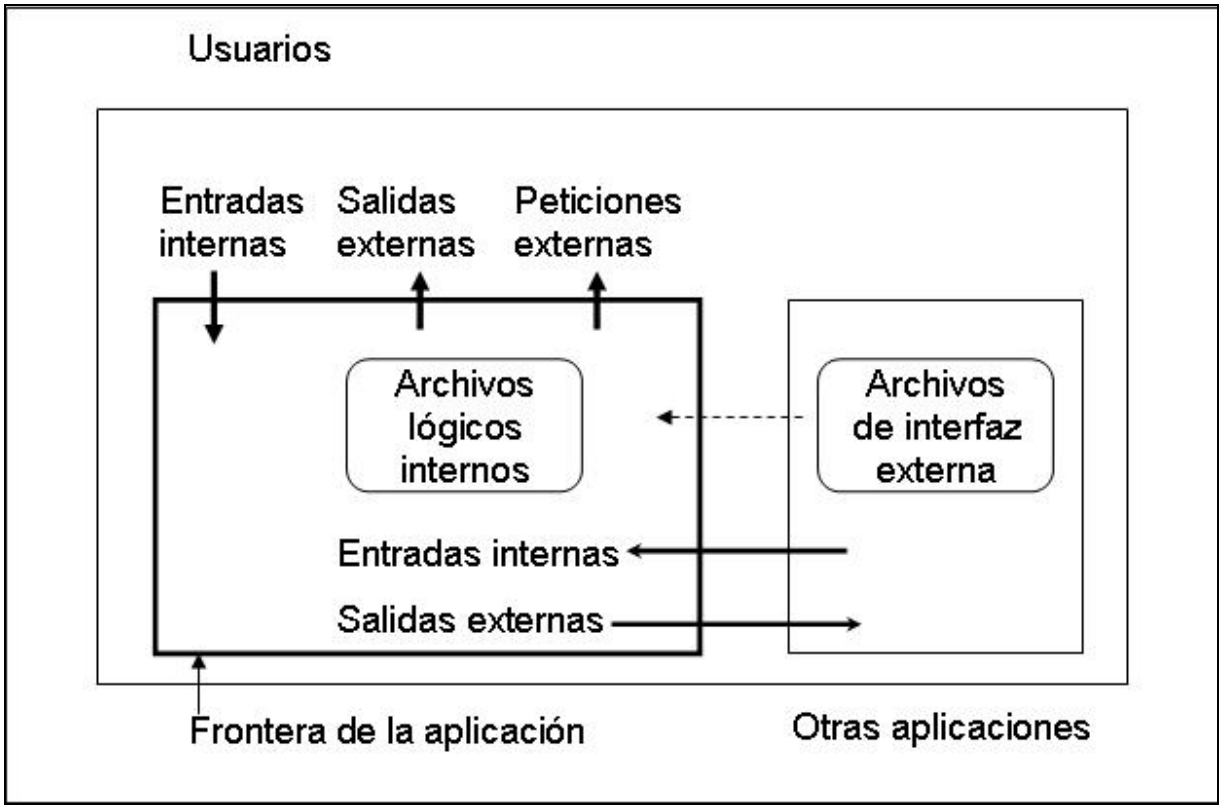


Figura 4.2 Funcionalidad reconocida en el conteo de FP

Como podemos observar en la figura 4.2, la frontera separa la aplicación que está siendo medida del dominio del usuario y de otras aplicaciones independientes.

4.3 Identificar todas las funciones de datos y su complejidad

Las funciones de datos se relacionan con datos lógicos almacenados y disponibles para ser actualizados, referenciados y extraídos.

Un dato disponible para ser actualizado es aquel que existe en un almacenamiento y puede ser modificado; referenciado es aquel al que se puede acceder de forma indirecta a través de algún atributo clave; extraído se refiere al dato que es tomado de su lugar de almacenamiento y trasladado hasta otro punto dentro o fuera de la aplicación.

Las funciones de datos se clasifican como:

- Archivos lógicos internos o ILF's (Internal Logical Files).

- Archivos de interfaz externa o EIF's (External Interface Files).

4.3.1 Archivos Lógicos Internos (ILF's)

Los ILF's son un grupo de *datos relacionados lógicamente* o *información de control*, *mantenidos* dentro de la frontera de la aplicación, que son *identificables por el usuario*.

El propósito de los ILF's es contener información mantenida a través de uno o más *procesos elementales* de la aplicación que se está midiendo.

A continuación se describen los conceptos utilizados en la definición anterior:

- *Identificable por el usuario*: se refiere a requerimientos definidos para procesos y/o grupos de datos que son reconocidos y entendidos tanto por los usuarios como por los desarrolladores. Por ejemplo, en un sistema de venta de boletos de avión, el usuario y el desarrollador deben coincidir en que la aplicación debe guardar la información referente al pasajero, itinerario, impuestos y formas de pago.

- *Relacionados lógicamente*: es que los datos se encuentren agrupados de acuerdo a una característica en común (de implementación, desempeño, etc.) que deriva de una relación física establecida en el requerimiento del usuario. Con el objetivo de mantener su existencia, un ILF no debe depender de otro ILF.

  En un diagrama de flujo los ILF's se identifican como los almacenamientos de datos. Por ejemplo, la información referente a hora de salida, número de vuelo, aeropuertos de salida/llegada y clase, pueden ser relacionados lógicamente con el itinerario de vuelo; el dato de costo de hospedaje e impuestos pagados no tienen relación con la información del itinerario por lo que no pueden formar parte de la agrupación anterior.

- *Datos*: colección de hechos o partes de información que son mantenidos dentro de la aplicación. Por ejemplo: número de cheque, monto, fecha, nombre de cliente y número de cuenta podrían ser mantenidos en un registro de forma de pago para cada boleto vendido y pagado con cheque.

- *Información de Control*: son datos usados por la aplicación que está siendo medida para determinar el flujo de un proceso elemental. Este especifica el qué, cuándo o cómo los datos son procesados. En el caso de un ILF, estos datos, reglas o parámetros son almacenados y mantenidos dentro de la aplicación. Por ejemplo, los usuarios mantienen los calendarios (hora y fecha) para establecer la ejecución de secuencias de eventos; los umbrales de temperatura se establecen en un termostato para el control del tiempo de calentamiento o enfriamiento en una habitación.

- *Mantenidos*: se refiere al hecho de que los datos son modificados a través de un proceso elemental de la aplicación. La información o datos de control pueden ser mantenidos a través de transacciones tales como agregar, cambiar, borrar, delegar, evaluar, otorgar, contener, poblar, revisar, actualizar, etc. Un ILF puede ser mantenido y contado por más de una aplicación, sin embargo, un ILF es contado una sola vez por aplicación.

- *Proceso elemental*: es la unidad de actividad más pequeña que tiene sentido para el usuario. Podemos entender como proceso elemental todas aquellas transacciones de alta, baja, cambio y consulta. Los procesos elementales cumplen con dos funciones básicas: dar mantenimiento a los datos que permanecen en reposo y mover los datos de adentro hacia fuera y viceversa de los límites de la aplicación.

Para los datos o la información de control que serán contados como ILF's se deben aplicar las siguientes **reglas** de la IFPUG:

- **El grupo de datos o información de control es lógico y debe ser reconocido por el usuario**

- **El grupo de datos es mantenido a través de procesos elementales dentro de la frontera de la aplicación que está siendo medida.**

Una vez que un grupo de datos ha sido identificado como un ILF dentro de la aplicación, este no puede ser contado también como un EIF dentro de la misma aplicación, incluso cuando éste es usado como referencia por otras transacciones, ni puede ser contado como un EIF durante un proyecto de mejora para esa aplicación.

A continuación se presentan ejemplos de grupos de datos o información de control que pueden ser considerados como ILF's:

- Datos de transacciones de la aplicación tales como registros de inventarios, registros de capacitación de empleados, registros de nómina, transacciones con tarjetas de crédito, venta de producto, llamadas de clientes o cuentas por pagar.
- Seguridad de la aplicación o claves de acceso mantenidas dentro de la aplicación.
- Datos de soporte para la operación del usuario contenidos dentro de la aplicación.
- Datos capturados mantenidos dentro de la aplicación.
- Datos de parámetros mantenidos dentro de la aplicación.
- Archivos de errores y sus descripciones mantenidas dentro de la aplicación.

Los siguientes son ejemplos de datos que con frecuencia son identificados erróneamente como ILF's. Es preciso no contar los siguientes almacenamientos como un ILF ya que no son tomados en cuenta por el FPA:

- Archivos temporales o iteraciones del mismo archivo.
- Archivos de trabajo.
- Archivos extraídos o vistas de archivos que contienen datos obtenidos de otros ILF's o EIF's antes de ser impresos o desplegados en una pantalla; éstos sin embargo sí son reconocidos como una parte del procesamiento necesario para producir un EO o EQ.
- Archivos incluidos por el tipo de tecnología usada, por ejemplo: "logs" de transacciones, archivos de parámetros,etc.

- Copias de un archivo que ya se contó, ha sido ordenado de diferente forma o mantenido en lugares separados.
- Índices alternativos, uniones, relaciones o conexiones, a menos que mantengan por separado atributos no clave.
- Datos históricos o de auditoria; deben ser contados junto con los datos de las transacciones de la aplicación.
- Archivos mantenidos por otras aplicaciones; estos se deben contar como EIF's.
- Datos de respaldo; esta capacidad es reconocida dentro de las características generales del sistema.
- Archivos que contienen transacciones incompletas, a menos que sean mantenidos por separado.

4.3.2 Archivos de interfase externa (EIF's)

Es un grupo de datos *relacionados lógicamente* o *información de control,* que son *identificables por el usuario y* referenciados por la aplicación que está siendo contada pero mantenida dentro de la frontera de una aplicación diferente.

El objetivo principal de un EIF es contener *datos* referenciados a través de uno o más *procesos elementales* de la aplicación que está siendo medida.

Un EIF contado para una aplicación debe estar dentro de un ILF en otra aplicación.

A continuación se describen los conceptos utilizados en la definición anterior:

- *Identificable por el usuario*: Igual que para un ILF.

- *Relacionados lógicamente*: Igual que para un ILF.

- *Datos*: colección de hechos o partes de información que son mantenidos dentro **otra** aplicación.

- *Información de Control*: son datos usados por la aplicación que está siendo medida para determinar el flujo de un proceso elemental. Este

especifica el qué, cuándo o cómo los datos son procesados. En el caso de un EIF, estos datos, reglas o parámetros son almacenados y mantenidos dentro de **otra** aplicación. Por ejemplo, la información de control se mantiene dentro del administrador de impresión y se lee por Power Point al momento de enviar un documento a la cola de impresión.

- *Mantenidos*: se refiere al hecho de que los datos son modificados a través de un proceso elemental de **otra** aplicación.

- *Proceso elemental*: Igual que para un ILF.

Los datos o información de control pueden ser mantenidos a través de transacciones como agregar, cambiar, borrar, delegar, evaluar, contener, poblar, revisar actualizar etc. Un EIF puede ser referenciado y contado como un EIF por más de una aplicación, sin embargo, un EIF es contado una sola vez por aplicación.

Para contar como un EIF los datos o información de control, se deben aplicar las siguientes de las **reglas** de la IFPUG:

- **El grupo de datos o información de control es lógico y reconocible por el usuario.**

- **El grupo de datos es referenciado por la aplicación que está siendo medida y externo a ella.**

- **El grupo de datos no es mantenido por la aplicación que está siendo medida.**

- **El grupo de datos es mantenido dentro de un ILF de otra aplicación.**

Una vez que un grupo de datos ha sido identificado como un EIF dentro de una aplicación, el EIF no puede ser contado otra vez dentro de la misma aplicación, incluso si éste es usado o referenciado por otras transacciones o contiene datos diferentes del mismo archivo.

A continuación se presentan ejemplos de grupos de datos o información de control que pueden ser considerados como EIF's:

- Datos de la aplicación extraídos y leídos de otras aplicaciones
- Seguridad en aplicaciones o claves de acceso mantenidas fuera de la aplicación.
- Datos de ayuda mantenidos fuera de la aplicación.
- Datos capturados, mantenidos fuera de la aplicación.
- Parámetros mantenidos fuera de la aplicación.
- Archivos de errores y sus descripciones mantenidos fuera de la aplicación.

Los siguientes son ejemplos de datos que con frecuencia son identificados erróneamente como EIF's. Es preciso no contar los siguientes almacenamientos como un EIF ya que no son tomados en cuenta por el FPA:

- Datos recibidos desde otra aplicación que mantiene uno o más ILF's dentro de la aplicación que está siendo medida; éstos son considerados como datos del proceso elemental de una función transaccional y deben ser contados como una entrada externa, EI [2].
- Datos que son mantenidos por la aplicación que está siendo medida pero accesados y usados por una aplicación diferente; éstos son contados como EIF para la aplicación accesante y como ILF para la aplicación medida.
- Datos formateados y enviados por la aplicación que está siendo contada a otras aplicaciones; estos debe ser contados como una salida / petición externa EO / EQ [3].
- Archivos temporales o iteraciones del mismo archivo.
- Archivos de trabajo.
- Archivos ya contados, ordenados de diferente forma.
- Archivos extraídos, o vistas de archivos que contienen datos obtenidos de EIF's previamente contados antes de ser impresos o mostrados, son reconocidos como parte del proceso necesario para producir la salida externa o consultas externas.
- Archivos incluidos por el tipo de tecnología usada, por ejemplo: "logs" de transacciones, archivos de parámetros,etc.

[2] Ver tema 4.4.1
[3] Ver tema 4.4.2

- Índices alternativos, uniones, relaciones o conexiones, a menos que mantengan por separado atributos no clave.
- Datos históricos, que deben ser contados todos juntos con los datos transaccionales de la aplicación.

4.3.3 Complejidad y Contribución de los ILF's y EIF's

El conteo físico de ILF's y EIF's junto con la *complejidad funcional* relativa de cada uno determina la contribución de los tipos de funciones de datos al conteo de FP's no ajustados.

A cada ILF/EIF identificado se le debe asignar una *complejidad funcional* basándose en el número de *tipos de elementos de datos* (Data Element Type, DET's) y *tipos de elementos de registros* (Register Element Type, RET's) asociados con el ILF o EIF.

- *Complejidad Funcional*: es el "peso"[4] asignado a cada función de datos. Puede ser bajo, promedio o alto de acuerdo con la matriz de complejidad[5] que considera el número de DET's y RET's que lo componen.

- *Tipos de elementos de datos* (DET's): son campos o atributos únicos reconocidos por el usuario.

- *Tipos de elementos de registros* (RET's): son subgrupos de elementos de datos (opcionales u obligatorios), reconocibles por el usuario contenidos dentro de un ILF o EIF. Los subgrupos son normalmente representados en un diagrama entidad – relación como subtipos de entidades o atributos de entidades, comúnmente nombrados relaciones padre – hijo. (El usuario tiene la opción de usar uno u ninguno de los subgrupos opcionales durante un proceso elemental que agrega o crea una instancia de los datos, el usuario debe usar al menos uno de los subgrupos mandatorios). La idea detrás de un RET en cuantificar la complejidad de las relaciones de datos dentro de un ILF/EIF.

[4] Peso, se refiere a la contribución de la función de datos en el conteo de FP's
[5] Ver tabla 4.6

Las siguientes **reglas** del IFPUG se aplican para el conteo de elementos de datos, campos o atributos para ILF's y EIF's (DET's) :

- **Se debe contar un DET para cada campo único identificable por el usuario mantenido dentro o extraído de un EIF o ILF**. Por ejemplo, los datos referentes al nombre del pasajero, número de boleto, aerolínea emisora, tarifa pagada y tipo de cambio deben ser contados como DET's ´para una entidad (ILF, EIF) llamado boleto, sin importar el número de boletos emitidos o la forma en que los datos son físicamente almacenados.

- **Cuando 2 o más aplicaciones mantienen y/o referencían diferentes DET's en el mismo ILF/EIF, se deben contar sólo los DET's que utiliza cada aplicación para medir el tamaño del ILF/EIF**. Por ejemplo, en un ILF que es actualizado por dos aplicaciones (A, B) y referenciado por una tercera (C) se contarían los DET's como se muestra en la tabla 4.1.

DET	Uso del DET	DET's contados
Número de parte	Llave primaria de A,B,C	DET para A, B, C
Nombre de la parte	Mantenido por A Referenciado por B Y C	DET para A DET para B, C
Uso semanal	Mantenido por B	DET para B
Departamento que lo usa	Mantenido por B	DET para B
Precio de compra	Mantenido por A Referenciado por B	DET para A DET para B
Nombre del proveedor	Mantenido por A Referenciado por B	DET para A DET para B
Calle del proveedor	Mantenido por A	DET para A
Ciudad del proveedor	Mantenido por A	DET para A
Estado del proveedor	Mantenido por A	DET para A
CP del proveedor	Mantenido por A	DET para A

Tabla 4.1 Ejemplo de Conteo

De acuerdo con la tabla anterior, al final del conteo debemos obtener que A cuenta con 8 DET's, B con 6 DET's y C con 2 DET's.

- **Se debe contar un DET por cada elemento de dato requerido por el usuario para establecer una relación entre dos ILF's/EIF's.** Este tipo de DET es identificado en un diagrama entidad – relación como una llave foránea.

Campos que aparecen más de una vez en un ILF/EIF debido a la tecnología o técnicas de implementación (como por ejemplo las llaves foráneas) se cuentan una sola vez en ese ILF/EIF. Sin embargo, un campo puede ser un DET para muchos ILF's/EIF's.

Campos repetidos que son idénticos en formato y existen para permitir múltiples ocurrencias del valor de un dato, se cuentan una sola vez en ese ILF/EIF. Por ejemplo, un ILF que contiene 12 campos para los montos mensuales y uno para el total anual, debería contarse con 2 DET's, uno para las cantidades mensuales y otro para el anual; se requería un DET adicional para identificar el mes.

Las siguientes **reglas** del IFPUG se aplican para el conteo de subgrupos de elementos de datos o tipos de elementos de registros para ILF's y EIF's (RET's) :

- **Se debe contar un RET para cada subgrupo opcional o mandatorio del ILF/EIF**. No se debe contar ningún RET que exista por requerimiento de la tecnología o metodología usada. Muchos Web Servers utilizan archivos de configuración, estos archivos son distribuidos junto con la aplicación y no deben ser contados.

- **Si no hay subgrupos, contar el ILF/EIF como un RET**.

4.4 Identificar las funciones transaccionales y su complejidad.

Las funciones de datos se relacionan con los datos lógicos almacenados y disponibles para actualización, referenciación y extracción.

Las funciones transaccionales -- entradas externas (EI's, External Inputs), salidas externas (EO's, External Outputs), peticiones externas (EQ's, External Inquiries) -- desempeñan los procesos de actualización y extracción que se esperarían ver en un modelo de procesos. Cada una de estas funciones transaccionales tiene su propio "peso" en el conteo de FP's no ajustados basado en su matriz única de complejidad.

4.4.1 Entradas Externas (EI's)

Una entrada externa es un *proceso elemental* en el cual se procesan *datos* o *información de control* que cruza la frontera de la aplicación de afuera hacia adentro. Estos datos pueden venir de una pantalla de captura (donde el usuario ingresa la información requerida por el proceso) o de otra aplicación.

La figura 4.3 representa un EI que actualiza dos ILF's

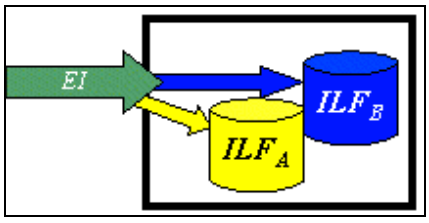


Figura 4.3 Entrada Externa (EI)

Los datos procesados *mantienen* uno o más ILF's; la información de control procesada puede o no mantener un ILF. El objetivo principal de un EI es mantener uno o más ILF's y/o alterar el comportamiento de la aplicación a través de su *lógica de procesamiento*.

A continuación se describen los conceptos utilizados en la definición anterior:

- *Proceso elemental:* es la unidad de trabajo más pequeña que tiene sentido para el usuario. Este proceso debe dejar la aplicación que está siendo contada en un estado consistente. Por ejemplo, el proceso elemental del llenado de un formulario para la captura de las prestaciones de un empleado, podría estar compuesto por tres

pantallas que, de no capturarse todos los datos obligatorios en cada una de éstas, el proceso elemental estaría incompleto.

- *Datos*: colección de hechos o partes de información procesados por las transacciones de entrada. Por ejemplo: Los datos serían los campos de información incluidos en las pantallas de captura del ejemplo anterior. Se esperaría ver el nombre del empleado, el porcentaje de contribución etc.

- *Información de Control*: son datos usados para influenciar un proceso elemental de la aplicación a ser contada. En el caso de EI's el almacenamiento de estos datos, reglas o parámetros es opcional.

- *Mantienen*: se refiere a la capacidad de modificar datos a través de un proceso elemental. Contar un EI por separado por cada actividad de mantenimiento de datos. Por ejemplo, añadir, cambiar, llenar, eliminar, revisar, actualizar, asignar, salvar y crear. Existen un sin fin de verbos que pueden ser usados para describir la actividad del mantenimiento. Sin embargo, debido a que una transacción debe ser un proceso elemental, no hay que contar las actividades de mantenimiento que no constituyan el proceso total.

- *Lógica de procesamiento:* incluye aquéllos requerimientos específicamente solicitados por un usuario para completar un proceso elemental. La lógica de procesamiento por si sola no debería ser usada para determinar EI's, EO's, EQ's. El proceso elemental de una entrada externa puede incluir, múltiples validaciones, filtros, reordenamientos, etc. El reordenamiento por ejemplo, no determina la unicidad de la transacción, es decir, la capacidad de reordenar no equivale a una transacción adicional.

  La lógica de procesamiento de un EI debe incluir:

  - Validaciones
  - Fórmulas matemáticas o cálculos
  - Conversión de valores
  - Filtrar y seleccionar datos de acuerdo a criterios específicos para comparar conjuntos de datos.
  - Análisis de condiciones para determinar cuáles aplican

- Actualización de ILF's (es obligatorio para un EI mantener uno o más ILF's)
- Referenciación de ILF's o EIF's
- Extracción de datos o información de control
- Creación de datos derivados
- Alteración del comportamiento del sistema
- Preparación y presentación de información fuera de la frontera
- Capacidad para aceptar datos o información de control que entren a la frontera de la aplicación (obligatorio para un EI)
- Reordenamiento de un conjunto de datos

Las siguientes son las **reglas** del IFPUG que se aplican por separado a las transacciones de datos o información de control de **EI's**.

Para los ***datos*** procesados que se cuentan como un **EI** se aplican las siguientes **reglas**:

- **Los datos deben ser recibidos desde fuera de la frontera de la aplicación.**

- **Los datos en al menos un ILF deben ser mantenidos a través de un proceso elemental de la aplicación.**

- **El proceso debe ser la unidad más pequeña de actividad que tenga sentido para el usuario (regla de un proceso elemental)**

- **Para el proceso identificado se debe aplicar una de las siguientes reglas:**

    1. **La lógica de procesamiento debe ser única o diferente de la desempeñada por otro EI dentro de la misma aplicación.**

    2. **El conjunto de DET's identificados es diferente del conjunto identificado para otro EI dentro de la aplicación.**

    3. **Los ILF's/EIF's referenciados son diferentes de aquellos referenciados por otro EI en la aplicación.**

Para la **información de control** procesada que se cuenta como un **EI** se aplican las siguientes **reglas**:

- **La información de control debe ser recibida desde fuera de la frontera de la aplicación.**

- **La información de control debe ser especificada por el usuario para asegurar el cumplimiento de los requerimientos de las funciones de negocio de la aplicación.**

- **El proceso debe ser la unidad más pequeña de actividad que tenga sentido para el usuario (regla de un proceso elemental).**

- **Para el proceso identificado se debe aplicar una de las siguientes reglas:**

    1. **La lógica de procesamiento debe ser única o diferente de la desempeñada por otro EI dentro de la misma aplicación**

    2. **El conjunto de DET's identificados es diferente del conjunto identificado para otro EI dentro de la aplicación**

    3. **Los ILF's/EIF's referenciados son diferentes de aquellos referenciados por otro EI en la aplicación**

A continuación se presentan **ejemplos** de grupos de datos o información de control que pueden ser considerados como EI's:

- Datos de funciones transaccionales usados para mantener un ILF tales como, una venta, un artículo perdido, una cita programada, una transferencia, una nueva contratación etc.
- Entradas que proveen información de control tales como: un sensor de sismos que reporte movimientos en la tierra.
- Mensajes de otra aplicación que requieren procesamiento
- Archivos de transacciones de otras aplicaciones que incluyen transacciones de diferentes tipos que requieren procesos separados y únicos, por ejemplo: ventas en efectivo y transacciones con tarjetas de créditos en lo cuales existen múltiples EI

- Entradas que mantienen un ILF, información de ayuda, mensajes, parámetros, etc.
- Datos ingresados por el usuario.
- Datos físicos que inician un procesamiento como por ejemplo la temperatura.

Los siguientes son ejemplos de datos que con frecuencia son identificados erróneamente como EI's:

- Datos de referencia que son leídos por la aplicación a partir de datos almacenados en otra aplicación pero que no son usados para mantener un ILF dentro de la aplicación que está siendo contada; estos se cuenta como un EIF
- Pantallas de menú que son utilizadas para la navegación o selección pero que no mantienen un ILF.
- Pantallas de acceso que facilitan al usuario la entrada a una aplicación.
- Múltiples métodos de invocar la misma lógica; por ejemplo dos acciones claves que desempeñan la misma función en la misma transacción en múltiples pantallas deben ser contados una sola vez.
- Funciones de edición de datos (copiar, pegar, arrastrar, etc.) propias del Sistema Operativo utilizadas para capturar mover información.
- Actualizar o cancelar datos de una pantalla.
- Respuestas a mensajes donde se solicita al usuario la confirmación de una transacción destructiva[6].
- Datos transferidos entre la parte on–line y batch dentro de la misma aplicación; no cruza la frontera de la aplicación.
- Datos transferidos entre cliente y servidor dentro de la misma aplicación; no cruza la frontera de la aplicación.

4.4.1.1 Complejidad y contribución de EI's

El conteo de EI's, junto con la *complejidad funcional* relativa para cada uno, determina la contribución de una entrada externa al conteo de FP's sin ajustar. A cada EI identificado se le debe asignar una complejidad

[6] Se le llama destructiva a la transacción que afecta información que no puede ser recuperada posteriormente, por ejemplo, borrar un registro.

funcional basada en el número de *DET's* y los *tipos de archivos referenciados* (*FTR's, File Types Referenced*) asociados con el EI.

- *Complejidad Funcional*: es el "peso" asignado a cada función de datos. Puede ser bajo, promedio o alto de acuerdo con la matriz de complejidad que considera el número de DET's y FTR's que lo componen.

- *Tipos de elementos de datos* (DET's): son campos o atributos únicos reconocidos por el usuario que cruzan la frontera de la aplicación. También son contadas como DET's algunas otras características específicas de las transacciones de los procesos elementales.

- *Tipos de archivos referenciados* (FTR's) o archivos referenciados: se refiere al número total de ILF's mantenidos o leídos y EIF's leídos por el EI.

Las siguientes **reglas** del IFPUG se aplican para el conteo de DET's (elementos de datos, campos, atributos) en **EI's:**

- **Contar un DET para cada campo o atributo único reconocible para el usuario, incluyendo atributos de llaves foráneas que cruzan (de entrada o salida) la frontera de la aplicación con el fin de completar el proceso elemental del EI.** Típicamente dichos campos o atributos mantienen un ILF. Por ejemplo, número de artículo, cantidad vendida y fecha se contarían cada uno como un DET en una transacción de venta, sin importar cómo está almacenada físicamente la información.

- **Un campo que no es ingresado por el usuario pero que a través de un EI está siendo calculado o extraído por la aplicación y está siendo mantenido en un ILF, no se debe contar como DET ya que no cruza la frontera.** Por ejemplo, la fecha del sistema, un valor calculado, etc.

- **Contar un DET para un campo lógico que físicamente se encuentra almacenado en múltiples campos pero que es requerido por el usuario como una sola pieza de información.**

Por ejemplo, la dirección completa que está almacenada en campos diferentes para calle, número, colonia etc.

- **Contar un DET por la capacidad de enviar un mensaje de respuesta del sistema fuera de la frontera de la aplicación para indicar que ha ocurrido un error durante el procesamiento, confirmar que el procesamiento está completo o verificar que debe continuar.** A pesar de que existen múltiples posibilidades de mensajes (error, confirmación, resultado etc.) contar sólo un DET por el proceso elemental.

- **Contar un DET por la capacidad de especificar la acción a ejecutar por el EI a pesar de que existan múltiples métodos de invocar el mismo proceso lógico.** Contar como un DET líneas de comando o funciones/acciones clave que provean la capacidad de especificar la acción; por ejemplo se contará un DET para un proceso que tenga la capacidad de generar un reporte en pantalla, exportarlo a Excel, mandarlo por correo, etc.

Las siguientes reglas del IFPUG se aplican para el conteo de FTR's en EI's:

- **Contar un FTR para cada ILF mantenido por el proceso elemental del EI.**

- **Contar un FTR para cada ILF/EIF leído durante el proceso del EI**

- **Contar sólo un FTR por cada ILF que es leído y mantenido por el EI**

4.4.2 Salidas Externas (EO's)

Una salida externa es un *proceso elemental* que genera *datos* o *información de control* que cruza la frontera de la aplicación de adentro hacia afuera.

La siguiente figura muestra un EO con dos ILF's de los cuales se obtienen datos derivados.

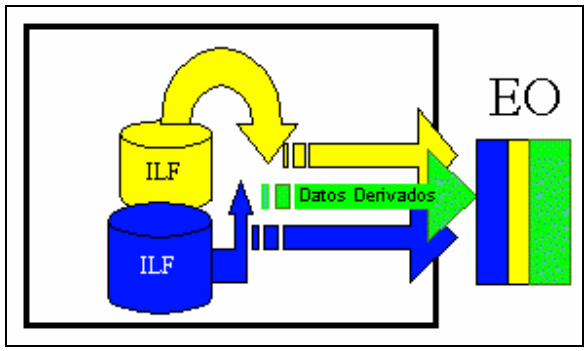


Figura 4.4 Salida Externa (EO)

El objetivo principal de un EO es presentar información a un usuario a través de una *lógica de procesamiento* además de extraer datos o información de control. La lógica de procesamiento debe contener al menos un cálculo o fórmula matemática, crear *datos derivados*, *mantener* uno o más ILF's y/o alterar el comportamiento del sistema.

Con los datos se crean reportes y archivos de salida que se envían a otras aplicaciones los cuales se crean a partir de uno o más ILF's/EIF's.

Todo EO tiene un "lado de entrada" el cual puede ser el criterio de búsqueda o parámetro que no mantiene un ILF. La información que cruza de afuera hacia adentro (lado de la entrada) son datos temporales.

A continuación se describen los conceptos utilizados en la definición anterior:

- *Proceso elemental:* igual que para un EI

- *Datos:* igual que para un EI

- *Información de control:* son datos utilizados para influenciar un proceso elemental de la aplicación que está siendo contada; en el caso de los EO's estos datos, reglas o parámetros deben ser enviados por la aplicación a un usuario u otra aplicación. Por ejemplo, señales de salida tales como una alarma un mensaje para activar o desactivar una línea de producción.

- *Datos derivados:* requieren procesamiento más que una extracción directa, conversión, y edición de información de uno o más ILF's/EIF's. Los datos derivados surgen de una transformación de datos existentes para crear datos adicionales.

- *Mantener:* se refiere a la habilidad de modificar datos a través de un proceso elemental.

- *Lógica de procesamiento:* igual que para un EI

  La lógica de procesamiento para un EO deberá incluir:

  - Validaciones
  - Fórmulas matemáticas o cálculos (un proceso elemental que procesa información y la presenta al usuario se debe considerar como un EO)
  - Conversión de valores
  - Filtrar y seleccionar datos de acuerdo a criterios específicos para comparar conjuntos de datos.
  - Análisis de condiciones para determinar cuáles aplican
  - Actualización de ILF's
  - Referenciación de ILF's o EIF's
  - Extracción de datos o información de control
  - Creación de datos derivados (un proceso elemental que procesa información y la presenta al usuario se debe considerar como un EO)
  - Alteración del comportamiento del sistema
  - Preparación y presentación de información fuera de la frontera (obligatorio para un EO)
  - Capacidad para aceptar datos o información de control que entren a la frontera de la aplicación
  - Reordenamiento de un conjunto de datos

Se deben aplicar las siguientes **reglas** del IFPUG para los datos o información de control que son procesados y contados como un **EO:**

- **Los datos o información de control deben salir de la frontera de la aplicación.**

- **La lógica del proceso elemental de un EO debe desempeñar una de las siguientes actividades:**

  1. **Contener al menos un cálculo o fórmula matemática.**
  2. **Crear datos derivados.**
  3. **Mantener al menos un ILF.**
  4. **Alterar el comportamiento de la aplicación.**

- **El proceso debe ser la unidad más pequeña de actividad que tenga sentido para el usuario (regla de un proceso elemental)**

- **El proceso debe dejar en un estado consistente a la aplicación a ser contada.**

- **Para el proceso identificado se debe aplicar una de las 3 reglas:**

  1. **La lógica de procesamiento debe ser única o diferente de la lógica desempeñada por otro EO dentro de la aplicación.**

  2. **El conjunto identificado de DET's es diferente del identificado para otro EO en la aplicación.** Cabe señalar que se refiere a diferentes DET's y no a diferentes datos en los mismos campos. Por ejemplo: un reporte contable para varios individuos que tienen diferentes datos en los mismos campos se cuenta como un sólo EO. Si se tuvieran dos reportes detallados se contarían como dos EO's debido a la lógica de procesamiento y los cálculos que requeriría cada uno.

  3. **Los ILF's/EIF's referenciados son diferentes de aquellos referenciados por otros EO's en la aplicación.**

A continuación se presentan **ejemplos** de grupos de datos o información de control que pueden ser considerados como EO's:

- Reportes que requieren el uso de cálculos o algoritmos, por ejemplo, un reporte de ventas mensual.
- Transferencia de datos, archivos y/o mensajes hacia otras aplicaciones cuando los datos son calculados o cuando ocurre una actualización como parte del proceso elemental, por ejemplo, un

archivo de transacciones enviado desde una aplicación a otra aplicación.
- Un cheque que, cuando es creado, simultáneamente actualiza el registro del cheque con el número de folio del cheque (número secuencial).
- Datos calculados que son mostrados en pantalla o guardados en archivos.
- Gráficas como diagramas de barras o de pastel que requirieron de cálculos previos para ser presentadas.
- Datos calculados enviados vía telefónica, por ejemplo, el saldo de una tarjeta de crédito.
- Cálculo de la prima de una póliza de seguro.

Los siguientes son ejemplos que con frecuencia son identificados erróneamente como EO's:

- Reportes idénticos con diferentes valores como reportes por departamento.
- Reportes o archivos que no tienen fórmulas, cálculos, datos derivados o que no mantienen un ILF dentro de la aplicación.
- Campos de totales contenidos en un reporte detallado (el reporte detallado es un EO).
- Actualización o cancelación de datos de una pantalla.
- Reordenamiento de un conjunto de datos sin alguna otra lógica de procesamiento.
- Datos almacenados que son leídos por otra aplicación (los datos no son procesados por un EO de la aplicación que se está contando).
- Archivos o pantallas de ayuda.
- Pantallas de "log-off" del sistema.
- Múltiples métodos de invocar el mismo proceso de salida.
- Mensajes de error que resultan de la edición o validación de un EI.
- Mensajes de confirmación de que los datos han sido procesados.
- Mensaje que requieren confirmación por parte del usuario.
- Datos idénticos enviados a más de una aplicación.
- Reportes a la medida que los usuarios directamente controlan a través del uso de lenguajes como SQL.
- Datos transferidos entre la parte on-line y batch de una aplicación.

- Datos transferidos entre el cliente y el servidor dentro de la misma aplicación.

4.4.2.1 Complejidad y contribución de EO's

El conteo de EO´s, junto con la *complejidad funcional* relativa para cada uno, determina la contribución de las salidas externas al conteo de FP's sin ajustar. A cada EO identificado se le debe asignar una complejidad funcional basada en el número de *DET's* y *FTR's,* asociados con el EO.

- *Complejidad Funcional*: es el "peso" asignado a cada función de datos. Puede ser bajo, promedio o alto de acuerdo con la matriz de complejidad[7] que considera el número de DET's y FTR's que lo componen.

- *Tipos de elementos de datos* (DET's): son campos o atributos únicos reconocidos por el usuario que cruzan la frontera de la aplicación.

- *Tipos de archivos referenciados* (FTR's) o archivos referenciados: se refiere al número total de ILF's mantenidos o leídos y el total de EIF's leídos por el EO.

Las siguientes **reglas** del IFPUG se aplican para el conteo de DET's (elementos de datos, campos, atributos) en **EO's:**

- **Contar un DET para cada campo o atributo único reconocible para el usuario que entra a la frontera de la aplicación y que es requerido para especificar qué, cuándo y/o cómo los datos son extraídos o generados por el proceso elemental del EO.** Estos campos son considerados como información de control o parámetros de procesamiento (del lado de la entrada).
- **Contar un DET para cada campo o atributo único reconocible para el usuario que sale de la frontera de la aplicación.** Incluso atributos de llaves foráneas o información de control (lado de la salida).

[7] Ver tabla 4.4

- **Si un DET entra y sale de la frontera, contarlo una sola vez para el proceso elemental.**

- **Contar un DET por la capacidad de enviar un mensaje de respuesta del sistema fuera de la frontera de la aplicación para indicar que ha ocurrido un error durante el procesamiento, confirmar que el procesamiento está completo o verificar si el proceso debe continuar.** A pesar de que existen múltiples posibilidades de mensajes (error, confirmación, resultado etc.) contar sólo un DET por todo el proceso elemental.

- **Contar un DET por la capacidad de especificar la acción a ejecutar por el EO a pesar de que existan múltiples métodos de invocar el mismo proceso lógico.** Por ejemplo, un botón de aceptar (OK), teclas de función (F1- ayuda), teclas de acción (Ctrl+C, copiar) o el clic del Mouse.

- **No contar como DET variables de páginas o datos generados por el sistema, incluyendo números de página, información de posicionamiento (renglón x de y), comandos de navegación (anterior, siguiente) o campos de fecha/hora.** Los DET's contados incluyen campos de fecha extraídos pero no fechas generadas por el Sistema, tales como las que se imprimen en un reporte.

- **No contar literales, incluyendo títulos de reportes, identificadores de pantalla, encabezados y nombres de campos.**

- **No contar campos que son mantenidos en un ILF durante un proceso elemental de un EO si el campo no cruza la frontera.**

- **Contar un DET por un campo lógico que está almacenado físicamente como múltiples campos pero que el usuario lo requiere como una sola pieza de información.** Por ejemplo, una fecha o nombre almacenada como tres campos pero usada como uno sólo se cuenta como un DET. Una dirección (calle, ciudad, CP) se debe contar como un DET en una etiqueta de dirección si va a ser considerado para representar un grupo de datos.

- **Contar un DET para cada tipo de etiqueta y cada tipo de equivalencia numérica en un despliegue gráfico.** Por ejemplo, una gráfica de "pastel" debe tener dos DET, uno para la categoría y uno para el porcentaje aplicable.

- **Contar un DET para la información textual que puede consistir en una palabra simple, una oración, un párrafo o muchos párrafos.**

Las siguientes **reglas** del IFPUG se aplican para el conteo de FTR's (tipos de archivos referenciados) en **EO's:**

- **Contar un FTR por cada ILF/EIF leído durante el procesamiento del EO.**

- **Contar un FTR para cada ILF mantenido por el proceso elemental de EO**

- **Contar un solo FTR por un ILF que es tanto leído como mantenido por el EO.**

4.4.3 Peticiones Externas (EQ's)

Una petición externa es un *proceso elemental* de la aplicación que consiste en la extracción de *datos* o *información de control* que cruza la frontera de la aplicación de adentro hacia afuera.

El objetivo principal de un EQ es presentar información a un usuario a partir de la extracción de datos o información de control de un ILF/EIF. La *lógica de procesamiento* no contiene formulas matemáticas o cálculos, no crea *datos derivados*, no *mantiene* ningún ILF ni altera el comportamiento de la aplicación.

Un EQ involucra las siguientes actividades mandatorias para que se considere que la transacción del EQ está terminada:

1. Una solicitud de información (generada dentro de la aplicación por un proceso o externamente a la aplicación por un proceso, persona, cosa u otra aplicación)

2. La extracción de uno o más ILF/EIF.
3. La entrega de la información al usuario.

La siguiente figura muestra un EQ que extrae información de un ILF

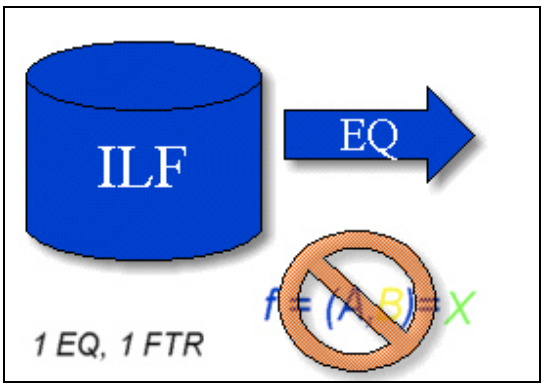


Figura 4.5 Petición Externa (EQ)

A continuación se describen los conceptos utilizados en la definición anterior:

- *Proceso elemental:* Es la unidad más pequeña de actividad que tiene significado para el usuario. El proceso debe dejar a la aplicación en un estado consistente. Por ejemplo, para desempeñar una particular extracción los usuarios han solicitado que ellos puedan ingresar cinco diferentes campos con información de control – modelo, color estilo, año de manufactura, garantía. Las posibles combinaciones de los cinco campos se logran a través de selecciones “y/o” entre ellos. Para que la transacción pueda ser considerada como un proceso elemental los registros que cumplan con los criterios de selección deben ser presentados al usuario que los solicitó.

- *Datos:* se refieren a los campos de información procesada por la petición. Un ejemplo podrían ser los datos extraídos y presentados como resultado de la solicitud descrita arriba.

- *Información de control:* son datos que influencian un proceso elemental de la aplicación a ser contada; en el caso de los EQ’s, estos datos reglas o parámetros especifican qué, cuándo o cómo se procesarán los datos. La información de control no es un proceso elemental por si sólo.

- *Datos derivados:* requieren procesamiento más que una extracción directa, la conversión, y edición de información de uno o más ILF's/EIF's. Los datos derivados surgen de una transformación de datos existentes para crear datos adicionales.

- *Mantener:* se refiere a la capacidad de modificar datos a través de un proceso elemental.

- *Lógica de procesamiento:* igual que para un EI.

Se deben aplicar las siguientes **reglas** del IFPUG para los datos o información de control que son procesados y contados como un **EQ:**

- **Los datos o información de control deben salir de la frontera de la aplicación.**

- **Los datos o información de control deben ser extraídos de uno o más ILF's/EIF's.**

- **La lógica de procesamiento no debe crear datos derivados.**

- **La lógica de procesamiento no debe contener fórmulas matemáticas o cálculos.**

- **La lógica de procesamiento no debe alterar el comportamiento de la aplicación.**

- **El proceso no debe mantener un ILF.**

- **El proceso debe ser la unidad más pequeña de actividad que tenga sentido para el usuario (regla de un proceso elemental).**

- **El proceso debe dejar en un estado consistente a la aplicación a ser contada.**
- **Para el proceso identificado se debe aplicar una de las 3 reglas:**

1. **La lógica de procesamiento debe ser única o diferente de la lógica desempeñada por otro EQ dentro de la aplicación.**

2. **El conjunto identificado de DET's es diferente del identificado para otro EQ en la aplicación.**

3. **Los ILF's/EIF's referenciados son diferentes de aquellos referenciados por otros EQ's en la aplicación.**

A continuación se presentan **ejemplos** de grupos de datos o información de control que pueden ser considerados como EQ's:

- Datos de funciones transaccionales que son extraídos y desplegados de uno o más ILF's/EIF's. Por ejemplo, datos de un empleado, datos de un pago, la descripción de un artículo
- Funciones del usuario: vistas, búsquedas, navegar, imprimir.
- Solicitudes implícitas (extracción previa a una función de cambio o eliminación de datos).
- Los reportes generados periódicamente que no contengan fórmulas, cálculos o datos derivados y que no mantengan un ILF.
- La entrega al usuario de datos, parámetros o información general del sistema mantenidos por el sistema. Por ejemplo, fecha y hora de la máquina, el estatus de la conexión de red, nombre de la máquina, dirección IP etc.
- Pantallas de ingreso al sistema.
- Extracción de datos mantenidos electrónicamente.
- Archivos enviados a otras aplicaciones que no contienen cálculos, fórmulas o datos derivados y no mantienen un ILF dentro de la aplicación que envía los datos (la transacción debe ser un EI para la aplicación que recibe.
- Extracción de correos electrónicos de una bandeja de entrada.

Los siguientes son ejemplos que con frecuencia son identificados erróneamente como EQ's:

- Múltiples métodos de invocar la misma lógica, por ejemplo, combinaciones de teclas que desempeñan la misma función sobre múltiples pantallas, se deben contar una sola vez.

- Peticiones que pueden ser lanzadas desde diferentes áreas o pantallas de una aplicación, se deben contar una sola vez
- Pantallas de menú que son usadas para navegación o selección y no extraen datos mantenidos.
- Reordenamientos de un conjunto de datos sin otra lógica de procesamiento.
- Mensajes de petición de confirmación de procesos por parte del usuario.
- Mensajes de error y/o confirmación.
- Documentación en línea del sistema.
- Datos transferidos entre la parte on-line y batch de una aplicación; nunca cruzan la frontera de la aplicación.
- Datos que pasan entre el cliente y el servidor dentro de la misma aplicación; nunca cruzan la frontera.
- Datos que no son extraídos de almacenamientos; por ejemplo, "hard code", código duro, datos que son establecidos por el programador durante el desarrollo y que no se pueden alterar.

4.4.3.1 Complejidad y contribución de EQ's

El conteo de EQs, junto con la *complejidad funcional* relativa para cada uno, determina la contribución de las peticiones externas al conteo de FP's sin ajustar. A cada EQ identificado se le debe asignar una complejidad funcional basada en el número de *DET's* y *FTR's,* asociados con el EQ.

A continuación se describen los conceptos utilizados en la definición anterior:

- *Complejidad Funcional*: es el "peso" asignado a cada función transaccional. Puede ser bajo, promedio o alto de acuerdo con la matriz de complejidad[8] que considera el número de DET's y FTR's que lo componen.

- *Tipos de elementos de datos* (DET's): son campos o atributos únicos reconocidos por el usuario que cruzan la frontera de la aplicación.

[8] Ver tabla 4.5

- *Tipos de archivos referenciados* (FTR's) o archivos referenciados: se refiere al número total de ILF's/EIF's leídos por el EQ.

Las siguientes **reglas** del IFPUG se aplican para el conteo de DET's (elementos de datos, campos, atributos) en **EQ's:**

- **Contar un DET para cada campo o atributo único reconocible para el usuario que entra a la frontera de la aplicación y que es requerido para especificar qué, cuándo y/o cómo los datos son extraídos y/o generados por el proceso elemental del EQ.** Estos campos son considerados como información de control, información de selección o parámetros de procesamiento (lado de la entrada).

- **Contar un DET para cada campo o atributo único reconocible para el usuario que sale de la frontera de la aplicación.** Incluso atributos de llaves foráneas o información de control (lado de la salida).

- **Si un DET entra y sale de la frontera, contarlo una sola vez para el proceso elemental.**

- **Contar un DET por la capacidad de enviar mensajes de respuesta del sistema fuera de la frontera de la aplicación para indicar que ha ocurrido un error durante el procesamiento, confirmar que el procesamiento está completo o verificar si el proceso debe continuar.** A pesar de que existen múltiples posibilidades de mensajes (error, confirmación, resultado etc.) contar sólo un DET por todo el proceso elemental.

- **Contar un DET por la capacidad de especificar la acción a ejecutar por el EQ a pesar de que existan múltiples métodos de invocar el mismo proceso lógico.** Por ejemplo, un botón de aceptar (OK), teclas de función (F1- ayuda), teclas de acción (Ctrl+C, copiar) o el clic del Mouse.

- **No contar como DET variables de páginas o datos generados por el sistema, incluyendo números de página, información de**

posicionamiento (renglón x de y), comandos de navegación (anterior, siguiente) o campos de fecha/hora. Los DET's contados incluyen campos de fecha extraídos pero no fechas generadas por el Sistema, tales como las que se imprimen en un reporte.

- **No contar literales, incluyendo títulos de reportes, identificadores de pantalla, encabezados y nombres de campos.**

- **Contar un DET por un campo lógico que está almacenado físicamente como múltiples campos pero que el usuario lo requiere como una sola pieza de información.** Por ejemplo, una fecha o nombre almacenado como tres campos pero usada como uno sólo se cuenta como un DET. Una dirección (calle, ciudad, CP) se debe contar como un DET en una etiqueta de dirección si va a ser considerado para representar un grupo de datos.

- **Contar un DET para cada tipo de etiqueta y cada tipo de equivalencia numérica en un despliegue gráfico.** Por ejemplo, una gráfica de "pastel" debe tener dos DET, uno para la categoría y uno para el porcentaje aplicable.

- **Contar un DET para la información textual que puede consistir en una palabra simple, una oración, un párrafo o muchos párrafos.**

Las siguientes **reglas** del IFPUG se aplican para el conteo de FTR's (tipos de archivos referenciados) en **EQ's:**

- **Contar un FTR por cada ILF/EIF leído durante el procesamiento del EQ.**

4.5 Determinar el conteo de FP's sin ajustar

El conteo de FP's sin ajustar (UFP, Unadjusted Function Point) se obtiene a partir del resultado de la identificación de las funciones transaccionales y datos (ver subtemas 4.4, 4.3 respectivamente).

Todos los componentes de una aplicación se miden con base en sus DET's, RET's o FTR's :

Tipo de Función	Componente	RET's	FTR's	DET's
Transacción	Entradas Externas (EI)		√	√
	Salidas Externas (EO)		√	√
	Peticiones Externas (EQ)		√	√
Datos	Archivos de Interfase Externa (EIF)	√		√
	Archivos Lógicos Internos (ILF)	√		√

Tabla 4.2 Componentes de una aplicación de software

El conteo ajustado de FP's (adjusted fuction point) es una combinación del UFP y las características generales del sistema (GSC, general system characteristics, tratado más adelante).

Para completar el conteo se deben conocer las reglas de FP's y la documentación de la aplicación; la calidad del conteo se puede mejorar con la participación de un experto en la aplicación.

Una vez que la frontera de la aplicación se ha establecido, el FPA se puede dividir en tres grandes secciones:

- FPA para tipos de funciones transaccionales
- FPA para tipos de funciones de datos
- FPA para GSC

Los expertos en conteo de FP's normalmente analizan primero las funciones de datos (ILF's/EIF's) por dos razones principales:

- Debemos saber cuáles ILF's y EIF's son mantenidos y/o referenciados por cada función transaccional, con el objetivo de asignar a cada uno su propio grado de complejidad.

- Si de inicio se identifican los almacenamientos para los datos, podemos validar sus designaciones originales como ILF's o como EIF's y poder proceder a identificar las funciones transaccionales.

Para que un **ILF** pueda ser contado como tal, la agrupación lógica de los datos debe ser actualizada o mantenida por al menos una entrada externa (EI) dentro de la aplicación.

Además, el ILF debe ser leído o referenciado por un EO, EQ u otro EI o esa capacidad debe ser planeada. La mayoría de las veces el ILF es leído o referenciado en la aplicación que está siendo medida, pero puede ser leído o referenciado dentro de otra aplicación.

Los ILF normalmente son representados por tipos de entidades en segunda o tercera forma normal (en un modelo entidad-relación de base de datos).

Para que un **EIF** pueda ser contado como tal, algunos de los datos de la agrupación lógica (sin importar desde dónde es mantenido) deben ser leídos o referenciados por al menos un EI, EO o EQ dentro de la aplicación que está siendo medida. Los datos son leídos o referenciados por una gran variedad de razones, por ejemplo, para su edición, para ser desplegados en una pantalla o para hacer cálculos o comparaciones.

FPA para tipos de funciones transaccionales

Los pasos para realizar el conteo de las funciones transaccionales son los que se muestran en la figura 4.6:

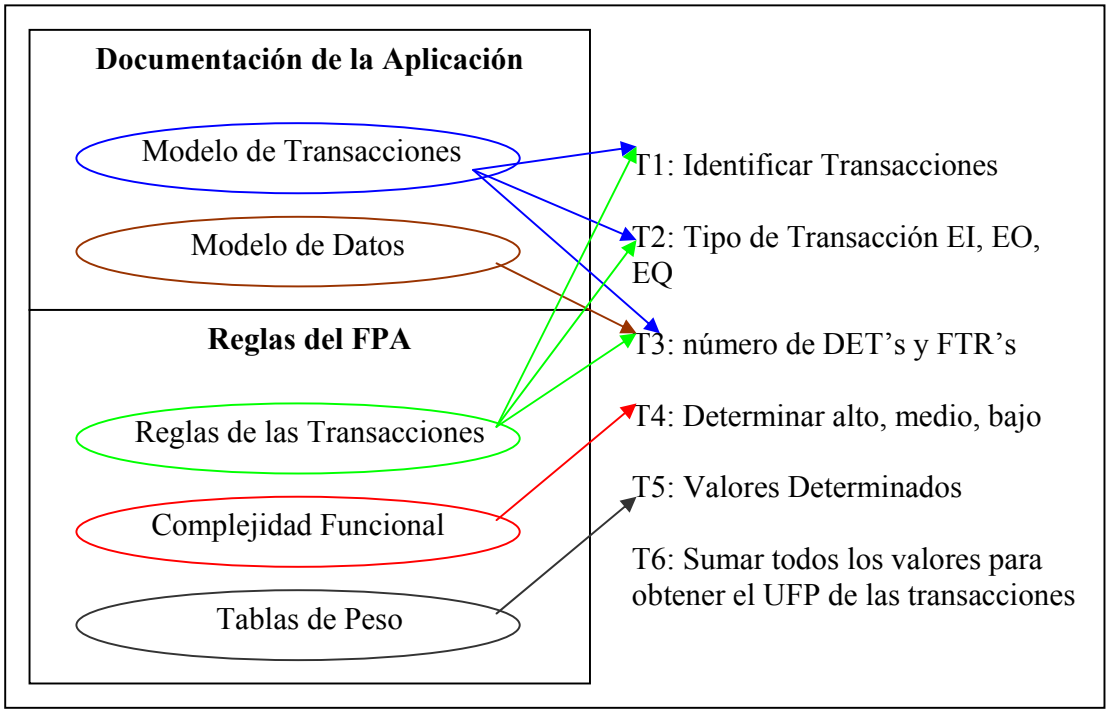


Figura 4.6 Pasos para el conteo de Funciones Transaccionales

Donde:

T1: Identificar las transacciones utilizando la documentación de la aplicación y las reglas de conteo para las transacciones.

T2: Determinar el tipo de transacción (EI, EO, EQ) utilizando la documentación de la aplicación y las reglas de conteo para las transacciones.

T3: Determinar el número de DET's y FTR's a partir de la documentación de la aplicación (modelos de datos y transacciones) y las reglas de conteo para las transacciones.

T4: Asignar un valor cualitativo (bajo, medio, alto) a cada transacción identificada con base en los DET's y FTR's que la compongan.

T5: Asignar un valor numérico distinto con base en el tipo y valor cualitativo otorgado.

T6: Obtener el conteo no ajustado de FP's (UFP) sumando los valores numéricos de las transacciones.

DET's de Funciones Transaccionales

EI – Campos de entrada de datos
Mensajes de Error
Valores Calculados
Botones

EO - Campos de datos en un reporte
Valores Calculados
Mensajes de error
Encabezados de columnas leídos desde un ILF

EQ - Del lado que recibe la petición:
Campos usados como parámetros de búsqueda
El clic del Mouse

Del lado que manda la respuesta a la petición
Campos desplegados en una pantalla

DET's para aplicaciones GUI[9]

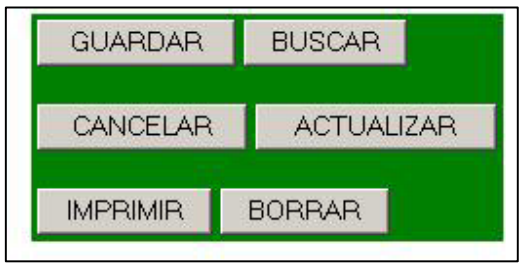
Para este tipo de aplicaciones un DET es información almacenada en un ILF o utilizada para invocar una transacción.

Debido a que al contar aplicaciones de tipo GUI la definición formal de DET de la IFPUG[10] no es de mucha ayuda (de hecho la IFPUG no detalla cómo contabilizar elementos como radio buttons, check boxes, pick list, drop down list, look ups o combo boxes), se presentan a continuación formas de contabilizar elementos propios de este tipo de aplicaciones:

[9] GUI-Graphical User Interface, Interfaz Gráfica de Usuario – cualquier aplicación donde el usuario se comunique con el sistema a través de pantallas.
[10] Ver tema 4.3.3 , definición de DET como "Tipo de Elemento de Dato"

Radio buttons: se utilizan para que, de un conjunto de opciones, el usuario sólo pueda seleccionar una, por ende, se debe contar un sólo DET por cada conjunto de radio buttons contenidos en una pantalla.

Check Box: a diferencia de los radio buttons, se puede seleccionar más de una opción a la vez por lo que cada elemento del conjunto debe ser contado como un DET.



Command buttons: Pueden especificar una petición de alta, baja, cambio o consulta. Un solo botón puede invocar uno o varios tipos de transacciones. De acuerdo a las reglas de conteo del IFPUG cada botón debe ser contado como un DET por la acción que invoque.

Iconos o imágenes: El despliegue de una imagen gráfica que le implique al sistema una transacción de consulta o carga, es decir, que no sean imágenes estáticas, se cuenta como un DET.

Sonidos: Muchas aplicaciones agregan efectos de sonido, los cuales deberán ser contados cada uno como un DET sin importar su duración.

Mensajes: Existen 3 tipos de mensajes generados en este tipo de aplicaciones: de error, de confirmación y de notificación. No se consideran procesos elementales o independientes pero son parte de un proceso elemental. La capacidad de enviar mensajes se cuenta como un DET para cada proceso elemental.

Conteo de EI's

Una buena fuente de información para determinar EI's son:

- Esquemas de Pantallas
- Documentación de Diseño
- Especificaciones Funcionales
- Requerimientos del Usuario
- Cualquier Formato de Captura
- Diagramas de Contexto
- Diagramas de Flujo de Datos

Una práctica común es que al estar leyendo la documentación de la aplicación o los requerimientos del usuario, se asocien los siguientes verbos con procesos elementales de EI's:

- Activar
- Almacenar (añadir)
- Añadir
- Cambiar
- Convertir (cambiar)
- Crear (añadir)
- Desconectar (cambiar o borrar)
- Editar (cambiar)
- Habilitar
- Modificar (cambiar)
- Mantener (añadir, cambiar o borrar)
- Reactivar (cambiar)
- Remover (borrar)
- Revisar (cambiar y borrar)
- Sobrescribir (cambiar)
- Actualizar (añadir, cambiar o borrar)
- Anular (cambiar y borrar)
- Borrar
- Cancelar
- Corregir (cambiar y borrar)
- Desasignar
- Deshabilitar
- Enviar (añadir, cambiar o borrar)
- Insertar (añadir y cambiar)
- Memorizar (añadir)
- Poner (añadir, cambiar y borrar)
- Remitir
- Remplazar (cambiar)
- Salvar (añadir, cambiar o borrar)
- Suspender (cambiar o borrar)

Para determinar la aportación de los EI's al conteo de FP's, se aplica la siguiente relación:

FTR's	DET's		
	1 – 4	5 – 15	> 15
< 2	Bajo (3)	Bajo (3)	Medio (4)
2	Bajo (3)	Medio (4)	Alto (6)
> 2	Medio (4)	Alto (6)	Alto (6)

Tabla 4.3 Matriz de complejidad para EI's

Conteo de EO's

Una buena fuente de información para determinar EO's son:

- Esquemas de Reportes
- Documentación de Diseño
- Especificaciones Funcionales
- Requerimientos de Usuario
- Descripciones de Bases de Datos
- Tamaño y Formato de Campos
- Esquemas de Reportes Gráficos

Una práctica común es que al estar leyendo la documentación de la aplicación o los requerimientos del usuario, se asocien los siguientes verbos y palabras con procesos elementales de EO's:

- Buscar
- Consultar
- Desplegar
- En línea
- Extraer
- Imprimir
- Navegar
- Obtener
- Peticiones
- Reportes
- Salida
- Seleccionar
- Vista

Para determinar la aportación de los EO's al conteo de FP's, se aplica la siguiente relación:

FTR's	DET's		
	1 – 5	6 – 19	> 19
< 2	Bajo (4)	Bajo (4)	Medio (5)
2 ó 3	Bajo (4)	Medio (5)	Alto (7)
> 3	Medio (5)	Alto (7)	Alto (7)

Tabla 4.4 Matriz de Complejidad para EO's

Conteo de EQ's

Una buena fuente de información para determinar EQ's son:

- Esquemas de Pantallas
- Documentación de Diseño
- Especificaciones Funcionales
- Esquemas de Tablas
- Requerimientos del Usuario
- Descripciones de Bases de Datos
- Listas tipo "Drop Down"
- Tamaño y Formato de Campos

Una práctica común es que al estar leyendo la documentación de la aplicación o los requerimientos del usuario, se asocien los siguientes verbos y palabras con procesos elementales de EQ's:

- Navegar
- Extraer
- Encontrar
- Listas
- Listas Desplegables
- Consultar
- Escanear
- Mostrar
- Desplegar
- Ir a traer algo de vuelta (fetch)
- Obtener
- Búsquedas
- Imprimir
- Reportes
- Seleccionar
- Vistas

Para determinar la aportación de los EQ's al conteo de FP's, se aplica la siguiente relación:

FTR's	DET's		
	1 – 5	6 – 19	> 19
< 2	Bajo (3)	Bajo (3)	Medio (4)
2 ó 3	Bajo (3)	Medio (4)	Alto (6)
> 3	Medio (4)	Alto (6)	Alto (6)

Tabla 4.5 Matriz de Complejidad para EQ's

FPA para tipos de Funciones de Datos

Los pasos para realizar el conteo de las funciones de datos son los que se muestran en la figura 4.7:

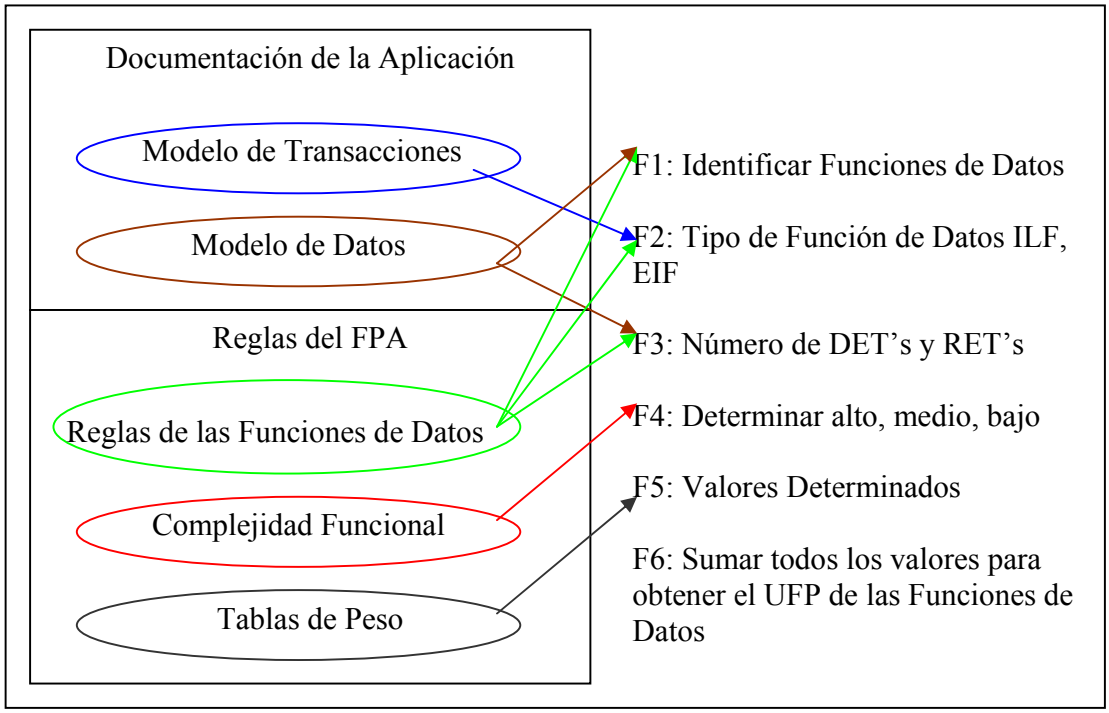


Figura 4.7 Pasos para Funciones de Datos

Donde:

F1: Identificar las funciones de datos utilizando la documentación de la aplicación y las reglas para el conteo de las funciones de datos.

F2: Determinar el tipo de función de datos (ILF, EIF) utilizando la documentación de la aplicación (modelos de datos y transacciones) y aplicando las reglas de conteo para este tipo de funciones.

F3: Determinar el número de DET's y FTR's a partir de la documentación de la aplicación (modelos de datos) y las reglas de conteo de las funciones de datos.

F4: Asignar un valor cualitativo (bajo, medio, alto) a cada función de datos identificada con base en los DET's y FTR's que las componen.

F5: Asignar un valor numérico con base en el tipo y valor cualitativo otorgado

F6: Obtener el conteo no ajustado de FP's, sumando los valores numéricos de las funciones de datos.

Conteo de ILF's

Una buena fuente de información para determinar ILF's es:

- Esquemas de Tablas
- Descripciones de Bases de Datos
- Diagramas Lógicos del Modelo de Datos
- Tamaño y Formato de Campos
- Documentación de Diseño
- Especificaciones Funcionales
- Requerimientos del Usuario

Para determinar la aportación de los ILF's al conteo de FP's, se aplica la siguiente relación:

RET's	DET's		
	1 – 19	20 – 50	> 50
1	Bajo (7)	Bajo (7)	Medio (10)
2 – 5	Bajo (7)	Medio (10)	Alto (15)
> 5	Medio (10)	Alto (15)	Alto (15)

Tabla 4.6 Matriz de Complejidad para ILF's

Conteo de EIF's

Una buena fuente de información para determinar EIF's es:

- Esquemas de Tablas
- Diagramas de Interfaces
- Descripciones de Bases de Datos
- Diagramas Lógicos del Modelo de Datos
- Tamaño y Formato de los Campos
- Documentación de Diseño
- Especificaciones Funcionales
- Requerimientos del Usuario

Para determinar la aportación de los EIF's al conteo de FP's, se aplica la siguiente relación:

RET's	DET's		
	1 – 19	20 – 50	> 50
1	Bajo (5)	Bajo (5)	Medio (7)
2 – 5	Bajo (5)	Medio (7)	Alto (10)
> 5	Medio (7)	Alto (10)	Alto (10)

Tabla 4.7 Matriz de Complejidad para EIF's

En resumen, el valor del conteo de FP's no ajustados se obtiene como sigue:

$$UFP = \sum (\text{CFD} + \text{CFT})$$

donde:

CFD = Contribución de funciones de datos
CFT= Contribución de funciones transaccionales

y

$$CFD = \sum \begin{array}{l} (\text{núm. ILFs de complejidad baja x } 7) + \\ (\text{núm. ILFs de complejidad media x } 10) + \\ (\text{núm. ILFs de complejidad alta x } 15) + \\ (\text{núm. EIFs de complejidad baja x } 5) + \\ (\text{núm. EIFs de complejidad media x } 7) + \\ (\text{núm. EIFs de complejidad alta x } 10) \end{array}$$

y

$$CFT = \sum \begin{array}{l} (\text{núm. EIs de complejidad baja x } 3) + \\ (\text{núm. EIs de complejidad media x } 4) + \\ (\text{núm. EIs de complejidad alta x } 6) + \\ (\text{núm. EOs de complejidad baja x } 4) + \\ (\text{núm. EOs de complejidad media x } 5) + \\ (\text{núm. EOs de complejidad alta x } 7) + \\ (\text{núm. EQs de complejidad alta x } 3) + \\ (\text{núm. EQs de complejidad alta x } 4) + \\ (\text{núm. EQs de complejidad alta x } 6) \end{array}$$

4.6 Determinar el Valor del Factor de Ajuste

La funcionalidad entregada por los sistemas de información incluye factores generales determinantes que no son suficientemente representados por las funciones transaccionales y de datos.

El valor del factor de ajuste (value adjustment factor, VAF) se calcula con base a la identificación de 14 características generales del sistema (general system characteristics GSC) :

1. Comunicación de Datos
2. Procesamiento de Datos Distribuidos
3. Desempeño
4. Configuración de Parámetros en los Recursos del Sistema
5. Rango de Transacciones
6. Datos Ingresados “on – line”

7. Eficiencia del usuario final
8. Actualizaciones “on-line”
9. Procesamientos Complejos
10. Reusabilidad
11. Facilidad de Instalación
12. Facilidad de Operación
13. Múltiples Sitios
14. Facilidad de Cambios

Las 14 GSC’s serán evaluadas independientemente y se les asignará un valor único llamado grado de influencia (degree of influence, DI). Cada característica tiene asociada descripciones que ayudan a determinar el DI de cada una.

Cada GSC debe ser evaluada en términos de su grado de influencia (DI) en una escala de 0 a 5:

0. No presente o no tiene influencia
1. Influencia Incidental
2. Influencia Moderada
3. Influencia Promedio
4. Influencia Significativa
5. Fuerte influencia de inicio a fin

A continuación se describen cada una de las GSC’s:

4.6.1 Comunicación de Datos

Describe el grado en el cual la aplicación se comunica directamente con el procesador.

La comunicación de datos se evalúa como sigue:

0. La aplicación está compuesta únicamente de procesos batch o es “stand – alone”
1. La aplicación es batch pero tiene entrada de datos remotamente o impresiones remotas
2. La aplicación es batch pero tiene entrada de datos remotamente e impresiones remotas

3. La aplicación recaba datos “on line” en el “front end” para enviarlos a un proceso batch o sistema de consulta
4. La aplicación es más que un “front end”, pero sólo soporta un tipo de protocolo de comunicación
5. La aplicación es más que un “front end” pero soporta más de un tipo de protocolo de información.

Por lo anterior, una aplicación con sólo procesos batch y sin interacción sería evaluada con 0. La mayoría de las aplicaciones, tanto “stand alone” como batch, tienen entrada de datos remotos así como capacidad de impresión. Las aplicaciones que tienen pantallas de captura de información pero que actualizan ILF a través de un proceso batch, deben ser evaluados con 3. Si la actualización ocurre con interacción se debe evaluar con 4.

Para evaluar con 5 deben existir múltiples tipos de protocolos de comunicación. Normalmente las aplicaciones batch reciben una evaluación de 0 a 3, aplicaciones “on line” de 3 a 4 y las de tiempo real, telecomunicaciones o sistemas de control de procesos reciben una evaluación de 4 ó 5.

4.6.2 Procesamiento de Datos Distribuidos

Describe el grado en que la aplicación distribuye datos entre sus componentes.

El procesamiento de datos distribuidos se evalúa como sigue:

0. La aplicación no ayuda a la transferencia de datos o funciones de procesamiento entre componentes del sistema.
1. La aplicación prepara los datos para el procesamiento del usuario en otro componente del sistema (por ejemplo una hoja de cálculo o un manejador DBMS que transforma las peticiones de los ambientes gráficos al lenguaje de la base de datos a la que se este conectando).
2. Los datos son preparados para ser transferidos, se transfieren y procesan en otro componente del sistema (no para procesamiento de usuarios finales)

3. La transferencia de datos y procesamiento distribuidos son “on line” y en una sola dirección
4. La transferencia de datos y procesamiento distribuidos son “on line” y en ambas direcciones
5. Las funciones de procesamiento son procesadas dinámicamente en el componente más apropiado del sistema.

Sólo aplicaciones distribuidas o de tiempo real son evaluadas dentro de esta categoría. La mayoría de las aplicaciones se evalúan con 0. Aplicaciones distribuidas antiguas pueden evaluarse con 1 ó 2. Aplicaciones cliente servidor o WEB se evalúan de 2 a 4 y de tiempo real, telecomunicaciones o sistemas de control de procesos deberían ser evaluadas de 0 a 5. Para ser evaluadas con 5, debe haber múltiples servidores o procesadores cada uno de los cuales debería ser seleccionado dinámicamente de acuerdo con su disponibilidad en tiempo real.

4.6.3 Desempeño

Describe el grado en el cual el tiempo de respuesta y las consideraciones de desempeño influencian el desarrollo de la aplicación.

Los objetivos de desempeño de la aplicación, establecidos o aprobados por el usuario influencian el diseño, desarrollo, instalación y soporte de la aplicación.

El desempeño se evalúa como sigue:

0. No fueron establecidos por el usuario requerimientos especiales de desempeño
1. Se establecieron y revisaron requerimientos de desempeño y diseño pero no se requirieron acciones especiales.
2. El tiempo de respuesta es crítico durante las horas pico, no se requirió diseño especial para la utilización del CPU. El “dead line” del procesamiento es para el siguiente día hábil.
3. El tiempo de respuesta es crítico durante todo el tiempo hábil. No se requirió diseño especial para la utilización del CPU.

Los requerimientos de “dead line” del procesamiento y las interfases del sistema son restrictivas.

4. Además, los requerimientos de desempeño establecidos por el usuario son lo suficientemente estrictos como para requerir tareas de análisis del desempeño en la fase de diseño.
5. Además, las herramientas de análisis del desempeño fueron utilizadas durante las fases de diseño, desarrollo y/o implementación para cubrir los requerimientos establecidos por el usuario.

Esta característica es muy similar al rango de transacciones, ambos requieren considerar el desempeño durante las fases de diseño, desarrollo e instalación. El tiempo de respuesta se refiere normalmente a procesos interactivos.

Normalmente aplicaciones batch reciben una evaluación de 0 a 4; aplicaciones “on line” de 0 a 4 y de tiempo real, telecomunicaciones o sistemas de control de procesos de 0 a 5.

4.6.4 Configuración de Parámetros en los Recursos del sistema.

Describe el grado en el cual las restricciones de los recursos de la computadora influencian el desempeño de la aplicación, es decir, si se requiere que la aplicación se ejecute en servidores con grandes cargas de procesamiento.

La configuración de parámetros en los recursos del sistema se evalúa como sigue:

0. No se incluyen restricciones operacionales explícitas o implícitas
1. Existen restricciones operacionales pero son menos restrictivas que una aplicación típica. No se necesita un esfuerzo especial para cumplir las restricciones
2. Se incluyen algunas consideraciones de seguridad o tiempos
3. Se incluyen requerimientos específicos del procesador para una pieza específica de la aplicación
4. Las restricciones de operación establecidas requieren consideraciones especiales en la aplicación, en el procesador central o en un procesador dedicado

5. Además existen consideraciones especiales en los componentes distribuidos del sistema.

La mayoría de las aplicaciones se evalúan con 2; para obtener de 3 a 5 la aplicación debe ser cliente servidor, de tiempo real, telecomunicaciones o sistemas de control de procesos. Aún así se necesitaría un procesador dedicado o múltiples procesadores, despachando las mismas transacciones y buscando el mejor medio.

4.6.5 Rango de Transacciones

Describe el grado en el cual el rango de transacciones del negocio influencia el desarrollo de la aplicación.

El rango de transacciones se evalúa como sigue:

0. No se anticipa ningún período “pico” de transacciones
1. Se anticipa un período pico de transacciones (quincenal, mensual, por temporada, anual)
2. Se anticipa un período pico semanal de transacciones
3. Se anticipa un período pico de transacciones diario
4. El usuario estableció en los requerimientos un rango alto de transacciones o el acuerdo del nivel de servicio es lo suficientemente alto para requerir tareas del análisis de desempeño en la fase de análisis
5. El usuario estableció en los requerimientos un rango alto de transacciones o el acuerdo del nivel de servicio es lo suficientemente alto para requerir tareas de análisis de desempeño, además se requiere el uso de herramientas de análisis del desempeño en las fases de diseño, desarrollo y/o instalación

Una evaluación de 4 requiere tareas de análisis de desempeño durante la fase de diseño. Una evaluación de 5 requiere del uso de herramientas de análisis de desempeño. Por lo regular aplicaciones batch reciben una evaluación de 0 a 3, aplicaciones “on line” de 0 a 4 y de tiempo real, telecomunicaciones o sistemas de control de procesos de 0 a 5.

4.6.6 Datos ingresados “on line”

Describen el grado en el cual los datos son ingresados a través de transacciones interactivas.

Los datos ingresados “on line”, se evalúan como sigue:

0. Todas las transacciones se procesan en modo batch
1. Del 1 al 7% de las transacciones son datos ingresados de forma interactiva
2. Del 8 al 15% de las transacciones son datos ingresados de forma interactiva
3. Del 16 al 23% de las transacciones son datos ingresados de forma interactiva
4. Del 24 al 30% de las transacciones son datos ingresados de forma interactiva
5. Más del 30% de las transacciones son datos ingresados de forma interactiva.

Las transacciones se refieren a los EI’s, EO’s y EQ’s.

Por lo regular, las aplicaciones batch reciben una evaluación de 0 a 1 y las online, de tiempo real, telecomunicaciones o sistemas de control de procesos se evalúan con 5.

4.6.7 Eficiencia del usuario final

Describen el grado de consideración de factores humanos y la facilidad de uso de la aplicación para el usuario.

Las funciones “on line” requieren un diseño orientado a lograr la eficiencia del usuario final; dicho diseño incluye:

- Ayudas de navegación (por ejemplo: teclas de función y menús generados dinámicamente)
- Menús
- Documentación y ayuda “on line”
- Movimiento automatizado del cursor

- Desplazamientos en pantalla
- Impresión remota (vía transacciones “on line”)
- Teclas de función pre asignadas
- Tareas batch enviadas desde transacciones “on line”
- Selección con el cursor de datos de la pantalla
- Alto uso de videos, resaltar texto, colores, subrayados y otros indicadores.
- Resguardo de documentación de transacciones “on line” del usuario
- Intefases con el Mouse
- Ventanas flotantes
- La menor cantidad de pantallas posible para terminar una función de negocio.
- Soporte multilenguaje.

La eficiencia del usuario final se evalúa como sigue:

0. No incluye ninguno de los puntos anteriores.
1. Uno o tres de los puntos anteriores.
2. De cuatro a cinco.
3. Seis o más, pero no existen requerimientos específicos del usuario relacionados con la eficiencia.
4. Seis o más de los puntos anteriores y los requerimientos establecidos por el usuario final son lo suficientemente fuertes como para requerir tareas de diseño para factores humanos (por ejemplo: uso de plantillas, aumentar la cantidad de valores por default)
5. Seis o más de los puntos anteriores y los requerimientos establecidos por el usuario final son lo suficientemente fuertes como para requerir el uso de herramientas y procesos especiales con el objetivo de mostrar que los objetivos han sido alcanzados (por ejemplo: herramientas de medición del “stress”)

Las aplicaciones batch que no tienen interacción se evalúan con 0. Pantallas de captura de información que no tengan plantillas y/o defaults dentro de la aplicación, deben ser evaluadas con 3; de tenerlas, se evalúan con 4. Para evaluar con 5 deben existir laboratorios para probar la

usabilidad de la aplicación más que la funcionalidad. Sistemas de tiempo real, telecomunicaciones o sistemas de control de procesos, no se evalúan en esta característica.

4.6.8 Actualizaciones "on line"

Describen el grado en el cual los ILF's se actualizan de manera "on-line".

Las actualizaciones "on line" se evalúan como sigue:

0. Ninguna
1. Se incluye la actualización "on line" de uno a tres ILF's. El volumen de actualizaciones es bajo y la recuperación es sencilla.
2. Se incluye la actualización "on line" de cuatro o más ILF's. El volumen de actualizaciones es bajo y la recuperación es sencilla.
3. Se incluye la actualización "on line" de la mayoría de los ILF's.
4. Además, la protección contra la pérdida de información es esencial y ha sido especialmente diseñada y programada en el sistema.
5. Además, volúmenes altos traen costos considerables dentro del proceso de recuperación. Intervienen procedimientos de recuperación altamente automatizados, con un mínimo de operación.

Aplicaciones batch sin interacción para actualizar ILF's se evalúan de 0 a 2. La mayoría de las aplicaciones "on line" actualizan ILF's y se evalúan con 3 o más. Si ha sido programada la protección de pérdida de información en el sistema (no sólo a través de backups), se evalúa con 4. Para evaluar con 5, debe existir una capacidad de recuperación altamente automatizada construida dentro de la aplicación. Sistemas de tiempo real, telecomunicaciones o sistemas de control de procesos reciben evaluación de 4 ó 5.

4.6.9 Complejidad de procesamiento

Describe el grado en el cual el procesamiento lógico influencia el desarrollo de la aplicación, tomando en cuenta que los siguientes 5 componentes estén presentes:

1. Control sensitivo (ejemplo: procesamiento especial de auditoría) y/o procesamiento de seguridad especifico para la aplicación.
2. Procesamiento lógico extensivo.
3. Procesamiento matemático extensivo
4. Procesamientos con gran cantidad de excepciones, resultado de transacciones incompletas que deben ser reprocesadas (por ejemplo: transacciones en cajeros automáticos incompletas, causadas por la interrupción de las comunicaciones, valores de datos faltantes o validaciones fallidas)
5. Procesamiento complejo para manejar múltiples posibilidades de entrada-salida (por ejemplo: multimedia o independencia de dispositivos)

La complejidad de procesamiento se evalúa como sigue:

0. Ninguna de las anteriores
1. Una de las anteriores
2. Dos de las anteriores
3. Tres de las anteriores
4. Cuatro de las anteriores
5. Cinco de las anteriores

Para evaluar esta característica se deben analizar los siguientes aspectos: ¿la aplicación proporciona una seguridad tal como para que ciertos individuos vean o ingresen datos que otros no pueden?, ¿existe una cantidad significativa de procesamiento lógico (if/then/else)?, ¿existe procesamiento matemático (más que sumas o restas, por ejemplo matemáticas simples)?, ¿existe edición o validación compleja?, ¿se incluyen recursos multimedia en la aplicación (por ejemplo voz)?

4.6.10 Reusabilidad

Describe el grado en el cual el código de la aplicación ha sido específicamente diseñado, desarrollado y soportado para ser utilizado en otra aplicación.

La reusabilidad se evalúa como sigue:

0. No hay código reutilizable
1. Hay código reutilizable dentro de la aplicación
2. Menos del 10% de la aplicación consideró más de una necesidad del usuario.
3. 10% o más de la aplicación consideró más de una necesidad del usuario.
4. La aplicación fue específicamente empaquetada y/o documentada para facilitar su re-uso y la aplicación es adaptada a nivel de código fuente.
5. La aplicación fue específicamente empaquetada y/o documentada para facilitar su re-uso y la aplicación es adaptada para el uso a través del mantenimiento de parámetros por el usuario.

El conteo de function points evalúa con 1 a aquellas aplicaciones que re-utilizan código. El software re-utilizable y estandarizado proporciona mayor funcionalidad al usuario pues es mayor la confiabilidad y la consistencia. Evaluaciones de 2 a 5 son asignadas en base a la funcionalidad resultante y el esfuerzo extra dedicado al desarrollo, documentación y la prueba de código que se espera sea utilizado en otras aplicaciones.

4.6.11 Facilidad de instalación

Describe el grado en el cual la conversión a partir de ambientes previos influenció el desarrollo de la aplicación y además se utilizó y probó durante la fase de pruebas del sistema un plan de instalación y/o herramientas de conversión.

La facilidad de instalación se evalúa como sigue:

0. El usuario no estableció consideraciones especiales y no se requiere configuración especial para la instalación.
1. El usuario no estableció consideraciones especiales pero sí se requiere una configuración especial para la instalación.
2. El usuario estableció requerimientos de conversión e instalación y se proporcionaron guías de instalación y conversión que fueron probadas. El impacto de la conversión en el proyecto no se considera como importante.

3. El usuario estableció requerimientos de conversión e instalación y se proporcionaron guías de instalación y conversión que fueron probadas. El impacto de la conversión en el proyecto se considera importante.
4. Además de la número 2, se proporcionó y fue probada la conversión automatizada y las herramientas de instalación.
5. Además de la número 3, se proporcionó y fue probada la conversión automatizada y las herramientas de instalación.

Los desarrolladores son requeridos con frecuencia para dedicar un gran esfuerzo en convertir datos pre-existentes en nuevos archivos de datos, llenar los archivos con los datos actuales o para desarrollar software de instalación. Es preciso considerar los requerimientos de dificultad o facilidad de conversión e instalación y asignarles una evaluación en relación con su importancia.

4.6.12 Facilidad de operación

Describe el grado en el cual la aplicación atiende a aspectos operacionales, tales como los procesos de arranque, respaldo y recuperación. La aplicación minimiza la necesidad de actividades manuales (por ejemplo: manejo de papel, grabación en cintas e intervención manual local).

La facilidad de operación se evalúa como sigue:

0. El usuario no estableció consideraciones operacionales especiales, sólo los procedimientos normales de respaldo.

1 a 4. Seleccionar cuál de los siguientes elementos se presentan en la aplicación. Cada elemento tiene el valor de 1 punto, si no se indica lo contrario:

- Se proporcionan procesos efectivos de arranque, respaldo y recuperación, pero se requiere la intervención de un operador.
- Se proporcionan procesos efectivos de arranque, respaldo y recuperación, pero no se requiere la intervención de un operador (contar como 2 elementos)

- La aplicación minimiza la necesidad del uso de cintas.
- La aplicación minimiza la necesidad del manejo de papel.

5 La aplicación es diseñada para operarse sin atención, es decir, que no se requiere la intervención de un operador para el funcionamiento del sistema sino sólo para encender o apagar la aplicación, aunque conserva la característica de la recuperación automática de errores.

A menos que se esté contando un sistema muy viejo, se deberá evaluar con 1 la falta de uso de cintas y papel; evaluar con 3 si se requiere la intervención de un operador para el arranque, respaldo y recuperación; se asigna 4 cuando no se requiere la intervención de un operador y 5 si la aplicación corre y recupera información por sí sola y automáticamente. Las aplicaciones on-line se evalúan con 3 y una evaluación mayor se aplica a líneas de producción, telecomunicación o sistemas de tiempo real, los cuales operan sin intervención de interfases humanas.

4.6.13 Múltiples sitios

Describe el grado en el cual la aplicación ha sido desarrollada para múltiples ubicaciones y organizaciones del usuario. La aplicación ha sido diseñada, desarrollada y soportada para ser instalada en múltiples sitios para múltiples organizaciones.

Esta característica se mide como sigue:

0. Los requerimientos no precisan la necesidad de más de un usuario o sitio de instalación.
1. En el diseño de la aplicación se consideró la necesidad de múltiples sitios y poder operar sólo bajo ambientes de hardware y software *idénticos*.
2. En el diseño de la aplicación se consideró la necesidad de múltiples sitios y poder operar bajo ambientes de hardware y software *similares*.
3. En el diseño de la aplicación se consideró la necesidad de múltiples sitios y poder operar bajo ambientes de hardware y software *diferentes*.

4. Se proporcionó y probó la documentación y el plan para soportar a la aplicación en múltiples sitios y la aplicación es como la descrita en los puntos 1 ó 2.
5. Se proporcionó y probó la documentación y el plan para soportar a la aplicación en múltiples sitios y la aplicación es como la descrita en el punto 3.

Esta característica incluye la funcionalidad del usuario y el esfuerzo requerido para entregar una aplicación que incluirá software y/o hardware en múltiples sitios; este DI podría reflejar sólo los dispositivos de entrada tales como terminales o PC's. Preguntas para evaluar esta característica serían: ¿Son idénticos el software y/o hardware?, ¿son similares (por ejemplo: Windows 95 y NT)? o ¿son diferentes (por ejemplo: Windows, Mac y UNIX)? ¿Se proporcionaron y probaron planes de soporte y su documentación?

4.6.14 Facilidad de cambio

Describe el grado en el cual la aplicación ha sido desarrollada para ser modificada fácilmente en su lógica de procesamiento o estructura de datos.

A su vez, sus características son las siguientes:

- Se proporcionan consultas flexibles y "reporteadores".
- Los datos de control del negocio se agrupan en tablas mantenidas por el usuario.

La facilidad de cambio se evalúa como sigue:

0. No existe un requerimiento especial del usuario para diseñar la aplicación que pueda minimizar o facilitar el cambio.

1 a 5. Seleccionar cuál de los siguientes elementos se presentan en la aplicación:

- Se proporcionan consultas flexibles y "reporteadores" que pueden manejar peticiones simples –por ejemplo, se aplicó la lógica "y/o" a sólo un ILF (se cuenta como un sólo elemento).

- Se proporcionan consultas flexibles y "reporteadores" que pueden manejar peticiones de complejidad promedio –por ejemplo, se aplicó la lógica "y/o" a más de un ILF (se cuenta como 2 elementos).
- Se proporcionan consultas flexibles y "reporteadores" que pueden manejar peticiones complejas – por ejemplo, combinaciones de lógica "y/o" en uno o más ILF (se cuenta como 3 elementos).
- Información de control del negocio se guarda en tablas que son mantenidas por el usuario mediante procesos interactivos en línea pero los cambios se aplican sólo durante días hábiles (contar como un sólo elemento).
- Información de control del negocio se guarda en tablas que son mantenidas por el usuario mediante procesos interactivos en línea y los cambios se aplican inmediatamente (contar como 2 elementos).

4.7 Cálculo del conteo de FP's Ajustado

Las 14 GSC's se concentran en el Valor del Factor de Ajuste (Value Adjustment Factor, VAF).

Para determinar el conteo ajustado de FP's se aplica el VAF al UFP; el valor resultante tiene una desviación de ± 35 por ciento.

Una aplicación batch tendrá una calificación total (grado total de influencia, TDI) de menos de 15; aplicaciones batch tipo front-end entre 15 y 30; una aplicación interactiva, entre 30 y 45 y sistemas de control de procesos, telecomunicaciones y de tiempo real entre 30 y 60.

Los siguientes pasos resumen el procedimiento para calcular el VAF:

1. Evaluar las 14 GSC's en una escala de 0 a 5 para determinar el grado de influencia (DI) de cada una.

2. Sumar los DI's de las 14 GSC's para producir el grado total de influencia (TDI).

3. Usar el TDI de la aplicación en la siguiente ecuación para calcular el VAF:

VAF = (TDI x 0.01) + 0.65

CALCULOS Y FORMULAS PARA FP'S

A continuación se proporciona el resumen del procedimiento para el conteo de FP's:

Conteo de FP's para proyectos de Desarrollo

El conteo de FP's para un proyecto de desarrollo considera 3 componentes de funcionalidad:

1. El conteo no ajustado de FP's de la aplicación, que consiste en determinar los EI's, EO's, EQ's, ILF's y EIF's.

2. La funcionalidad de conversión, que consiste en transferir la información previa a los nuevos ILF's a través del software (este componente se refiere a la entrada de datos desde archivos viejos [contados como EI's o datos de entrada dentro de los nuevos ILF's ya contados] y posiblemente un EO para un reporte de conversión).

3. El valor del factor de ajuste de la aplicación.

Cálculos para proyectos de Desarrollo

La siguiente fórmula se utiliza para calcular el conteo de FP's en proyectos de desarrollo:

DFP=(UFP+CFP) x VAF

Donde:

- DFP es el conteo de FP's del proyecto de desarrollo.

- UFP es el conteo de FP's no ajustado.
- CFP es el número de FP's incluidos por una conversión de información
- VAF es el valor del factor de ajuste.

Conteo de FP's para proyectos de Mejora

El conteo de FP's para un proyecto de mejora considera 3 componentes de funcionalidad:

1. El conteo no ajustado de FP's de la aplicación que consiste en determinar los EI's, EO's, EQ's, ILF's y EIF's los cuales son:

    - Añadidos por el proyecto de mejora (funciones que no existían previamente – por ejemplo, un nuevo EQ, tres nuevos EI's, un nuevo ILF, un nuevo EO, etc.)
    - Cambiados por el proyecto de mejora (funciones que existían previamente pero ahora tienen diferentes campos, FTR's o que requieren un procesamiento diferente)
    - Eliminados por el proyecto de mejora (funciones que han sido eliminadas de la aplicación – por ejemplo, un reporte borrado).

2. La funcionalidad de conversión, que consiste en transferir la información previa a los nuevos ILF's a través del software (se refiere al ingreso de los datos desde archivos viejos [contados como EI's o datos de entrada dentro de los nuevos ILF's ya contados] y posiblemente un EO para el reporte de conversión).

3. Dos valores para el factor de ajuste de la aplicación (los VAF's podrían cambiar como parte del proyecto; en tal caso, habría un VAF previo y el nuevo VAF).

Cálculos para proyectos de Mejora

La siguiente fórmula se utiliza para calcular el conteo de FP's en proyectos de mejora:

$$MFP= [(ADD+ MDFA + CFP) \times VFAD] + (DEL \times VFAA)$$

Donde:

- MFP es el conteo de FP's del proyecto de Mejora.
- ADD es el conteo de FP's no ajustado de aquellas funciones Añadidas por el proyecto de mejora.
- MDFA es el conteo de FP's de aquellas funciones Modificadas por el proyecto de mejora (este componente refleja el valor de las funciones después de aplicadas las modificaciones, no sólo los campos añadidos por la modificación; un error típico es contar sólo DET's y FTR's o RET's que cambiaron, pero todo lo concerniente a la función se debe contar para considerar el esfuerzo involucrado en probar la funcionalidad existente tanto como la cambiada).
- CFP es el número de FP's incluidos por una Conversión de datos.
- VFAD es el Valor del Factor de Ajuste de la aplicación Después del proyecto de mejora.
- DEL es el conteo de FP's sin ajustar para aquellas funciones que fueron Eliminadas por el proyecto de mejora.
- VFAA es el Valor del Factor de Ajuste de la aplicación Antes del proyecto de mejora.

Conteo de FP's para proyectos de Mantenimiento

El conteo de FP's para un proyecto de mantenimiento considera 2 componentes de funcionalidad (la funcionalidad del esfuerzo de conversión no se incluye en el conteo de la aplicación por ser parte del esfuerzo de desarrollo y no de la aplicación una vez establecida):

1. El conteo no ajustado de FP's de la aplicación que consiste en determinar los EI's, EO's, EQ's, ILF's y EIF's.
2. El valor del factor de ajuste de la aplicación.

Existen dos puntos en el tiempo en los cuales se puede realizar el conteo de FP's de la aplicación:

1. Cuando la aplicación es entregada inicialmente (la aplicación no fue medida antes ni durante el desarrollo)
2. Cuando un proyecto de mejora ha cambiado el valor de la funcionalidad de la aplicación (cuando se instala el proyecto de

mejora, el conteo previo de FP's de la aplicación debe ser actualizado para reconocer las modificaciones hechas a la aplicación). El proyecto de mejora puede haber modificado la aplicación a través de:

- Añadir (nueva) funcionalidad, incrementar el tamaño de la aplicación en FP's.
- Cambiar funcionalidad; esto es incrementar, decrementar o no tener efecto en el tamaño de la aplicación en FP's.
- Borrar funcionalidad; decrementar el tamaño de la aplicación en FP's.
- Cambiar el valor del factor de ajuste a través de añadir, decrementar o no tener efecto en el tamaño de la aplicación en FP's.

Estas variaciones se ven reflejadas en las dos fórmulas del conteo presentadas a continuación.

Cálculo de FP's para aplicaciones iniciales

La siguiente fórmula se utiliza para calcular el conteo de FP's en aplicaciones sin conteo previo de FP's:

$$IFP = ADD \times VAF$$

Donde:

- IFP es el conteo de FP's de la aplicación Inicial.
- ADD es el conteo de FP's no ajustado de la funcionalidad proveída por el proyecto de desarrollo.
- VAF es el valor del factor de ajuste.

Cálculo de FP's para aplicaciones después de una mejora

La siguiente fórmula se utiliza para calcular el conteo de FP's de la aplicación después de la mejora:

$$AFP = [(NFPA + ADD + MDFD) - (MDFA + DEL)] \times VAFD$$

Donde:

- AFP es el conteo Ajustado de FP's de la aplicación.
- NFPA es el conteo de FP's no ajustado de la aplicación, Antes del proyecto de mejora.
- ADD es el conteo de FP's no ajustado de aquellas funciones Añadidas por el proyecto de mejora.
- MDFD es el conteo de FP's de aquellas funciones Modificadas por el proyecto de mejora (este componente refleja el valor de los FP's Después de aplicadas las modificaciones).
- MDFA es el conteo de FP's de aquellas funciones Modificadas por el proyecto de mejora, Antes de ser aplicados los cambios.
- DEL es el conteo no ajustado de FP's de aquellas funciones eliminadas por el proyecto de mejora.
- VAFD es el valor del factor de ajuste de la aplicación Después del proyecto de mejora.

CAPÍTULO 5

CASO PRÁCTICO

5. Caso Práctico

5.1 Introducción

Mexicana de Aviación es una de las principales empresas en el sector de Aeronáutica, a nivel mundial es considerada como una aerolínea de tamaño medio debido a su volumen de boletos vendidos y tiene presencia de oficinas de venta en toda la República mexicana, Colombia, Argentina, Brasil, Venezuela, El Salvador, Guatemala, Estados Unidos, Canadá y en la Unión Europea. Sus oficinas centrales se ubican en la Ciudad de México y mantiene convenios con otras aerolíneas para tener cobertura mundial.

Por estrategia de negocio, el departamento de Sistemas de Mexicana ha operado en outsourcing con EDS de México desde 1998 y junto con él ha establecido nuevas estrategias de desarrollo y tecnología.

A continuación se describe un área de oportunidad que EDS detectó en uno de los procesos de Mexicana de Aviación.

5.2 Situación Actual

La figura 5.1 muestra el flujo de la situación actual del seguimiento financiero de la venta de boletos:

5.2.1 Problemática de la Situación Actual

El sistema actual para controlar las ventas, CIE (Control de Ingresos y Egresos), es un sistema off-line instalado localmente en cada oficina de ventas de Mexicana. Puesto que el CIE emite reportes impresos que son enviados a la oficina matriz, la cantidad de retrabajo por capturas en los diferentes departamentos es excesiva y la explotación de la información no es oportuna, ni veraz.

Durante el proceso del corte de caja del representante Mexicana reporta grandes pérdidas por fraudes, dinero en efectivo que no es reportado,

o dólares que son reportados como pesos mexicanos ya que no se tiene forma de validar que el dinero que se está reportando corresponda con el total de las formas de pago que recibió el representante.

También se presentan fraudes durante el proceso del corte de caja de la oficina de ventas.

Debido a que el sistema CIE trabaja de manera local, los costos de mantenimiento y operación son muy altos ya que los técnicos de soporte requieren trasladarse a cada oficina para atender los problemas y recompilar la aplicación en cada PC si se necesitan cambios, esto ocasiona que no todas las oficinas tengan la misma versión del sistema y que la corrección de información se tenga que hacer en el momento de la captura en la oficina central.

Al no tener la información en línea con la oficina central, el proceso depende de que los cajeros cierren oportunamente y de la eficiencia del servicio de mensajería.

La información de los cortes de caja de oficinas de Mexicana en el interior de la República Mexicana o en el extranjero tarda de uno a cuatro meses en llegar a la oficina central para ser procesada, esto ocasiona retrasos importantes en los procesos de pago de comisiones, impuestos y cobranzas a clientes.

Las proyecciones de ocupación en los vuelos se tienen que hacer con información de cuatro meses anteriores, ya que no cuentan con la información completa de meses inmediatos por la tardanza que se tiene durante el envío por mensajería y la captura manual que se hace en la oficina central.

El mismo boleto es capturado hasta diez veces en diferentes sistemas o Excel ya que la aerolínea no cuenta con sistemas unificados que puedan compartir la información. Esto ocasiona grandes costos en nómina y en tiempo al tener personal dedicado exclusivamente a la captura en los sistemas de cada área; cuando están próximas las fechas de pago de impuestos al fisco, aumenta el pago de tiempo extra al personal para que terminen la captura a tiempo.

Flujo de Situación Actual del proceso: Seguimiento de la Venta de boletos.

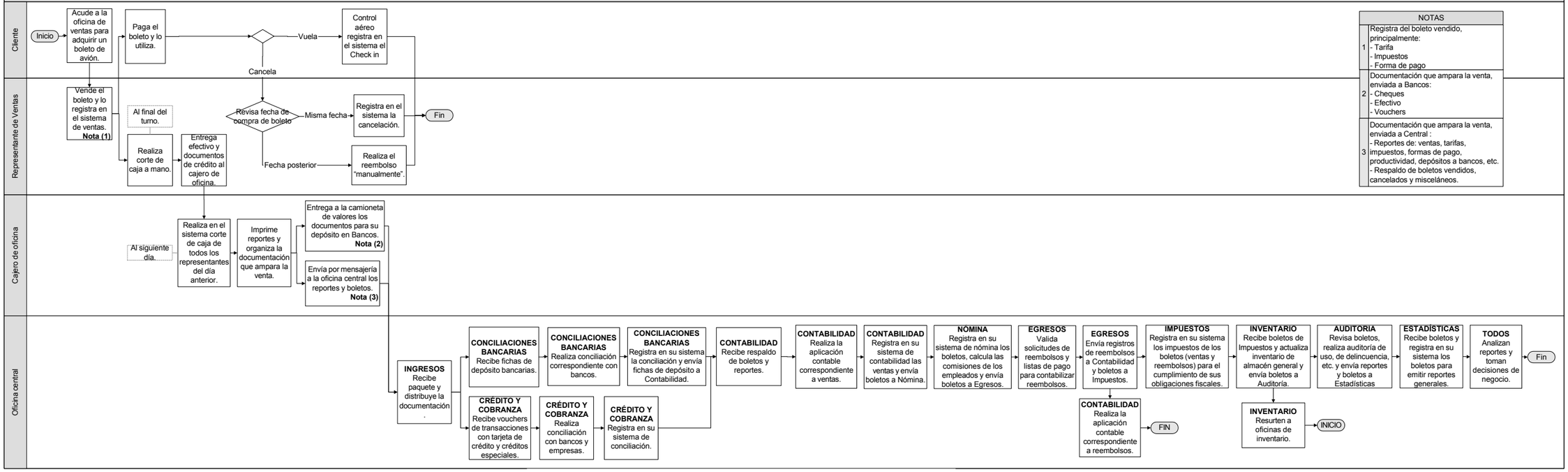


Figura 5.1 Flujo de la situación actual del seguimiento financiero de la venta de boletos.

Los ingresos de la aerolínea se ven afectados negativamente por la demora en la recepción de la información y por el pobre control que tiene sobre el manejo del efectivo en las oficinas de ventas. Para cuando los fraudes son detectados los boletos ya fueron usados.

En resumen, la problemática de la situación actual se centra en dos puntos:

1) Sistemas off-line en oficinas de ventas y corporativo
2) Fraudes por control deficiente del manejo del efectivo

NOTA: Toda la información anterior fue proporcionada por el departamento de Auditoría de Mexicana de Aviación.

5.3 Situación Deseada

La figura 5.2 muestra el flujo de la situación deseada para el seguimiento financiero de la venta de boletos:

De acuerdo con el diagrama anterior, se requiere contar con un Sistema que permita registrar de forma ágil las operaciones comerciales (Ventas, Reembolsos, Cancelaciones, Listas de pago, Depósitos bancarios y Operaciones Misceláneas) que se realizan en los diferentes puntos de venta, ofreciendo a las áreas Comerciales y Financieras un uso y administración más eficiente sobre esta información.

5.4 Solución Propuesta

Como parte de la solución[1], EDS de México propuso a Mexicana de Aviación el diseño e implementación de un sistema financiero, llamado O2C (Order to Cash), cuyos módulos soportaran la operación de las diversas áreas de negocio, de acuerdo con la relación mostrada en la tabla 5.1.

[1] La solución completa contempló también: cambios en el personal (despidos, retiros, nuevas contrataciones y capacitación) y reingeniería de procesos de negocio.

Área de negocio	Nombre Módulo	Funcionalidad
Inventario	KARDEX	Inventario de boletos
Ingresos	INGRESOS	Registro de ventas, cancelados, reembolsos y cobranzas)
	EVA	Proceso batch para la extracción de ventas automáticas
Contabilidad	CIERRER	Cortes de caja de representante y reportes
	CIERREO	Corte de caja de la oficina y reportes para Finanzas
	CIERREC	Cierre corporativo, generación de pólizas contables y reportes
Conciliaciones bancarias	CONCILIACIÓN	Conciliación bancaria, edos. de cuenta, cheques, transferencias
Crédito y cobranza	VOUCHERS	Conciliación de vouchers VI, MC, AX, Sanborns
	H2H	Proceso batch para la carga del detalle de ventas por PROSA, AMEXCO y Sanborns
	COVRA	Proceso batch para carga de vouchers
VTP's (Viajes Todo Pagado)	VTP	Pago de comisiones, facturación de proveedores, débitos

Tabla 5.1 Módulos del Sistema Financiero

Para ejemplificar el análisis por FP de un Sistema de TI realizaremos el conteo de la pantalla del corte de caja del representante (CIERRER).

5.5 Requerimientos

Llevar registro electrónico de los cortes de caja de los representantes de ventas.

Flujo de Situación Deseada del proceso: Seguimiento de la Venta de boletos.

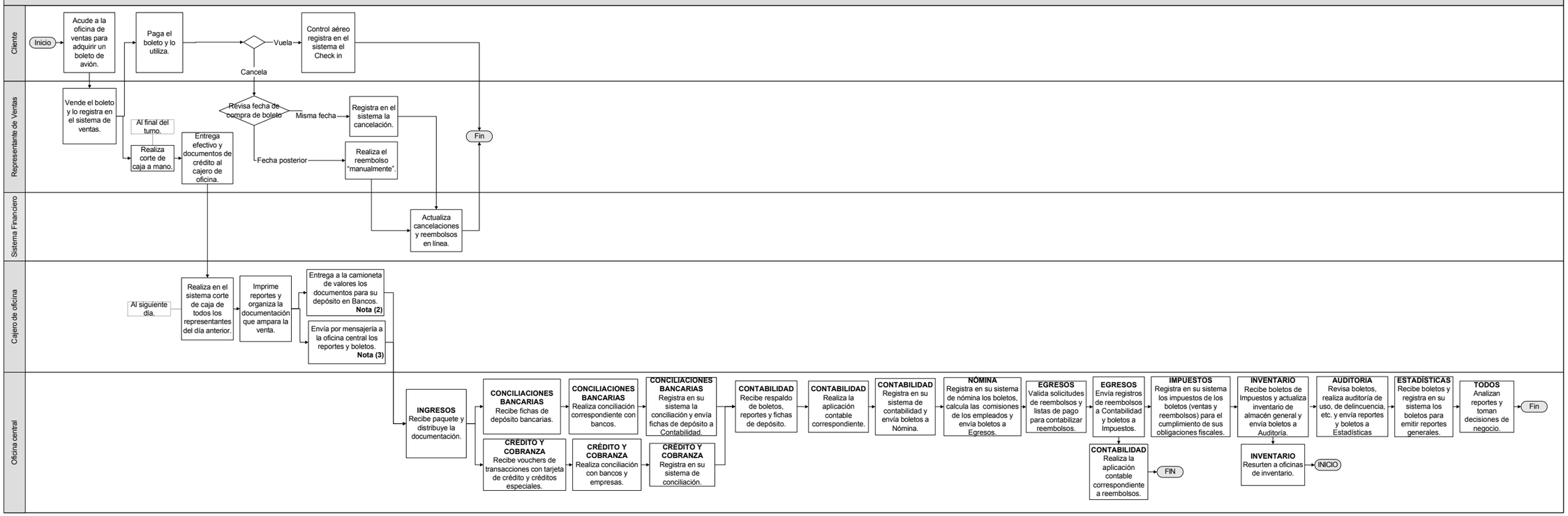


Figura 5.2 Flujo de la situación deseada del seguimiento financiero de la venta de boletos.

5.6 Proceso de Conteo

Desarrollar una pantalla que permita:

- Realizar el balance entre lo que reporta el sistema y lo que el representante declara en su corte de caja
- Consultar el detalle de las liquidaciones cerradas.

Para determinar el número de FP's que contendrá dicha pantalla se presenta el análisis siguiendo los pasos correspondientes.

1. Determinar el tipo de conteo

Es un conteo base, por tratarse de un sistema nuevo.

2. Identificar los alcances del conteo y la frontera de la aplicación.

El alcance del conteo abarca únicamente al módulo del cierre del representante.

La figura 5.3 muestra la frontera de la aplicación y los sistemas con los que interactúa.

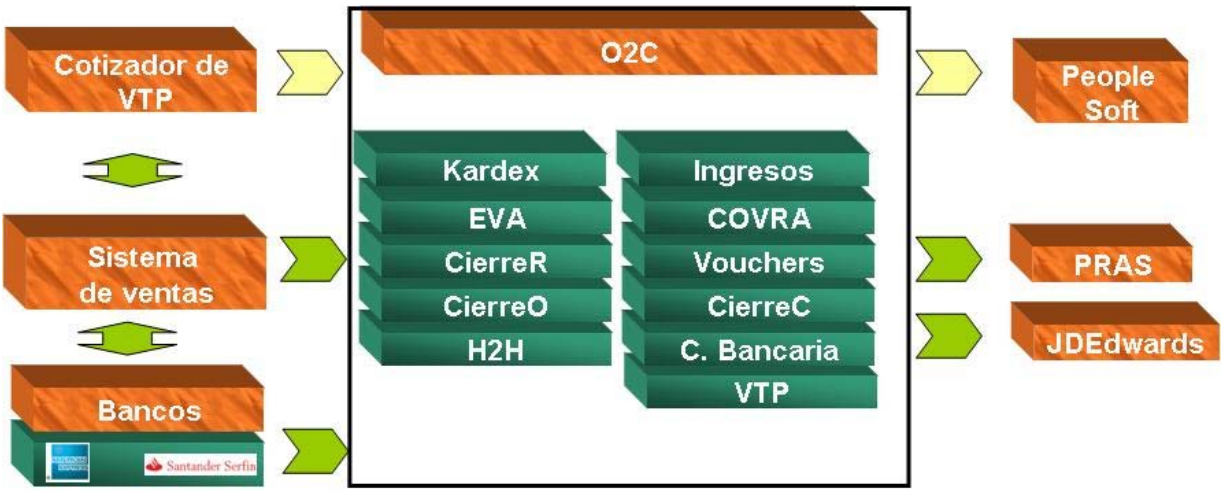


Figura 5.3 Frontera de la aplicación

Donde:
People Soft es el sistema que se encarga de la nómina.
JDEdwards es el sistema que se encarga de la contabilidad.
PRAS (Passenger Revenue Accounting System) es el sistema que se encarga de contabilizar los ingresos por transportación aérea.

3. Identificar todas las funciones de datos y su complejidad

Para identificar esta funcionalidad se deben analizar los requerimientos detallados[2] que se derivaron del requerimiento de alto nivel del usuario[3].

Las siguientes son las características que debe tener el sistema para cumplir con las necesidades del usuario:

Para el control y registro electrónico de los cortes de caja de los representantes es necesario que el sistema cuente con una pantalla para generar el cierre del representante de ventas la cual deberá registrar y totalizar los montos por cada forma de pago de todas las transacciones registradas para ese representante durante su jornada de trabajo en la oficina de ventas donde se haya firmado.

Previo a la carga inicial de datos, el sistema debe validar que:

- el representante no tenga boletos en espera de revisión
- todos los boletos que sirvan para abordar tengan su itinerario asociado
- todas las transacciones con tarjeta de crédito (vouchers) estén asociadas con las formas de pago de los boletos, esto es, marcar qué boletos fueron pagados con qué vouchers considerando el número de tarjeta, número de autorización y el monto de la transacción.

El sistema deberá permitir la captura de los importes que el representante tenga físicamente por cada forma de pago (pesos, dólares, cheques, pagarés de tarjeta de crédito, créditos especiales de Mexicana,

[2] Ver anexo 1. Requerimientos detallados
[3] Ver anexo 2. Requerimientos de alto nivel

etc.), comparar contra lo que el sistema genera en sus diferentes formas de pago y validar el cuadre de liquidación.

Se considerará que una liquidación “cuadra” cuando la diferencia entre el gran total en pesos capturado por el representante contra el gran total en pesos calculado por el sistema es igual a cero o se encuentra dentro del rango permitido (±10 centavos de dólar al tipo de cambio del día).

En caso de que existan montos en moneda extranjera deben ser convertidos a moneda nacional al tipo de cambio del día para poder hacer el “cuadre”.

Deberá permitir capturar el tipo de cambio para dólares a la compra cuando el representante reciba dólares y tenga que dar cambio en pesos. Cuando se tenga en caja una diferencia positiva de dólares con respecto a lo que indica el sistema, deberá ingresarse el detalle de cuántos dólares se compraron y el tipo de cambio de compra de los mismos, en una pantalla que contendrá la diferencia encontrada al tipo de cambio del día. El tipo de cambio podrá ser modificado por el usuario en caso de ser diferente al que se presente en la pantalla.

La diferencia del punto anterior en moneda local se substraerá del monto de efectivo en caja.

El sistema deberá permitir al usuario agregar formas de pago que no hayan sido presentadas en la carga inicial de datos.

Una vez que exista el cuadre la liquidación debe ser almacenada en el sistema para la posterior explotación de la información. El sistema deberá generar un número de liquidación que será único y marcar todos los boletos que amparan los montos de la liquidación con tal número.

Cerrada la liquidación, no se deben permitir modificaciones a los boletos cerrados ni a la propia liquidación.

Cuando el representante así lo requiera, se deberá mostrar el detalle de boletos (mediante un hipervínculo) que conforman el monto por cada forma de pago.

Debe emitir un reporte de la liquidación cerrada, bajo los criterios de búsqueda de oficina, empleado, fecha, turno o número de liquidación, el cual podrá imprimir y deberá contener la siguiente información.

- Desgloce de importes por forma de pago.
- Lista de boletos que amparan los montos de la liquidación y que fueron asociados con ella.
- Nombres del representante y el cajero de la oficina y espacio para la firma de cada uno de ellos.

El flujo de los procesos descritos en los requerimientos se muestran en los diagramas de actividad: 5.4 Cierre de representante, 5.5 Cargar datos, 5.6 Cerrar liquidación, 5.7 Consultar liquidación y 5.8 Consultar detalle de boletos.

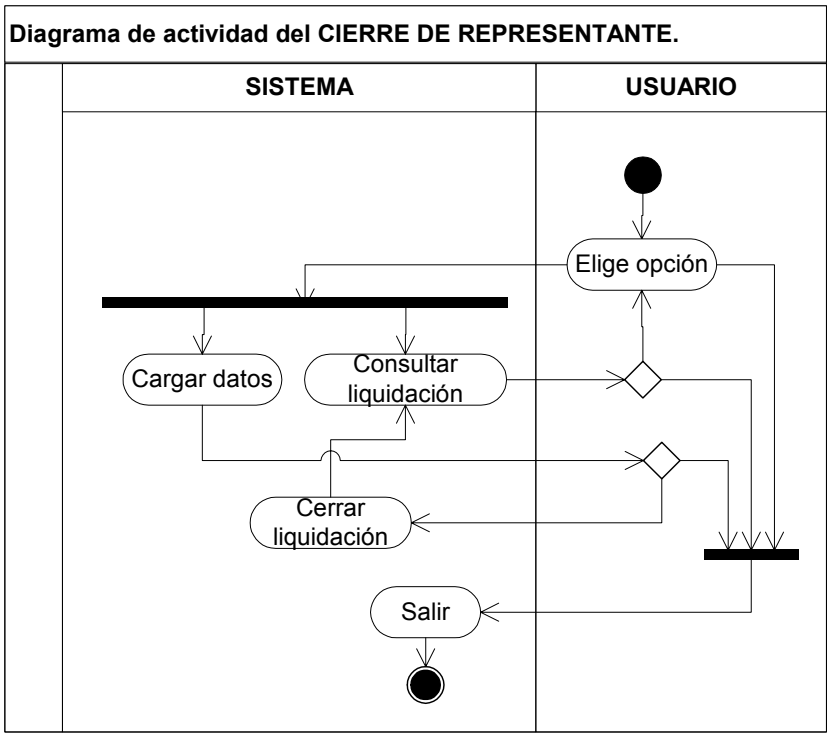


Diagrama 5.4 Cierre de representante

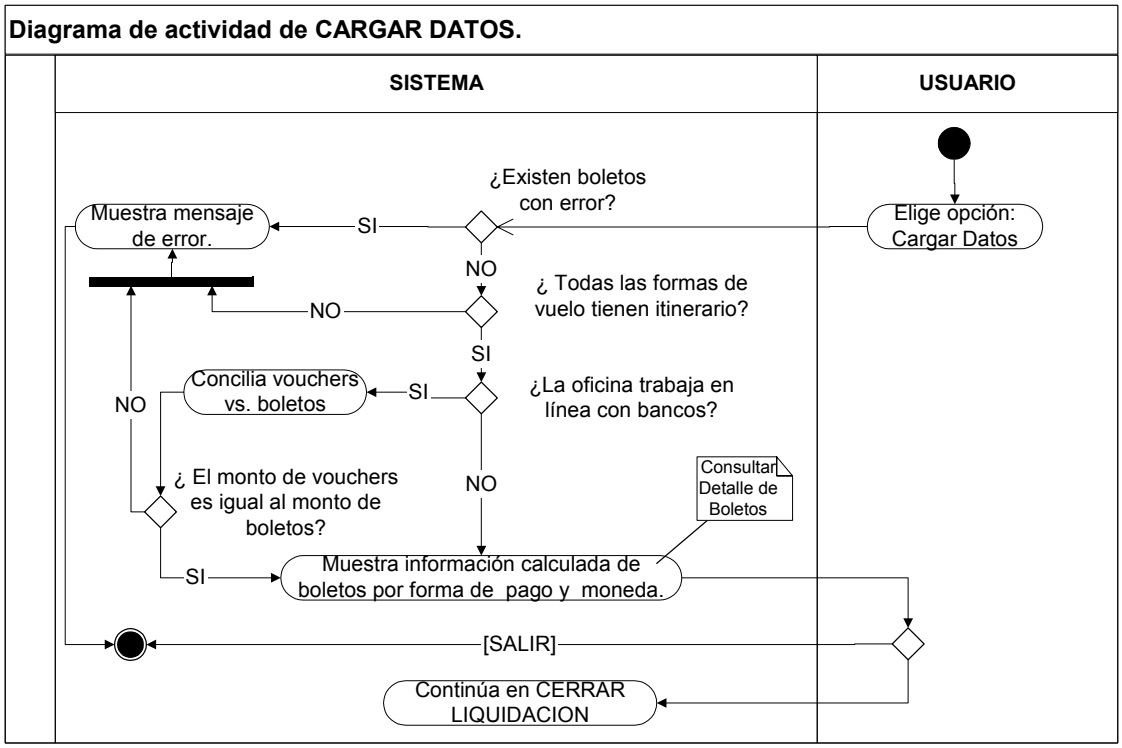


Diagrama 5.5 Cargar Datos

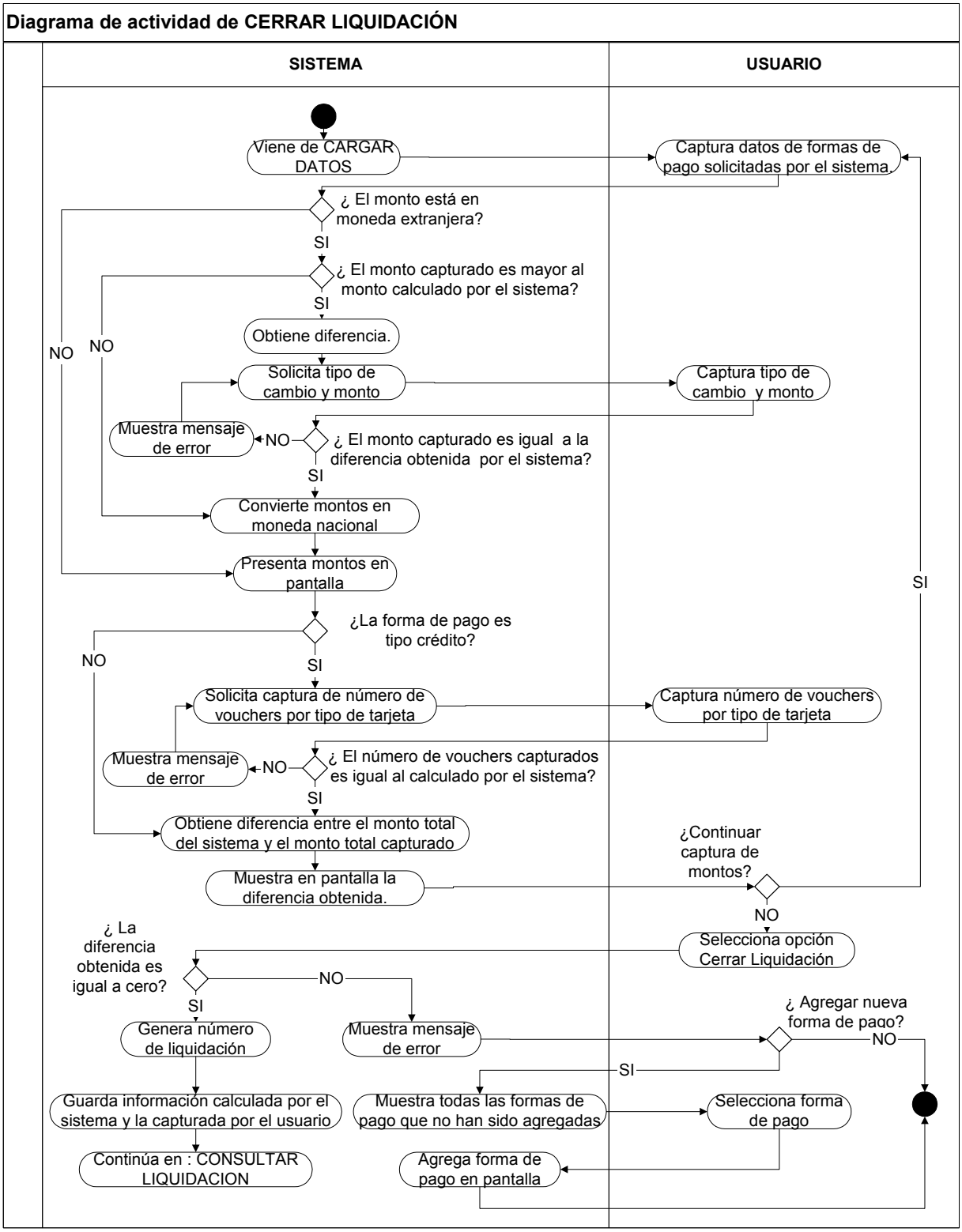


Diagrama 5.6 Cerrar Liquidación

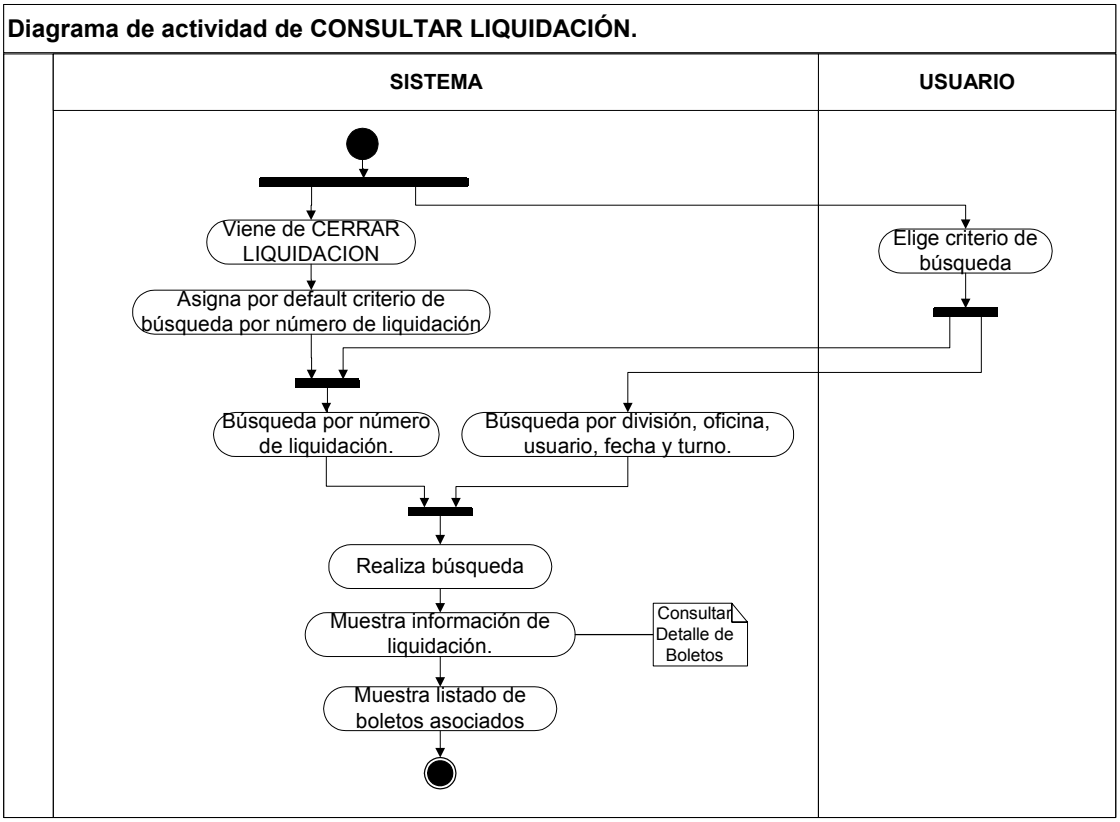


Diagrama 5.7 Consultar Liquidación

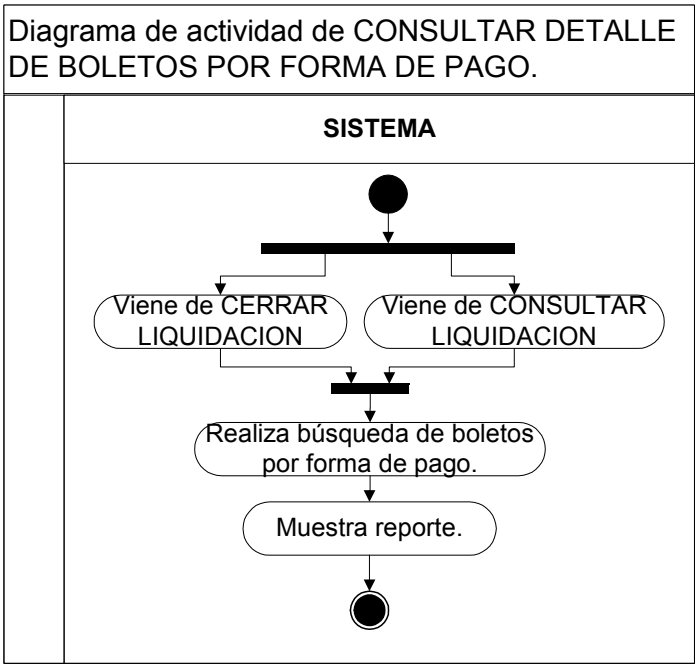


Diagrama 5.8 Consultar detalle de boletos por forma de pago

Con base en el modelo de datos[4] se identifican las siguientes funciones de datos:

- *Empleados*: contiene toda la información referente a los usuarios que tienen una firma y clave de acceso válidos para el sistema (nombre, puesto, domicilio, clave de empleado etc.)

- *Divisiones*: contiene la información referente a las divisiones geográficas (Área Metropolitana, Sur Golfo, Norte, Estados Unidos y Canadá, Sudamérica etc.) a las cuales pertenecen los puntos de venta de Mexicana (nombre y código).

- *Oficinas*: contiene la información referente a los puntos de venta donde opera el sistema (código IATA, nombre, dirección, división a la que pertenece, etc.)

- *Formas Contables*: almacena el listado de formatos para la expedición de boletos que puede usar la aerolínea (código de formato, descripción, etc.).

- *Tarjetas de Crédito*: guarda la información sobre los tipos de tarjeta aceptados como forma de pago (código, descripción BIN, etc.)

- *Datos de los Boletos*: contiene la información completa de los boletos expedidos por los representantes de ventas (impuestos, formas de pago, comisiones, itinerario, nombre del pasajero, tarifas, etc.).

- Cobranza Diaria: registra los datos de los ingresos recibidos en los puntos de venta por concepto de pago de créditos que otorga Mexicana a personas morales. (número de cliente, monto pagado, número de estado de cuenta, etc.)

- *Liquidación de Representante*: almacena el detalle de los cortes de caja de los representantes de ventas (código de empleado, fecha de corte, montos por forma de pago, etc.).

[4] Ver anexo 3. Modelo de datos

- *Catálogo de Formas de Pago*: guarda la información sobre las formas de pago aceptadas en los puntos de venta (código de forma de pago, descripción, etc.)

- *Vouchers*: contiene los datos de las transacciones con tarjeta de crédito que ejecutan los representantes de venta (número de tarjeta, nombre del tarjetahabiente, monto, fecha, etc.).

- *Boletos en espera de revisión*: almacena los datos de los boletos que presentan algún error derivado de las validaciones previas al cierre de la liquidación (código de error, número de boleto, impuestos, tarifas, formas de pago, etc.)

Complejidad y Contribución de los ILF's y EIF's

La complejidad de las funciones de datos identificadas en el conteo se muestra en la tabla 5.2

				Complejidad			
Descripción	**Tipo de Función**	**DET**	**RET**	**B**	**M**	**A**	**Tabla de Referencia**
Catálogo de Empleados	ILF	25	2	0	1	0	emp emp_agcy
Catálogo de Divisiones	ILF	3	1	1	0	0	div
Catálogo de Oficinas	ILF	17	1	1	0	0	agcy_master
Catálogo de Formas Contables	ILF	9	1	1	0	0	stock_form
Catálogo de Tarjetas de Crédito	ILF	7	2	1	0	0	cc_type credit_card
Tabla de Datos de los Boletos	ILF	84	7	0	0	1	sale_doc misc_fee sold_cpn tran_fop detl_misc tax_commision error_sale_doc

Tabla de Cobranza Diaria	ILF	16	1	1	0	0	collected_daily
Tabla de Liquidación de Representante	ILF	43	4	0	1	0	emp_liq detail_emp_liq emp_liq_back detail_emp_liq_back
Catálogo de Formas de Pago	ILF	4	1	1	0	0	fop
Tabla de vouchers	ILF	22	1	1	0	0	covra_trans
Boletos en espera de revisión	ILF	89	9	0	0	1	eva_detl_misc eva_error_profiles eva_error_sale_doc eva_errors eva_misc_fee eva_sale_doc eva_sold_cpn eva_tax_commision eva_tran_fop

Tabla 5.2 **Complejidad y Contribución de los ILF's y EIF's**

4. Identificar las funciones transaccionales y su complejidad.

Con base en la funcionalidad solicitada en los requerimientos detallados[5] y los prototipos de pantalla[6], se identifican las siguientes funciones transaccionales:

- *Mostrar Listado de Agentes*: se clasifica como EQ que consulta 1 ILF y está compuesto por 2 DET del ILF + 1 DET por la capacidad de enviar mensajes + 2 DET por información de control (división y oficina) + 1 DET por combo box.

- *Mostrar Listado de Divisiones*: se clasifica como EQ que consulta 1 ILF y está compuesto por 2 DET del ILF + 1 DET por la capacidad de enviar mensajes + 1 DET por combo box.

- *Mostrar Listado de Oficinas*: se clasifica como EQ que consulta 1 ILF y está compuesto por 2 DET del ILF + 1 DET por la capacidad de

[5] Ver anexo 1. Requerimientos detallados
[6] Ver anexo 4. Prototipos de pantalla

enviar mensajes + 1 DET por información de control (oficina) + 1 DET por combo box.

- *Validar que no existan boletos con error*: se clasifica como EQ que consulta 1 ILF y está compuesto por 3 DET del ILF + 1 DET por la capacidad de enviar mensajes + 1 DET por el botón de función + 2 DET por información de control (empleado y oficina).

- *Validar que todas las formas de vuelo tengan itinerario asociado:* se clasifica como EQ que consulta 2 ILF (datos de boleto, formas) y está compuesto por 1 DET (cat. Formas para checar si es forma de vuelo) + 3 DET (cat. datos grales. del boleto- empleado, agencia, # forma) +1 DET (cat. Cupones- para checar que exista al menos un itinerario) + 1 DET por mensajes error + 2 DET por información de control (empleado, agencia) + 1DET por botón de función.

- *Conciliar vouchers vs boletos*: se clasifica como EI que consulta 4 ILF (empleados, datos boletos, datos voucher, cat. Tarjetas crédito) y está compuesto por 13 DET (datos de voucher y boletos) + 2 DET por información de control (empleado, agencia) + 1 DET por la capacidad de enviar mensajes de error + 1 DET por campo mantenido (num. Voucher) + 1DET por botón de función.

- *Mostrar información de montos por forma de pago y por moneda*: se clasifica como EO que consulta 4 ILF (datos grales. del boleto, formas de pago, tipo tarjeta, cobranza) y está compuesto por 13 DET por datos calculados + 1 DET por la capacidad de enviar mensajes + 2 DET por información de control + 1DET por botón de función + 3 DET por cajas de texto en reporte final (moneda nacional, moneda extranjera, núm. de vouchers).

- *Validar que no exista una diferencia positiva en moneda extranjera*: se clasifica como EO que consulta 1 ILF y está compuesto por 1 DET por dato capturado + 1 DET por forma de pago + 1 DET por tipo moneda + 1 DET por botón de función (botón o tabulador) + 1 DET por dato de la diferencia calculada.

- *Justificar diferencia positiva en moneda extranjera*: se clasifica como EO que consulta 1 ILF y está compuesto por 1 DET por campo de

solicitud de tipo de cambio + 1 DET por campo donde captura el monto + 1 DET por capacidad de enviar mensajes + 1 DET por botón de función + 1 DET por dato calculado de conversión + 1 DET por resultado de la diferencia + 1 DET por botón de agregar para ingresar más de 1 tipo de cambio)

- *Convertir montos a moneda nacional*: se clasifica como EO que consulta 1 ILF de liquidación de representante (donde está el tipo de cambio) y está compuesto por 1 DET por monto recibido + 1 DET por monto calculado + 1 DET por clave de forma de pago + 1 DET por moneda + 1 DET por botón de función.

- *Conciliar número de vouchers por tipo de tarjeta*: se clasifica como EO que consulta 1 ILF (vouchers) y está compuesto por 1 DET por dato capturado + 1 DET por capacidad de enviar mensajes + 1 DET por dato calculado de núm. Vouchers + 2 DET por información de control (forma pago y tipo tarjeta)

- *Calcular balance de liquidación*: se clasifica como EO que consulta 5 ILF (datos grales boleto, formas de pago, tipo tarjeta, cobranza, vouchers) y está compuesto por 2 DET por datos calculados (total en moneda nacional por forma de pago, Núm. de vouchers por tarjeta) + 1 DET por capacidad de enviar mensajes + 2 DET por información control (empleado, oficina) + 1DET por botón de función + 3 DET por cajas de texto en reporte final (moneda nacional, moneda extranjera, núm. de vouchers).

- *Agregar forma de pago*: se clasifica como EQ que consulta 1 ILF (formas de pago) y está compuesto por 1 DET por información de control (forma de pago) + 1 DET por capacidad de enviar mensajes + 1 DET por botón de función + 4 DET por información mostrada (2 campos con información del sistema y 2 de cajas de texto para captura).

- *Guardar detalle de liquidación*: se clasifica como EI que consulta 4 ILF (liquidación, datos del boleto, vouchers y cobranza) y está compuesto por 19 DET de datos de la liquidación + 2 DET por datos de boleto (marca boleto y cambia estatus) + 2 DET por voucher

(marca voucher y cambia estatus) + 2 DET por cobranza (marca cobranza y cambia estatus)

- *Consultar Liquidación cerrada*: se clasifica como EQ que consulta 1 ILF de liquidación de agente y está compuesto por 15 DET por contenido del reporte + 1 DET por botón de función + 6 DET por información de control (parámetros de búsqueda).

- *Mostrar listado de boletos asociados a la liquidación cerrada*: se clasifica como EQ que consulta 1 ILF (datos de boleto) y está compuesto por 2 DET de datos.

- *Mostrar detalle de boletos por forma de pago*: se clasifica como EQ que consulta 3 ILF (datos de boleto, formas valorables y tarjetas de crédito) y está compuesto por 6 DET de datos + 1 DET por información de control + 1 DET por la capacidad de enviar mensajes + 1 DET por botón de función.

Complejidad y Contribución de los EI's, EO's y EQ's

La complejidad de las funciones transaccionales identificadas en el conteo se muestra en la tabla 5.3

DESCRIPCIÓN	Tipo de Función	FTR	DET	Complejidad		
				B	M	A
Mostrar Listado de Agentes	EQ	1	6	1	0	0
Mostrar Listado de Divisiones	EQ	1	4	1	0	0
Mostrar Listado de Agencias	EQ	1	5	1	0	0
Validar que no existan boletos con error	EQ	1	7	1	0	0
Validar que todas las formas de vuelo tengan itinerario asociado	EQ	2	9	0	1	0
Conciliar vouchers vs boletos	EI	4	18	0	0	1
Mostrar información de montos por forma de pago y por moneda	EO	4	20	0	0	1
Validar que no exista una diferencia	EO	1	5	1	0	0

positiva en moneda extranjera.						
Justificar diferencia positiva en moneda extranjera.	EO	1	7	1	0	0
Convertir montos a moneda nacional	EO	1	5	1	0	0
Conciliar número de vouchers por tipo de tarjeta	EO	1	5	1	0	0
Calcular balance de liquidación	EO	5	9	0	0	1
Agregar forma de pago.	EQ	1	7	1	0	0
Guardar detalle de liquidación	EI	4	25	0	0	1
Consultar Liquidación cerrada	EQ	1	22	0	1	0
Mostrar listado de boletos asociados a la liquidación cerrada.	EQ	1	2	1	0	0
Mostrar detalle de boletos por forma de pago.	EQ	3	9	0	1	0

Tabla 5.3 **Complejidad y Contribución de los EI's, EO's y EQ's**

5. Determinar el conteo de FP's "sin ajustar"

Con base en el análisis anterior se obtiene el siguiente resultado del conteo de FP's sin ajustar, el cual se muestra en la tabla 5.4.

	BAJO			MEDIO			ALTO			TOTAL
	#	Peso	Subtotal	#	Peso	Subtotal	#	Peso	Subtotal	
ILF's	7	7	49	2	10	20	2	15	30	99
EIF's	0	5	0	0	7	0	0	10	0	0
EI's	0	3	0	0	4	0	2	6	12	12
EO's	4	4	16	0	5	0	2	7	14	30
EQ's	6	3	18	3	4	12	0	6	0	30
			83			32			56	
									UFP =	**171**

Tabla 5.4 **Conteo de FP's sin ajustar**

6. Determinar el valor del factor de ajuste

El grado de influencia de las características generales del sistema es el siguiente:

- *Comunicación de datos*: se evalúa con 4 debido a que se trata de una aplicación en línea que trabaja sobre el protocolo TCP/IP.

- *Procesamiento de datos distribuidos:* se evalúa con 4 debido a que se trata de una aplicación WEB con peticiones y respuestas entre el cliente y el servidor.

- *Desempeño:* se evalúa con 5 debido a que las pantallas de la aplicación deben tener un tiempo de respuesta de 2 a 10 segundos y durante el desarrollo se utilizan herramientas para analizar el desempeño y consumo de recursos de las transacciones.

- *Configuración de Parámetros en los Recursos del sistema:* se evalúa con 5 debido a que el sistema comparte sus recursos de hardware (Web Server y base de datos) con otros sistemas por lo que se requiere de una configuración especial para administrar el uso de disco duro, procesadores y memoria RAM.
- *Rango de Transacciones:* se evalúa con 5 debido a que se estima que la aplicación tendrá un gran número de transacciones diarias con 3 horarios “pico”.

- *Datos ingresados “on line”:* se evalúa con 5 debido a que aproximadamente el 85% de los datos son ingresados “on-line”.

- *Eficiencia del usuario final:* se evalúa con 4 debido a que se tienen un gran número de consideraciones para mejorar los tiempos de captura del usuario en el sistema.

- *Actualizaciones “on-line” :* se evalúa con 4 debido a que el 85 % de las transacciones son actualizadas on-line y se realizan 3 respaldos de información diarios.

- *Procesamientos Complejos:* se evalúa con 3 debido a que el sistema cumple con 3 aspectos de esta característica.

- *Reusabilidad:* se evalúa con 4 debido a que la mayoría del código está diseñado para que sea re-utilizado por otros componentes. Se utiliza programación orientada a objetos.

- *Facilidad de Instalación:* se evalúa con 1 debido a que para la instalación del sistema se requiere migración de datos pero no es un requerimiento del usuario.

- *Facilidad de Operación:* se evalúa con 5 debido a que todos los procesos de mantenimiento y respaldo del sistema son automáticos.

- *Múltiples Sitios:* se evalúa con 1 debido a que la aplicación requiere de ambientes de hardware y software idénticos para funcionar.

- *Facilidad de Cambios:* se evalúa con 3 debido a que los parámetros de consulta utilizados son dinámicos.

La tabla 5.5 presenta el grado de influencia proporcionado por las características del sistema.

	DI
1. Comunicación de Datos	4
2. Procesamiento de Datos Distribuidos	4
3. Desempeño	5
4. Configuración de Parámetros en los Recursos del Sistema	5
5. Rango de Transacciones	5
6. Datos Ingresados “on – line”	5
7. Eficiencia del usuario final	4
8. Actualizaciones “on-line”	4
9. Procesamientos Complejos	3
10. Reusabilidad	4
11. Facilidad de Instalación	1
12. Facilidad de Operación	5

13. Múltiples Sitios	1
14. Facilidad de Cambios	3
Grado de Influencia Total (TDI) =	**53**

Tabla 5.5 Grado de influencia de las características del sistema

Por lo tanto, el valor del factor de ajuste es el siguiente:

VAF = (53 x 0.01) + 0.65

VAF = 1.18

NOTA: Este factor representa la influencia del ambiente en el cual trabajará el sistema, influencia que no es tomada en cuenta durante el análisis de la funcionalidad, por lo tanto su valor varía de acuerdo a cada ambiente de trabajo.

7. Calcular el conteo de FP's ajustado

Por tratarse de una aplicación nueva, se utiliza la siguiente expresión para determinar el conteo de FP's ajustado:

IFP = ADD x VAF

Donde:

- IFP es el conteo de FP's de la aplicación Inicial.
- ADD es el conteo de FP's no ajustado de la funcionalidad proveída por el proyecto de desarrollo.
- VAF es el valor del factor de ajuste.

IFP = 171 x 1.18 = 201.78

AFP = 202

El resultado obtenido representa el tamaño, desde el punto de vista funcional, que tendrá la pantalla para realizar el cierre de los representantes de las oficinas de venta.

Con la determinación del tamaño en FP's el análisis termina, pero cabe hacer algunas observaciones al respecto:

Los 202 FP's obtenidos representan el tamaño de la funcionalidad a desarrollar. Dentro de este conteo están incluidos los componentes que serán reutilizables (ILF's, EIF's) o que se cuentan una sola vez en todo el análisis (ej. Factor de ajuste VAF). Por lo tanto, en el conteo de los subsecuentes componentes del sistema no deberá ser considerada la contribución en FP's de las mismas funciones de datos y de las características generales del sistema.

Tomando en cuenta que la industria de TI considera los siguientes valores mostrados en la tabla 5.6 como estándares para la métrica de productividad[7]:

FP / Mes (ISBSG Industry Standard)	12.5
Horas / Mes	145
Días / Mes	20
Horas / Día	7.25
Horas / FP	11.6

Tabla 5.6 Estándares de productividad para la industria de TI

Entonces, para esta pantalla se tiene lo siguiente:

1 mes = 12.5 FP
? mes = 202 FP

Por lo tanto, se requeriría un ingeniero durante 16.2 meses para desarrollar toda la funcionalidad estimada.

Considerando que otro estándar de la industria es que el costo mensual por un ingeniero es de 3600 USD, el costo de la pantalla sería:

[7] http://www.ifpug.org/ifpug

3600 x 16.2 = 58 320 USD

El esfuerzo requerido para el desarrollo de esta pantalla sería entonces de 60,000 USD en un lapso de 16 meses. Estos datos son para una estimación inicial que servirá de base para el resto del análisis del sistema y al final se tendrá un tamaño y costo aproximado de toda la funcionalidad que provee la aplicación al usuario.

En la práctica las empresas consideran además sus propios datos históricos o variables de entorno para ajustar esta base a su caso particular.

Debido a que el manejo de estas variables para obtener conteos a la medida queda fuera del alcance de este trabajo de tesis, se proporciona información anexa respecto a las herramientas que existen actualmente en el mercado para realizar dicha tarea[8].

[8] Ver anexo 5. Herramientas de software para el cálculo de métricas

CONCLUSIONES

CONCLUSIONES

Para que el área de sistemas o una empresa que desarrolle software pueda asegurar a sus clientes la entrega de productos de calidad, es necesario que durante todos los procesos involucrados en el ciclo de vida del proyecto se hayan seguido las mejores prácticas que dictan las metodologías según la naturaleza del proceso.

El análisis del tamaño de un proyecto mediante FPA asegura a todos los involucrados en el negocio que al final del conteo se tendrá:

- un número que representa fielmente toda la funcionalidad del sistema solicitada por el cliente.

Y, de manera adicional:

- una primera versión de la documentación técnica (el modelo de datos y el modelo de transacciones) de cada proceso elemental analizado, dependiendo del tipo del conteo.

- la documentación completa del negocio que se automatiza con el sistema (reglas de negocio, niveles de servicio, características del ambiente de implementación).

Esta información no se tendría con otras metodologías en la fase inicial del ciclo de vida del proyecto, lo que hace a FPA la metodología más completa en cuanto a definición del sistema se refiere.

La generación de esta documentación hace que el proceso de estimación por FP's requiera más tiempo que cualquier otra de las metodologías planteadas en este trabajo pero no hay que perder de vista que esta información además de sustentar los resultados del conteo, será la base para comenzar el desarrollo de la solución una vez que se llegue a un acuerdo con el cliente.

La estimación del tamaño del software será más precisa en la medida en que los requerimientos detallen con claridad la funcionalidad esperada por el cliente, que la documentación del negocio (reglas de negocio, niveles

de servicio, etc.) se encuentre bien establecida y que el personal involucrado esté capacitado en la técnica del conteo y conozca el negocio.

A partir del tamaño en FP's se determina el tiempo y el costo de implementación, los dos factores más importantes para los actores del negocio y del sistema, pues les permite analizar la factibilidad del proyecto, planear la distribución del esfuerzo de desarrollo y determinar el balance entre el esfuerzo requerido para construir la aplicación y la ganancia obtenida.

Es preciso basarse en la última versión liberada por la IFPUG de las reglas de conteo para capacitar al personal en los nuevos tópicos y así poder estimar proyectos de las nuevas tecnologías que estén disponibles en el mercado.

Parte de una buena cultura de calidad reside en que las empresas cuenten con un registro histórico de sus propias métricas, esto implica tener una buena documentación de todos los procesos que se involucran en el ciclo de vida del sistema, desde la propuesta inicial hasta los manuales de usuario. Si se tiene esta información, los futuros conteos serán más precisos.

En México, la mayoría de las empresas que invierten en una nueva solución de software optan por aquella que les implique el menor costo en el menor tiempo, dejando en segundo plano la calidad del producto terminado.

Para que en México la metodología de FPA sea adoptada con éxito por la industria de TI se requerirá un cambio en la cultura de los dueños del negocio para darle mayor prioridad al aspecto de la calidad, tanto de los procesos como del producto terminado, sobre el tiempo y costo.

ANEXOS

ANEXOS

1. Requerimientos detallados

Cliente	Área Comercial y de Finanzas	**# de requerimiento:**	**1.2.4.1 Liquidación de Representante**
Sistema (Nombre/módulo):	Sistema o2c-omx Módulo: Cierre de Representante Proceso: Liquidación de Representante	**Fecha de apertura del requerimiento:**	19/04/02

Fuente del Requerimiento: Documento de requerimientos de alto nivel

Iniciado por: Cliente ■ EDS ☐

Prioridad: Alta ■ Media ☐ Baja ☐

Servicio al que pertenece el requerimiento:

Reporte de problema ☐ Mantenimiento o de sistema ☐ Desarrollo de Nuevo Sistema ■

Tipo de requerimiento:

Técnico ☐ Negocio ■ Planeación ☐ Otro ☐

Descripción del requerimiento

- Realizar pantalla para generar el cierre por representante
  - Registrar y totalizar los montos de las formas de pago de todas las transacciones hechas por el representante de ventas
  - Permitir la captura de los importes que se tengan en caja y el tipo en que estos estén
  - Realizar la validación contra lo que el sistema genera en sus

diferentes formas de pago.
    - Validar el cuadre de liquidación. Ésto es que las cantidades ingresadas deben cuadrar con lo que marca el sistema. Realiza las conversiones necesarias de moneda.

- Mostar detalle de boletos (mediante liga).
    - La Pantalla debe mostrar, cuando el usuario lo requiera, la lista de boletos que conforman el monto por cada forma de pago incluyendo reembolso, también debe poder revisarse hasta el detalle de cada boleto.

- Emitir reporte de liquidación
    - Deberá imprimirse la pantalla de liquidación conteniendo:
        - Desglose de los importes tal como lo muestra en pantalla.
        - Desglose de boletos que comprenda dicha liquidación.
        - Nombre de representante y espacio para firmar.
        - Espacio para nombre y firma del Cajero.

- Capturar tipo de cambio para dólares de compra.
    - Cuando se tenga en caja una diferencia positiva en dólares con respecto a lo que indica el sistema, deberá ingresarse el detalle de cuántos dólares se compraron y el tipo de cambio de los mismos, en una pantalla con las siguientes características:
        - La pantalla contendrá la diferencia encontrada al tipo de cambio del día.
        - El tipo de cambio podrá ser modificado por el usuario en caso de ser diferente al que en la pantalla se presente.

- La diferencia del punto anterior en moneda local se substraerá del monto de efectivo en caja.

- Permitir un margen de 10 cts. USD como diferencia en el cuadre de la liquidación por la pérdida de centavos en las conversiones de moneda local a dólares
- Permitir consulta de liquidaciones cerradas
- No permitir modificaciones a liquidaciones cerradas
- Permitir la elección de fechas para la consulta o el cierre de

liquidación
- Permisos para usuarios
  - El cierre podrá ser realizado por cajero o por representante
  - El representante deberá tener solo privilegio para realizar el cierre y todas las opciones de este cierre solo para su información
  - El cajero deberá tener privilegio para realizar el cierre y todas las opciones de este cierre, para cualquier representante de la oficina a la que pertenezca.

Justificación/Antecedentes
El sistema necesita conciliar las ventas de cada representante con lo que se encuentra en caja en cada forma de pago y cada moneda.

Restricciones para solución
(*Fecha esperada de entrega, calendario, presupuesto, desempeño, recursos, otros proyectos, etc.)*
- Se requiere que la información de todas las transacciones (manuales y automáticas) se encuentren en la base de datos, ya sea ésta por proceso automático o captura del mismo usuario.

Criterios de aceptación
Que las gerencias de la Comercial y de Finanzas conozcan y aprueben el prototipo de las pantallas planteadas para el desarrollo del sistema así como el análisis correspondiente.

2. Requerimientos de alto nivel

Definición de Alcance del Proyecto

Proyecto **:** O2C-OMX

Visión

Establecer las bases tecnológicas y operativas para el desarrollo de nuevas aplicaciones bajo el esquema de Internet, tomando en cuenta puntos como la metodología de implementación y administración de la plataforma tecnológica que lo soportará, traduciendo este nuevo esquema en oportunidades para nuestro cliente de incursionar en el uso de tecnologías que apoyen de forma más eficiente la operación de las oficinas de ventas de la aerolínea.

Alcance del proyecto

Proveer a Mexicana de una herramienta que permita de forma ágil el registro de las operaciones comerciales (Venta, Reembolso, Cancelado, Lista de pago, Depósitos bancarios y Misceláneos), que se realizan en los diferentes puntos de venta, ofreciendo a las áreas Comerciales y Financieras un uso y administración más eficiente sobre esta información.

Se generará un sistema bajo esquema Internet/intranet el cual muestre en forma sencilla, amigable y sobre todo confiable la información de la venta automática y manual de las oficinas de Mexicana de Aviación, que pueda de manera fácil y veraz realizar el cierre o balance por agente de ventas y por oficina, cuadrando todos los aspectos, tanto en documentos físicos que apoyan a la venta, como en el tipo de pago como es dinero en efectivo, dólares, crédito y demás formas de pago que ofrece Mexicana de Aviación. Este sistema deberá cubrir los actuales procesos que soporta el sistema CIE, pero debidamente depurados y optimizados con la información almacenada de manera centralizada.

Funciones principales de los agentes involucrados en la Oficinas de Ventas.

Representante de ventas

a) En esta fase, el representante sustituirá el procedimiento de liquidación en papel de venta actual, ésta la ingresará directamente en el módulo de Cierre por Representante inmediatamente de ingresar la venta manual del día de cierre.

Cajero oficinas MX

a) Control Electrónico del Inventario de Boletos de la Oficina.
b) Realizar las actividades de recepción de las liquidaciones de cada representante (dinero y soportes físicos de boletos y formas de pago) e ingreso de información en nuevo módulo de cierre de venta.
c) Ingresar adicionalmente fichas de depósitos bancarios (de efectivo y crédito) así como los sobres de crédito que envía a Xola.
d) Envío de documentación soporte a áreas solicitantes a más tardar al siguiente día de operación
e) Apoyo a aclaraciones solicitadas por las áreas administrativas y contables de MX

Entregables:

Durante el transcurso del proyecto se generará la siguiente documentación: reporte de observaciones para cada aplicación que ingrese al proceso de pruebas y control de la calidad y reporte de resultados según herramientas, por lo que no se tendrá que esperar al cierre del proyecto para obtener beneficios tangibles en este proyecto.

Entregables del Proyecto de Alto Nivel

1. Plan de trabajo
2. Roles y responsabilidades

3. Desarrollo del sistema de acuerdo a los Requerimientos detallados y análisis con FP's de la funcionalidad de los siguientes módulos:
   - Inventario de boletos kardex
   - Ingresos (ventas manuales, misceláneas y cobranza de créditos de mexicana)
   - Egresos (solo se contemplará la lista de pagos concentrada y la captura de los reembolsos realzados por la oficina)
   - Cierre representante de ventas
   - Cierre por oficina
   - Catálogos
   - Configuración del sistema
   - Seguridad del sistema
   - Productividad y desempeño
4. Capacitación a alas de america
5. Capacitación a centro de atención tecnológica
6. Documentación del proyecto

Productos, Servicios, Operaciones y Organizaciones NO afectadas por el proyecto

- **No se contabilizaran LA VENTA percibidos directamente desde el sistema O2C-OMX, esto se hará mediante la generación de la cinta BSP que se ingresará a PRAS**

Beneficios.

• Mejorar la imagen de la empresa (trabajo manual, organización, etc.).

• Herramientas para agilizar cierres diarios y liquidaciones de venta en las oficinas de boletos.

• Almacenamiento de boletos ATB y captura única de boletos manuales en un sistema común.

• Captura única de datos en las oficinas de boletos para su explotación en XOLA

• Generación del reporte de ventas automatizado en Xola una vez que las oficinas hayan realizado su cierre.

- Consultas en línea para conocer el detalle de la información de ventas de la oficina desde nivel agente.
- Mayor control en el KARDEX

Factores Críticos o Riesgos

Existen algunos factores críticos que deben tomarse en cuenta para llevar a cabo las actividades del proyecto, por lo que es de suma importancia que el equipo de trabajo tome conciencia de la solución de éstos.

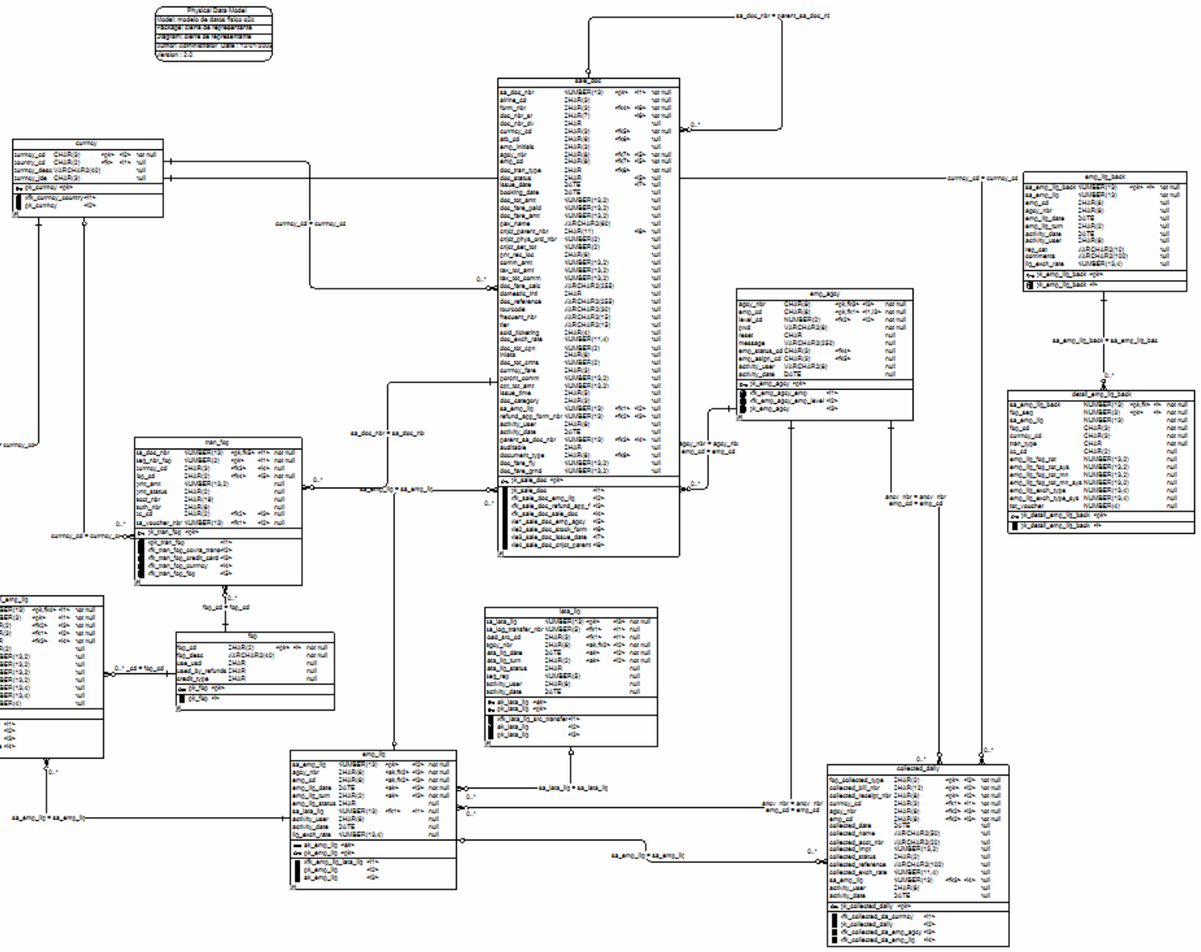
Entre los riesgos identificados por el momento son:

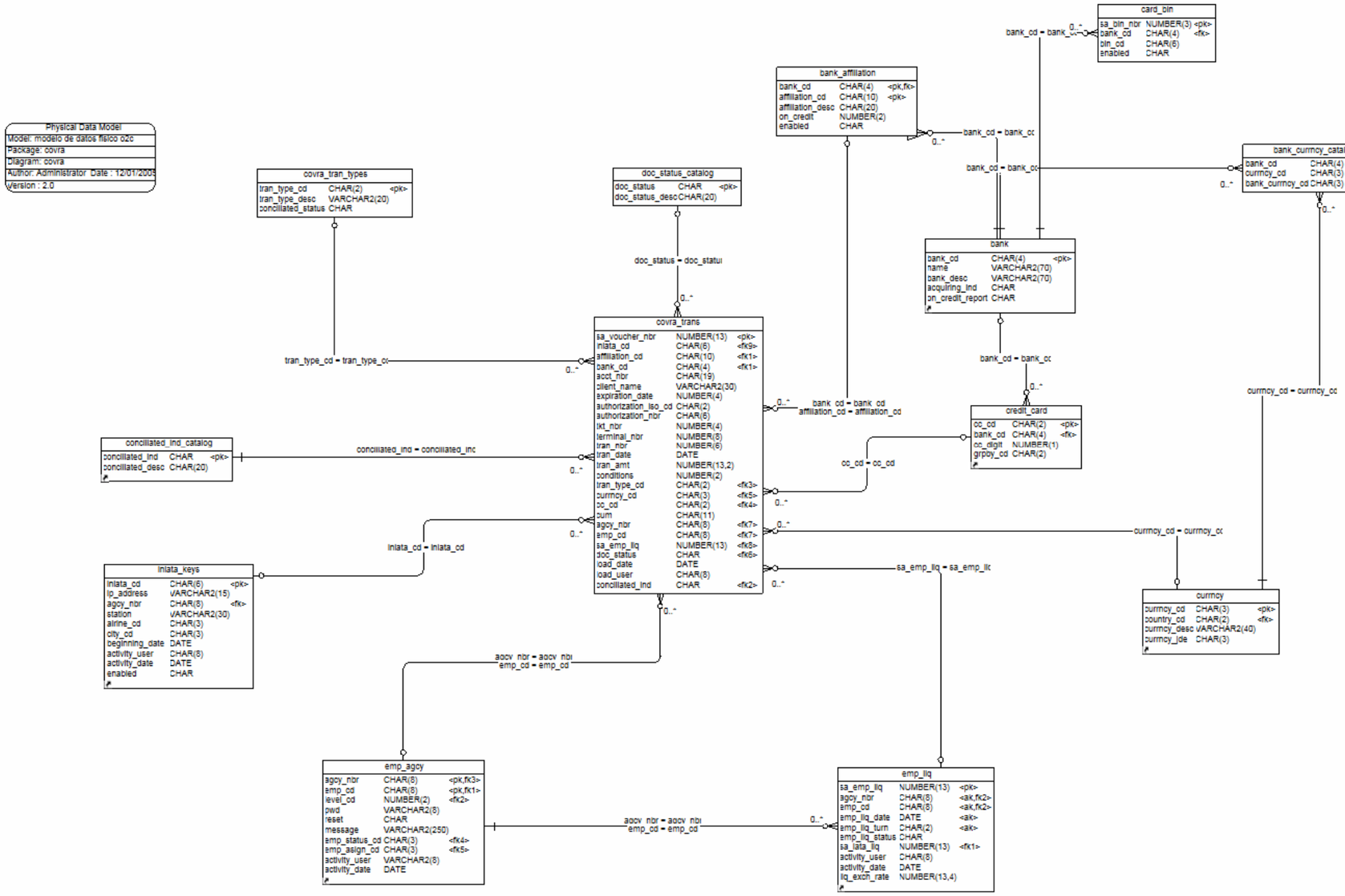
- Existe un proyecto alterno que será la creación y alimentación de la Base de Datos por parte del proyecto EVA Extracción de Ventas Automáticas (esta será la captura de boletos automáticos y transmisión casi online, la fecha de término de este proyecto tiene que estar sincronizada con este proyecto.
- Apoyo de cliente para análisis de procesos.
- Implementar estándares de control de cambios y nuevos requerimientos.
- Validar pruebas integrales, detectar riesgos, problemas actuales y futuros.
- Realizar el cierre de puntos de ventas en el día de operación, con el fin de que sean registrados los movimientos en el mismo día que son contabilizados en la remesa de fondos. Esto para lograr el cuadre de remesa y depósitos bancarios y a su vez con el archivo de movimientos bancarios
- Captura correcta de la información ingresada al nuevo módulo de cierre de venta de oficinas MX
- Definir responsables de cada una de las áreas involucradas que asesoren y den seguimiento al desarrollo de este proyecto
- Presentación de desarrollo y alcances del proyecto así como del procedimiento a difusión normativa y en caso de que aplique sindicatos involucrados
- Difusión de nuevo procedimiento a Oficinas de Venta MX previas a la capacitación
- Capacitación a cajeros de cada una de las oficinas de venta MX y al Centro de Atención Tecnológica previos a la liberación de la aplicación

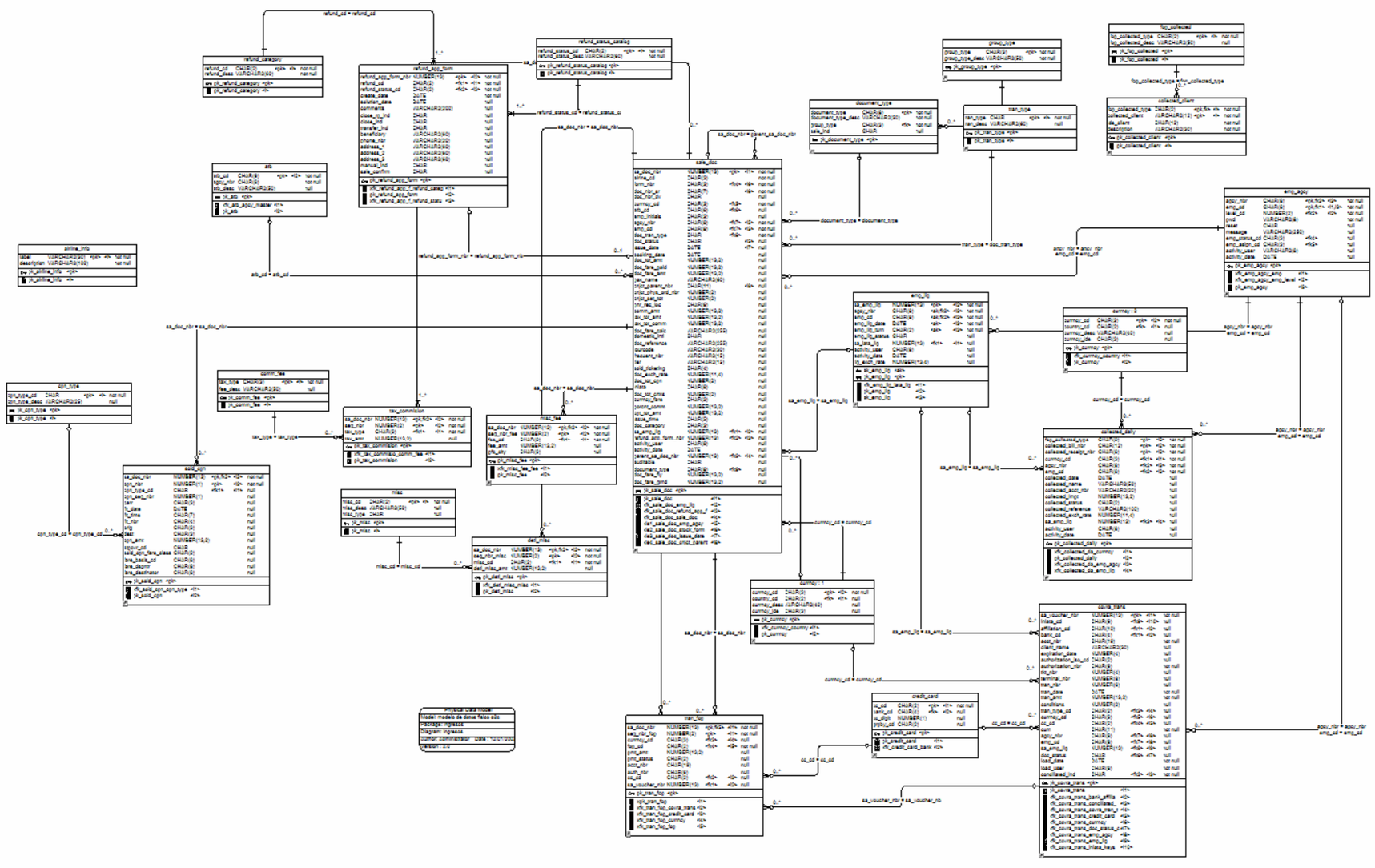
Consideraciones y supuestos

a) Se considera pagos en efectivo todas aquellas percepciones por concepto de vuelo y misceláneos que hayan sido cubiertos por el pasajero en efectivo, cheques personales y de viajero ó bajo el registro de las terminales puntos de venta. Se omite pagos realizados con tarjeta de crédito American Express.
b) Cada tipo de pago maneja un solo tipo de moneda
c) Ninguna oficina de ventas MX puede realizar un reembolso ó cancelación de pagos realizados en punto de venta
d) Existe más de una oficina de venta por ciudad
e) Existen más de 1 cierre de punto de venta al día
f) Existen más de una terminal de punto de venta en cada oficina de ventas MX
g) El cajero capturará detalle de depósitos de remesas de fondos y el cierre de punto de venta requerido para el cuadre de información
h) El número de afiliación de oficina, referencias bancarias y número de remesas son únicas y excluyentes
i) Se genera una remesa de fondos por turno en cada una de las oficinas MX. Y por ende existe más de 1 remesa al día por oficina
j) El tipo de moneda manejado en la oficina de venta MX es únicamente Moneda Nacional y Dólares
k) En caso de falla de sistema, aplicará el procedimiento de contingencia que será estipulado previo a la liberación de la aplicación
l) Eliminar captura en el actual CIE una vez determinada la fecha de paralelo del nuevo módulo de cierre de venta

3. Modelo de datos


Physical Data Model
model: modelo de casa feito ass
Package: diario de representante
Diagram: diario de representante
author: administrador data: 14/11/2022
version : 2.0
currency
currency_id CHAR(3) <pk> <fk> not null
country_id CHAR(2) <fk> <i1> null
currency_desc VARCHAR2(40) null
currency_obs CHAR(30) null
fk (fk_currency) <pk>
pk (pk_currency) <pk>
fk_currency_country <i1>
fk_currency <i2>
sa_doc
sa_doc_nbr NUMBER(12) <pk> <i1> not null
airline_cd CHAR(3) <fk> not null
form_nbr CHAR(3) <fk> <i2> not null
doc_nbr_st CHAR(7) <i2> not null
doc_nbr_dt CHAR null
currency_cd CHAR(3) <fk2> not null
stn_cd CHAR(8) <fk3> null
emd_initials CHAR(3) null
agcy_nbr CHAR(8) <fk7> <i3> not null
emd_cd CHAR(8) <fk7> <i3> not null
doc_tran_type CHAR <i3> not null
doc_status CHAR <i3> null
issue_date DATE <i1> null
booking_date DATE null
doc_tot_amt NUMBER(12,2) null
doc_fare_calc NUMBER(12,2) null
doc_fare_amt NUMBER(12,2) null
pax_name VARCHAR2(50) null
orist_parent_nbr CHAR(11) <i2> null
orist_chk_dig_nbr NUMBER(2) null
orist_iata_tid NUMBER(2) null
chr_iata_loc CHAR(8) null
comm_amt NUMBER(12,2) null
tax_tot_amt NUMBER(12,2) null
tax_tot_comm NUMBER(12,2) null
doc_fare_calc VARCHAR2(255) null
domestic_int CHAR null
doc_reference VARCHAR2(255) null
tourcode VARCHAR2(50) null
frequent_nbr VARCHAR2(12) null
tax VARCHAR2(12) null
addl_ticketing CHAR(3) null
doc_endt_date NUMBER(11,4) null
doc_tot_com NUMBER(2) null
mask CHAR(1) null
doc_tot_crms NUMBER(2) null
currncy_fare CHAR(3) null
comm_currncy NUMBER(12,2) null
cnt_tot_amt NUMBER(12,2) null
issue_time CHAR(5) null
doc_category CHAR(3) null
sa_emp_id NUMBER(12) <fk1> <i2> null
refund_app_form_nbr NUMBER(12) <fk2> <i2> null
activity_user CHAR(8) null
activity_date DATE null
parent_sa_doc_nbr NUMBER(12) <fk3> <i1> null
auditable CHAR null
document_type CHAR(8) <fk8> null
doc_fare_fc NUMBER(12,2) null
doc_fare_grnd NUMBER(12,2) null
fk (fk_sale_doc) <pk>
pk (pk_sale_doc) <pk>
fk_sale_doc <i1>
fk_sale_doc_emp_id <i2>
fk_sale_doc_refund_app_f <i3>
fk_sale_doc_sale_doc <i1>
fk1_sale_doc_emp_agcy <i2>
fk2_sale_doc_stock_form <i3>
fk3_sale_doc_issue_date <i7>
fk4_sale_doc_orist_parent <i8>
sa_doc_nbr = parent_sa_doc_nbr
0..*
currncy_cd = currncy_cd
0..*
sa_doc_nbr = sa_doc_nbr
0..*
agcy_nbr = agcy_nbr
emp_cd = emp_cd
0..*
currncy_cd = currncy_cd
0..*
anpc nbr = anpc nbr
emp_cd = emp_cd
0..*
currncy_cd = currncy_cd
0..*
currncy_cd = currncy_cd
0..*
sa_doc_nbr = sa_doc_nbr
0..*
sa_emp_id = sa_emp_id
0..*
sa_emp_id = sa_emp_id
0..*
sa_emp_id = sa_emp_id

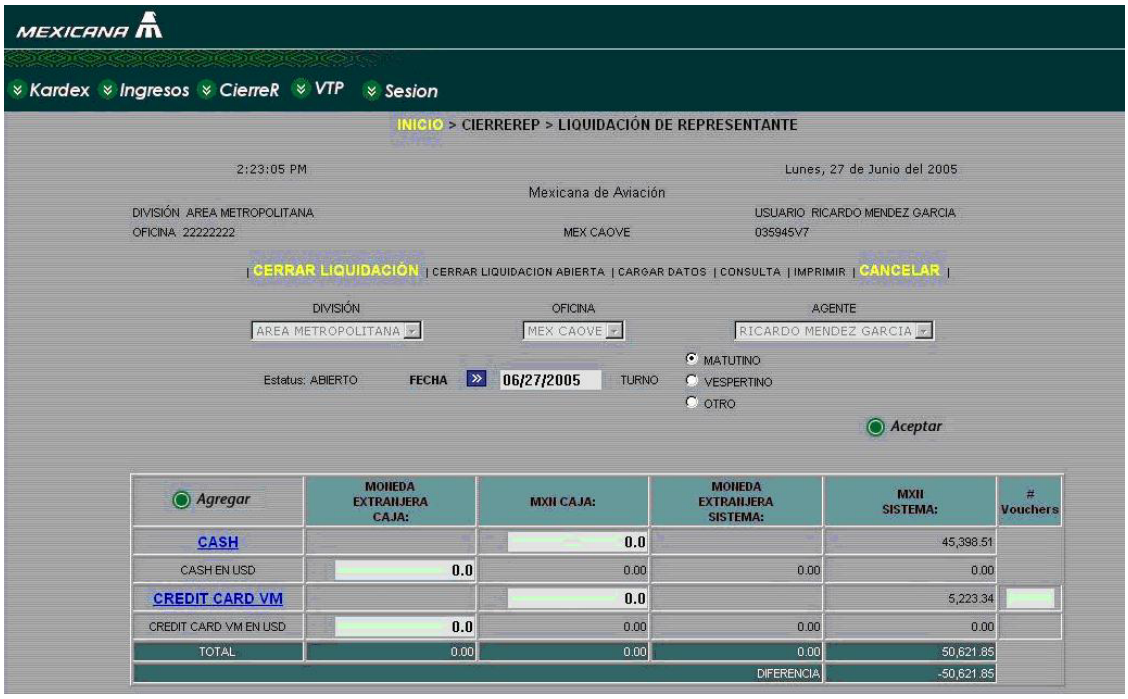

Physical Data Model
Model: modelo de datos fisico o26
Package: covra
Diagram: covra
Author: Administrator Date : 12/01/2005
Version : 2.0
covra_tran_types
tran_type_cd CHAR(2) <pk>
tran_type_desc VARCHAR2(20)
conciliated_status CHAR
doc_status_catalog
doc_status CHAR <pk>
doc_status_desc CHAR(20)
doc_status = doc_statu
0..*
tran_type_cd = tran_type_cc
covra_trans
sa_voucher_nbr NUMBER(13) <pk>
iniata_cd CHAR(6) <fk9>
affiliation_cd CHAR(10) <fk1>
bank_cd CHAR(4) <fk1>
acct_nbr CHAR(19)
client_name VARCHAR2(30)
expiration_date NUMBER(4)
authorization_iso_cd CHAR(2)
authorization_nbr CHAR(6)
ikt_nbr NUMBER(4)
terminal_nbr NUMBER(8)
tran_nbr NUMBER(6)
tran_date DATE
tran_amt NUMBER(13,2)
conditions NUMBER(2)
tran_type_cd CHAR(2) <fk3>
currncy_cd CHAR(3) <fk5>
cc_cd CHAR(2) <fk4>
sum CHAR(11)
agcy_nbr CHAR(8) <fk7>
emp_cd CHAR(8) <fk7>
sa_emp_liq NUMBER(13) <fk8>
doc_status CHAR <fk6>
load_date DATE
load_user CHAR(8)
conciliated_ind CHAR <fk2>
bank_affiliation
bank_cd CHAR(4) <pk,fk>
affiliation_cd CHAR(10) <pk>
affiliation_desc CHAR(20)
on_credit NUMBER(2)
enabled CHAR
bank_cd = bank_cc
card_bin
sa_bin_nbr NUMBER(3) <pk>
bank_cd CHAR(4) <fk>
bin_cd CHAR(6)
enabled CHAR
bank_currncy_catalog
bank_cd CHAR(4) <pk,fk2>
currncy_cd CHAR(3) <pk,fk1>
bank_currncy_cd CHAR(3)
bank
bank_cd CHAR(4) <pk>
name VARCHAR2(70)
bank_desc VARCHAR2(70)
acquiring_ind CHAR
on_credit_report CHAR
bank_cd = bank_cc
bank_cd = bank_cd
affiliation_cd = affiliation_cd
credit_card
cc_cd CHAR(2) <pk>
bank_cd CHAR(4) <fk>
cc_digit NUMBER(1)
grpby_cd CHAR(2)
cc_cd = cc_cd
currncy_cd = currncy_cc
conciliated_ind_catalog
conciliated_ind CHAR <pk>
conciliated_desc CHAR(20)
conciliated_ind = conciliated_inc
iniata_cd = iniata_cd
iniata_keys
iniata_cd CHAR(6) <pk>
ip_address VARCHAR2(15)
agcy_nbr CHAR(8) <fk>
station VARCHAR2(30)
airline_cd CHAR(3)
city_cd CHAR(3)
beginning_date DATE
activity_user CHAR(8)
activity_date DATE
enabled CHAR
sa_emp_liq = sa_emp_liq
currncy_cd = currncy_cc
currncy
currncy_cd CHAR(3) <pk>
country_cd CHAR(2) <fk>
currncy_desc VARCHAR2(40)
currncy_ide CHAR(3)
agcy_nbr = agcy_nbr
emp_cd = emp_cd
emp_agcy
agcy_nbr CHAR(8) <pk,fk3>
emp_cd CHAR(8) <pk,fk1>
level_cd NUMBER(2) <fk2>
pwd VARCHAR2(8)
reset CHAR
message VARCHAR2(250)
emp_status_cd CHAR(3) <fk4>
emp_asign_cd CHAR(3) <fk5>
activity_user VARCHAR2(8)
activity_date DATE
emp_liq
sa_emp_liq NUMBER(13) <pk>
agcy_nbr CHAR(8) <ak,fk2>
emp_cd CHAR(8) <ak,fk2>
emp_liq_date DATE <ak>
emp_liq_turn CHAR(2) <ak>
emp_liq_status CHAR
sa_iata_liq NUMBER(13) <fk1>
activity_user CHAR(8)
activity_date DATE
liq_exch_rate NUMBER(13,4)


refund_cd = refund_cd
refund_category
refund_cd CHAR(3) <pk> <fk> not null
refund_desc VARCHAR2(60) not null
pk_refund_category <pk>
refund_app_form
refund_app_form_nbr NUMBER(12) <pk> <fk> not null
refund_cd CHAR(3) <fk> <fk> not null
refund_status_cd CHAR(2) <fk> <fk> not null
create_date DATE not null
solution_date DATE null
comments VARCHAR2(300) null
close_no_ind CHAR null
close_ind CHAR null
transfer_ind CHAR null
beneficiary VARCHAR2(60) null
chana_nbr VARCHAR2(20) null
address_1 VARCHAR2(60) null
address_2 VARCHAR2(60) null
address_3 VARCHAR2(60) null
manual_ind CHAR null
sale_confirm CHAR null
fk_refund_app_form <pk>
fk_refund_app_f_refund_categ <fk>
fk_refund_app_form <pk>
fk_refund_app_f_refund_statu <fk>
refund_status_catalog
refund_status_cd CHAR(2) <pk> <fk> not null
refund_status_desc VARCHAR2(60) not null
pk_refund_status_catalog <pk>
refund_status_cd = refund_status_cd
sa_doc_nbr = sa_doc_nbr
parent_sa_doc_nbr = parent_sa_doc_nbr
document_type
document_type CHAR(3) <pk> not null
document_type_desc VARCHAR2(60) not null
group_type CHAR(3) <fk> not null
sale_ind CHAR null
pk_document_type <pk>
group_type
group_type CHAR(3) <pk> not null
group_type_desc VARCHAR2(60) not null
pk_group_type <pk>
tran_type
tran_type CHAR(3) <pk> <fk> not null
tran_desc VARCHAR2(60) null
pk_tran_type <pk>
fqp_collected
fqp_collected_type CHAR(2) <pk> <fk> not null
fqp_collected_desc VARCHAR2(30) null
pk_fqp_collected <pk>
fqp_collected_type = fqp_collected_type
collected_client
fqp_collected_type CHAR(2) <pk,fk> <fk> not null
collected_client VARCHAR2(12) <pk> <fk> not null
de_client CHAR(12) not null
description VARCHAR2(60) not null
pk_collected_client <pk>
atb
atb_cd CHAR(8) <pk> <fk> not null
agcy_nbr CHAR(4) not null
atb_desc VARCHAR2(32) null
pk_atb <pk>
fk_atb_agcy_master <fk>
atb_cd = atb_cd
refund_app_form_nbr = refund_app_form_nbr
sale_doc
sa_doc_nbr NUMBER(12) <pk> <fk> not null
airline_cd CHAR(3) not null
form_nbr CHAR(7) <fk> not null
doc_nbr_br CHAR(7) <fk> not null
doc_nbr_ck CHAR null
currncy_cd CHAR(3) <fk> not null
atb_cd CHAR(8) <fk> null
emp_initials CHAR(3) null
agcy_nbr CHAR(4) <fk> <fk> not null
emp_cd CHAR(8) <fk> <fk> not null
tran_type CHAR <fk> not null
doc_status CHAR null
issue_date DATE <fk> not null
booking_date DATE null
doc_tot_amt NUMBER(13,2) null
doc_fare_paid NUMBER(13,2) null
doc_fare_amt NUMBER(13,2) null
pax_name VARCHAR2(49) null
rmbr_parent_nbr CHAR(11) <fk> null
rmbr_chck_and_nbr NUMBER(2) null
rmbr_sel_ind NUMBER(2) null
orc_rec_loc CHAR(6) null
comm_amt NUMBER(13,2) null
tax_tot_amt NUMBER(13,2) null
tax_tot_comm NUMBER(13,2) null
doc_fare_calc VARCHAR2(222) null
domestic_ind CHAR null
doc_reference VARCHAR2(222) null
outside VARCHAR2(30) null
tacsent_nbr VARCHAR2(12) null
tax VARCHAR2(12) null
void_ticketing CHAR(1) null
doc_exch_rate NUMBER(11,4) null
doc_tax_amt NUMBER(2) null
misc CHAR(4) null
doc_tot_cmis NUMBER(2) null
currncy_fare CHAR(3) null
comm_comm NUMBER(13,2) null
tot_tax_amt NUMBER(13,2) null
issue_time CHAR(5) null
doc_category CHAR(3) null
sa_emp_hj NUMBER(12) <fk> <fk> null
refund_app_form_nbr NUMBER(12) <fk> <fk> null
activity_user CHAR(8) null
activity_date DATE null
parent_sa_doc_nbr NUMBER(12) <fk> <fk> null
audittable CHAR null
document_type CHAR(3) <fk> null
doc_fare_hj NUMBER(13,2) null
doc_fare_cmsd NUMBER(13,2) null
pk_sale_doc <pk>
fk_sale_doc
fk_sale_doc_emp_hj <fk>
fk_sale_doc_refund_app_f <fk>
fk_sale_doc_sale_doc <fk>
rel_sale_doc_emp_agcy <fk>
rel_sale_doc_asapt_form <fk>
rel_sale_doc_issue_date <fk>
rel_sale_doc_prior_parent <fk>
document_type = document_type
tran_type = doc_tran_type
agcy_nbr = agcy_nbr
emp_cd = emp_cd
amcp_agcy
agcy_nbr CHAR(4) <pk,fk> <fk> not null
emp_cd CHAR(8) <pk,fk> <fk> not null
level_cd NUMBER(2) <fk> <fk> not null
pwd VARCHAR2(8) null
reset CHAR null
message VARCHAR2(222) null
emp_status_cd CHAR(3) <fk> null
emp_assgn_cd CHAR(3) <fk> null
activity_user VARCHAR2(8) null
activity_date DATE null
pk_amcp_agcy <pk>
fk_amcp_agcy_amcp <fk>
fk_amcp_agcy_amcp_level <fk>
fk_amcp_agcy <fk>
airline_info
label VARCHAR2(20) <pk> <fk> not null
description VARCHAR2(100) not null
pk_airline_info <pk>
sa_doc_nbr = sa_doc_nbr
comm_fee
fee_type CHAR(3) <pk> <fk> not null
fee_desc VARCHAR2(32) null
pk_comm_fee <pk>
tax_commission
sa_doc_nbr NUMBER(12) <pk,fk> <fk> not null
seq_nbr NUMBER(2) <pk> <fk> not null
tax_type CHAR(3) <fk> <fk> not null
tax_amt NUMBER(13,2) null
pk_tax_commission <pk>
fk_tax_commissio_comm_fee <fk>
fk_tax_commission <fk>
tax_type = tax_type
misc_fee
sa_doc_nbr NUMBER(12) <pk,fk> <fk> not null
seq_nbr_fee NUMBER(2) <pk> <fk> not null
fee_cd CHAR(3) <fk> <fk> not null
fee_amt NUMBER(13,2) null
fm_city CHAR(3) null
pk_misc_fee <pk>
fk_misc_fee_fee <fk>
fk_misc_fee <fk>
misc
misc_cd CHAR(2) <pk> <fk> not null
misc_desc VARCHAR2(32) null
misc_type CHAR null
pk_misc <pk>
fk_misc <fk>
misc_cd = misc_cd
def_misc
sa_doc_nbr NUMBER(12) <pk,fk> <fk> not null
seq_nbr_misc NUMBER(2) <pk> <fk> not null
misc_cd CHAR(2) <fk> <fk> not null
def_misc_amt NUMBER(13,2) null
pk_def_misc <pk>
fk_def_misc_misc <fk>
fk_def_misc <fk>
sa_doc_nbr = sa_doc_nbr
sa_doc_nbr = sa_doc_nbr
sa_doc_nbr = sa_doc_nbr

4. Prototipo de pantallas

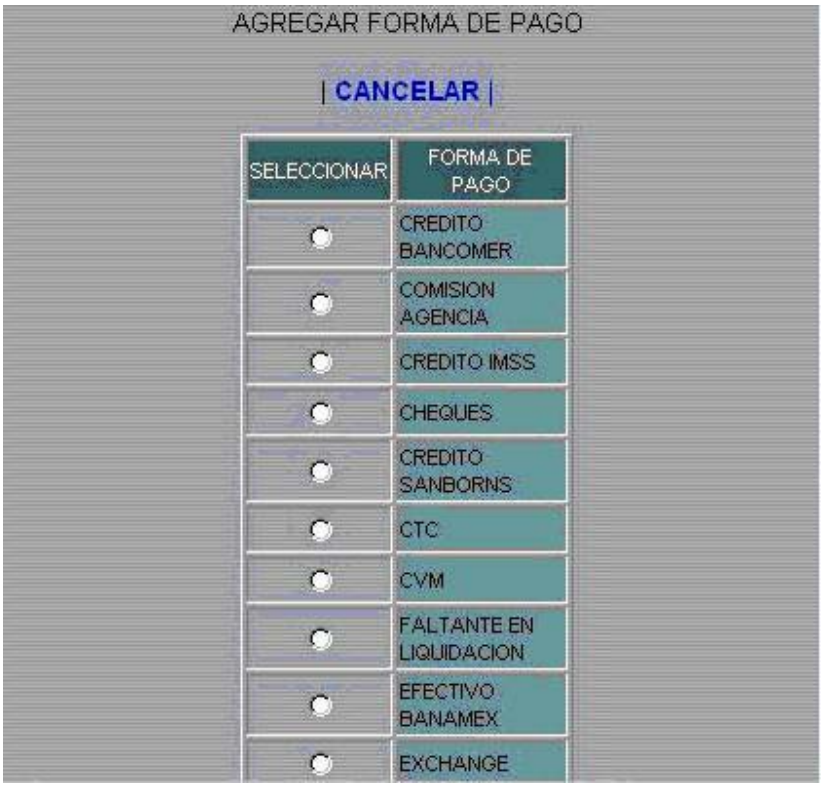
4.1 Principal

4.2 Carga de datos

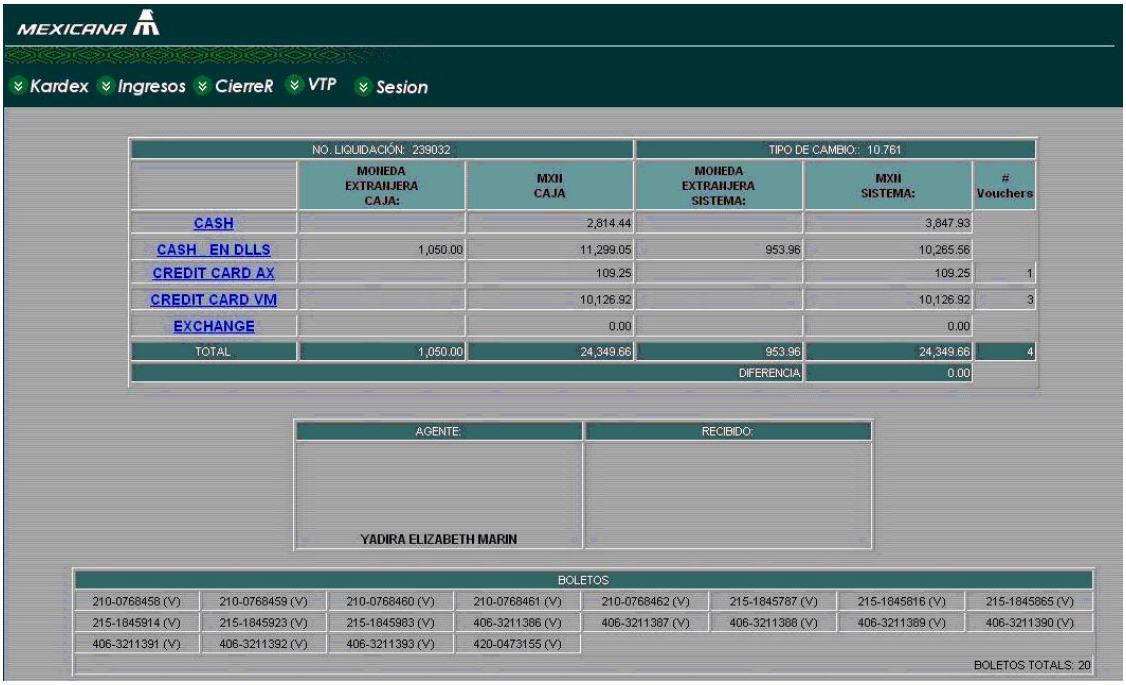


4.3 Diferencia positiva de dólares

4.4 Agregar forma de pago



4.5 Consulta



4.6 Boletos por forma de pago

5. Herramientas de software para el cálculo de métricas

- **COSMOS** (Software Cost Modeling System- Sistema de modelado del costo del software)

  Facilita la tarea de la estimación de proyectos de desarrollo de software permitiendo estimar el tamaño y el esfuerzo utilizando los modelos de Function Points, COCOMO y Rayleigh.

- **SLIM:** Sistema de base de datos para el análisis de la productividad.

  La entrada principal para SLIM son las líneas de código, FP's, objetos, CSCI o cualquier medición válida o función a ser creada. El modelo utiliza los rangos de tamaño de entrada : mínimo, deseado y máximo. Otras entradas importantes son:

  - Lenguaje de programación: varias opciones o mezclas.
  - Tipo de sistema: negocio, científico, tiempo real, de control, etc.
  - Información de ambiente: herramientas, métodos, prácticas, uso de la base de datos, estándares.
  - Experiencia: habilidades y capacidades del personal
  - Parámetro de la productividad: un factor macroscópico determinado a partir de la calibración de datos históricos. Se refiere a un factor de ajuste confiable que refleja con precisión la complejidad de la aplicación y la eficiencia de la organización en la construcción de software. Esto es un parámetro sensitivo capaz de medir la productividad real y la mejora del proceso. SLIM contiene un sistema experto para determinar el parámetro de la productividad del proceso cuando el usuario no cuenta con datos históricos.
  - Restricciones de administración: fecha límite de entrega, costo máxima, mínimo y máximo número de miembros del equipo, confiabilidad requerida en el momento en que el sistema entra en producción así como las probabilidades deseadas de cada restricción.
  - Contabilidad de costos: factores económicos como inflación, tarifas, etc.

- Flexibilidad: definición de fases, tiempo y esfuerzo aplicado a cada fase, basados en la propia historia de la organización.

- **ACEIT** (Automated Cost Estimating Integrated Tools – Herramientas Integradas de Estimación de Costos Automatizadas)

  Es una familia de aplicaciones que dan soporte al administrador del proyecto y a los analistas de costos y finanzas durante todas las fases del ciclo de vida del proyecto. Las aplicaciones ACEIT se utilizan para analizar, desarrollar, compartir y generar reportes de estimación de costos, a través de una suite que automatiza tareas clave de análisis y simplifica o estandariza el proceso de estimación.

- **Charismatek**: herramienta que proporciona soporte total para realizar el análisis de Function Points definidos por la IFPUG en su Manual de prácticas de conteo versiones 4.0, 4.1 y 4.2.

- **COCOPro** : basado bajo el método del modelo constructivo de costos de Barry Bohem, CoCoPro estima los recursos necesarios para completar un proyecto de desarrollo de software. El programa utiliza funciones exponenciales y atributos para calcular los costos de desarrollo.

- **Cost Xpert** : esta herramienta de estimación de costos de software se calibra para reflejar los últimos estándares de la industria, genera estimaciones de costos y tiempo y automáticamente genera una estructura de trabajo en arbol para utilizarla como un punto de partida para el plan de proyecto. También es ajustable a los procesos de la organización.

- **Galorath**: Empresa que ofrece 3 soluciones para la estimación del software, la planeación y el control del proyecto: SEER-SEM, SEER-SEM Client, SEER-AccuScope & SEER-SSM.

  El software SEER proporciona herramientas que brindan soporte para la toma de decisiones y la optimización de los procesos para estimar costos, trabajo, calendarización, confiabilidad y riesgos asociados con proyectos de tecnología de la información, sistemas embebidos y desarrollo de software comercial.

GLOSARIO DE TÉRMINOS

AFP	Adjusted Function Point Se refiere al valor ajustado del conteo de function points.
Aplicación	Conjunto de componentes que interactúan entre si para automatizar un proceso
Aplicación Batch	Se refiere a una aplicación que no tiene interacción con un usuario.
Benchmark	Practica que consiste en comparar una empresa, producto, servicio etc. contra el mejor en el mercado.
Bottom Up	Método de estimación que analiza desde los componentes más específicos hacia los componentes más generales.
CCV	Costo del Ciclo de Vida de un proyecto de software.
CEO	Chief Executive Officer, gerente ejecutivo general .
CIO	Chief Info Officer, gerente de TI.
COCOMO	Constructive Cost Model.
COO	Chief Operating Officer, gerente de operaciones.
Dato	Colección de hechos o partes de información que son mantenidos dentro de la aplicación.
Dato derivado	Surgen de una transformación de datos existentes para crear datos adicionales.

DET	Data Element Type, campo o atributo único reconocido por el usuario.
DI	Degree of Influence, contribución de una característica general al cálculo del valor de ajuste.
ECS	Estimación del Costo del Software.
EDS	Electronic Data Systems, segunda empresa más grande del mundo en la consultaría de TI.
EI	External Input, entrada externa, son datos o información de control que cruzan la frontera de la aplicación de afuera hacia dentro y que mantienen un ILF/EIF.
EIF	External Interface File, archivo de interfaz externa, es un conjunto de datos lógicos que se encuentran fuera de la frontera de la aplicación.
EO	External Output, salida externa, son datos o información de control que cruza la frontera de la aplicación de adentro hacia fuera, tiene datos derivados y/o mantiene un ILF/EIF.
EQ	External Inquiry, petición externa, datos o información de control que cruzan la frontera de adentro hacia fuera que no contienen datos derivados no mantienen ningún ILF/EIF.
FP	Function Point, puntos de función, es la métrica utilizada para dimensionar el tamaño de un software en la medida de su funcionalidad.
FPA	Function Point Analysis, conjunto de reglas para determinar el tamaño de una aplicación en FP's.

Frontera de la Aplicación	Es el límite entre la aplicación que se está midiendo y otras aplicaciones independientes.
FTR	File Type Referenced, archivos referenciados: se refiere al número total de ILF's mantenidos o leídos y EIF's leídos por un proceso elemental.
GSC	General System Characteristics, características generales del sistema, se refiere a los requerimientos no funcionales del sistema
GUI	Graphical User Interface, interfaz gráfica de usuario, son los components del sistema con los que interactua el usuario final.
IEEE	Institute of Electrical and Electronics Engineers, instituto internacional cuyo objetivo principal el promover el avance de las teorías y las practicas de la electro-tecnología
IFPUG	International Fuction Points Users Group, grupo intgernacional de usuarios de FP's, organización encargada de regular y emitir las reglas para el conteo de FP's.
ILF	Internal Logical Files, archivos lógicos internos, conjuntos de datos lógicos que son mantenidos dentro de la aplicación.
Información de control	Son datos usados por la aplicación que está siendo medida para determinar el flujo de un proceso elemental.
ISBSG	International Software Benchmarking Standards Group, es una organización internacional sin fines de lucro establecida en 1997 que se encarga de crear y promover estándares comparativos del software

ISO:	International Standard Organization.
LOC	Lines of Code, es la métrica utilizada para dimensionar el tamaño de un software basándose en el número de líneas de código.
Lógica de procesamiento	Se refiere a las reglas de negocio que deben respetar los procesos elementales.
Mantener	Capacidad de modificar datos a través de un proceso elemental.
Metodología	Conjunto de reglas o pasos estandarizados para alcanzar un fin.
Métrica	Estándar con el cual un proceso o producto puede ser evaluado objetivamente.
Aplicación on – line	Aplicación que actualiza la información el momento en que el usuario interactúa con el.
Proceso Elemental	Unidad de trabajo más pequeña que tiene sentido para el usuario.
RET	Record Element Type, tipo de elementos de registro, subgrupos de elementos de datos (opcionales u obligatorios), reconocibles por el usuario contenidos dentro de un ILF o EIF.
SEI	Centro Federal de Investigación cuya misión es impulsar la práctica de la ingeniería del software para mejorar la calidad de los sistemas
TDI	Total Degree of Influence, grado total de influencia, sumas de las contribuciones de todas las características generales del sistema para determinar el valor de ajuste de FP's.

Top Down	Método de estimación que analiza desde los componentes más generales hacia los componentes más especificos.
UFP	Unadjusted Function Point, conteo de puntos de función sin ajustar.
Usuario	Actor de negocio que interactua con la aplicación.
VAF	Value Adjustment Factor, valor del factor de ajuste del conteo de FP.
Web	Término coloquial utilizado para referirse a la World Wide Web.

Bibliografía

Function Point Analysis. Measurement Practices for Successful Software Projects
David Garmus, David Herron
Addison – Wesley
Estados Unidos, 2004
4a Edición

IT Measurement. Practical Advice from the Experts
International Function Points Users Group
Addison – Wesley
Estados Unidos, 2004
1ª Edición

Parametric Estimating Handbook
Departamento de Defensa de los Estados Unidos
Estados Unidos, 1999
2ª Edición

La Necesidad de la Planeación Estratégica
Conrado Aguilar Cruz

Introducción a la Computación
Peter Norton
McGraw – Hill
México, 2000

En Internet

International Function Point Users Group:
www.ifpug.org

International Software Benchmarking Standards Group:
www.isbsg.org.au

Longstreet Consulting Inc:
www.softwaremetrics.com

NASA:
www.jsc.nasa.gov/bu2/PCEHHTML/pceh.html

Revista Fortune :
www.fortune.com

www.aceit.com/Pages/Products/ProductsPage.aspx

www.charismatek.com.au

www.galorath.com/tools_soft.html

www.totalmetrics.com

portal.acm.org