UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CIENCIAS

MATERIAL DE APOYO PARA EL CURSO MÉTRICAS DE SOFTWARE

ACTIVIDAD DE APOYO A LA DOCENCIA

QUE PARA OBTENER EL TÍTULO DE:

LICENCIADO EN CIENCIAS DE LA COMPUTACIÓN

P R E S E N T A:

JESÚS IVÁN SAAVEDRA MARTÍNEZ

DIRECTOR DE TESIS:
DR. FRANCISCO VALDÉS SOUTO

Ciudad Universitaria, Cd. Mx., 2017

1. Datos del alumno
Saavedra
Martínez
Jesús Iván
55 27 56 26 50
Universidad Nacional Autónoma de México
Facultad de Ciencias
Ciencias de la Computación
309203094

2. Datos del tutor
Dr
Francisco
Valdés
Souto

3. Datos del sinodal 1
Dra
Hanna
Jadwiga
Oktaba

4. Datos del sinodal 2
M en C
María Guadalupe Elena
Ibargüengoitia
González

5. Datos del sinodal 3
Dr
Morales
Trujillo
Miguel Ehécatl

6. Datos de sinodal 4
Dra
Selene Marisol
Martínez
Ramírez

7. Datos del trabajo escrito
Material de apoyo para el curso Métricas de Software
Medición y estimación de software
180 p
2017

Índice

Introducción

Motivación

El programa de estudios de Ciencias de la Computación de la Facultad de Ciencias de la UNAM contiene la asignatura Métricas de Software, de la cual actualmente no contamos con suficientes libros en la biblioteca "Ricardo Monges López" (Facultad de Ciencias) que abarquen los temas vigentes de medición y estimación de software.

En la última consulta realizada en la biblioteca en septiembre del 2017, al buscar en las computadoras “Métricas de Software”, “Software Metrics”, “Medición de Software”, “Software Measurement” o “Estimación de Software” no arroja ningún resultado, la consulta de “Software Estimation” dio como único resultado el libro de Alain Abran “Software Project Estimation: The Fundamentals for Providing High Quality Information to Decision Makers” que abarca temas de estimación de software. Al extender la consulta a la Biblioteca Central, se encontraron algunos resultados como “Software measurement and estimation : a practical approach” de Laird y Brennan, “Applied software measurement : assuring productivity and quality” de Capers Jones y “Software estimation best practices, tools & techniques : a complete guide for software project estimators” de Chemuturi. Sin embargo, el contenido de estos libros no cubren por completo los temas que se desarrollan en este documento.

Actualmente la asignatura de Métricas de Software tiene un temario en el que algunas de las métricas y métodos de estimación que se abordan ya no son vigentes, ya que en los últimos años se han desarrollado y utilizado métricas y métodos de estimación que no presenta el temario.

Es necesario que los estudiantes de la carrera de Ciencias de la Computación tengan conocimientos generales de las métricas que acompañan a la Ingeniería de Software para poder llevar a cabo de manera exitosa un proyecto de software. Para ello, los estudiantes deben contar con material de apoyo suficiente para adquirir y poner en práctica dichos conocimientos. El propósito de este trabajo es proporcionar a los estudiantes dicho material de apoyo.

Objetivo

El objetivo general de este trabajo es proporcionar material para el curso de Métricas de Software con temas vigentes, que a su vez sirva como apoyo de consulta tanto a los alumnos que tomen el curso como a los profesores y ayudantes que lo impartan.

El objetivo específico es aportar a los estudiantes los conocimientos generales de las métricas que acompañan a la Ingeniería de Software, haciendo énfasis en las métricas que requieren del tamaño de software, así como resolver problemas reales del entorno a través de la investigación, definición y explicación las distintas metodologías y definiciones de métricas de software existentes y qué tipos de métricas son utilizadas actualmente en la industria de software.

Contenido

Este trabajo consiste en un material de consulta general respecto de las métricas que van relacionadas con la Ingeniería de Software, como lo son:

- los conceptos clave para llevar a cabo el desarrollo de medidas de software,
- las mediciones de software más comunes en la industria,
- el proceso de estimación de un proyecto,
- los modelos de estimación utilizados en el desarrollo y
- algunas técnicas para llevar a cabo el seguimiento (monitoreo) del proyecto mediante metodologías de gestión de proyectos de software.

Estos temas son los que se abarcan hoy en el curso de Métricas de Software que se ha impartido desde 2015, este trabajo consiste de dos partes, la primera es el presente documento donde se desarrolla cada tema a detalle, la segunda en una serie de presentaciones en diapositivas para la clase, donde de manera resumida y didáctica se presentan los temas del curso.

Al ser Métricas de Software una asignatura optativa de séptimo u octavo semestre, el alumno que la cursa debió haber aprobado previamente la materia de Ingeniería de Software de sexto semestre. Sin embargo, en el primer capítulo de este documento se da una breve introducción a la Ingeniería de Software con el objetivo de retomar los fundamentos que de ella son indispensables para comprender de mejor manera las Métricas de Software, en el segundo capítulo se da una introducción a las Métricas de Software, abarcando los fundamentos y la definición de los conceptos básicos de éstas y describiendo los aspectos que se pueden medir en el software y algunos tipos de medidas utilizadas en la práctica.

En el tercer capítulo se presenta la descripción de algunos estándares de medición haciendo énfasis en los métodos de medición de tamaño funcional donde su resultado puede ser utilizado, por ejemplo, como entrada a un modelo de estimación. El cuarto capítulo presenta una alternativa a la medición de tamaño funcional, la aproximación que permite calcular un tamaño funcional de manera rápida en etapas tempranas del ciclo de vida de un proyecto o, en general, cuando no se tiene la suficiente información para realizar una medición.

El quinto capítulo trata sobre la estimación de proyectos de software, una vez que se ha descrito para qué nos sirven las Métricas de Software y como aplicar un método de medición podemos realizar estimaciones de proyectos de software confiables. Cuando estimamos un proyecto (esfuerzo, tiempo y/o costo) se debe llevar a cabo el seguimiento (monitoreo) de éste mediante metodologías de gestión de proyectos de software, descritas en el capítulo sexto, con el fin de poder llevar a cabo de manera exitosa el proyecto.

Con base en este documento se desarrolló material didáctico que consiste en una serie de diapositivas que abarcan los temas abordados en el curso, con el propósito de tener material necesario para impartir un curso de Métricas de Software. La forma en la que se debe utilizar el material didáctico se encuentra en el Anexo C – Guía de uso de material didáctico.

Las estrategias de enseñanza, además de las diapositivas que sirven para exponer los temas, incluyen una serie de ejemplos y ejercicios propuestos para poner en práctica lo aprendido e investigar más sobre cada tema.

Introducción a la Ingeniería de Software

El objetivo de este capítulo es retomar los fundamentos de la Ingeniería de Software que son indispensables para comprender de mejor manera las Métricas de Software, por que utilizar métricas y para que nos sirven estas métricas (capítulo 2).

Este capítulo termina con una serie de ejercicios propuestos para poner en práctica lo aprendido e investigar más sobre el tema.

La información que se presenta fue recopilada principalmente de la guía del Cuerpo de Conocimiento de la Ingeniería de Software (Software Engineering Body of Knowledge – SWEBOK) [1] y del libro “Ingeniería del Software. Un enfoque desde la guía SWEBOK” [2], las definiciones de conceptos fueron recopiladas principalmente del “Glosario de Terminología de Ingeniería de Software” [3].

1.1. Fundamentos de Ingeniería de Software

1.1.1. ¿Arte o ingeniería?

En 1974 la conferencia que impartió el profesor Donald Knuth de la universidad de Stanford con motivo de la recepción del premio Turing comenzó de la siguiente manera [2]:

> *"Si la programación de computadoras quiere llegar a ser una parte importante del desarrollo e investigación en las ciencias de la computación, deberá transitar desde la programación como arte a la programación como ciencia disciplinada…"*

Después de tratar diferentes aspectos de los términos de ciencia y arte, la conferencia concluyó con lo siguiente:

> *"La programación es un arte, por que aplica conocimiento acumulado, requiere habilidades e ingenio, y especialmente por que produce objetos bellos…"*

La Ingeniería de Software es hoy en día una actividad de trabajo en grupo y no una actividad que se desarrolla de manera individual, un ingeniero de software que se enfrenta a un proyecto de desarrollo trabaja sobre un marco de restricciones económicas, de plazos, costos y calidad. Por lo tanto, las acciones y decisiones de esta ingeniería provienen de la aplicación de métodos y técnicas para organizar los recursos de acuerdo con planes y objetivos definidos.

Por lo anterior, se debería de tratar al desarrollo de software como ciencia disciplinada, sometida al estudio científico y basada en técnicas y métodos aceptados por las personas que se dedican al desarrollo de software.

1.1.2. ¿Qué es la ingeniería y que es el software?

El Instituto de Ingeniería Eléctrica y Electrónica (Institute of Electrical and Electronics Engineers – IEEE) define la ingeniería de la siguiente manera:

> *Definición.* La **ingeniería** [3] es la aplicación de un enfoque sistemático, disciplinado y cuantificable de estructuras, máquinas, productos, sistemas o procesos.

La ingeniería tal como hoy la entendemos, siempre resulta en algún artefacto concreto, que puede ser de uso final o como un elemento para ser utilizado en otros procesos de ingeniería. En cuanto a la Ingeniería de Software, su resultado útil son las aplicaciones, las cuales los usuarios utilizan para hacer más eficaz, controlado o eficiente su trabajo, sin embargo, es importante notar que para llegar al resultado final de la Ingeniería de Software, se necesitan varios artefactos concretos intermedios que son reutilizados para poder obtener como resultado las aplicaciones.

Las diferentes ramas o disciplinas de la ingeniería difieren en el objetivo de la producción, pero todas ellas tienen en común tres aspectos específicos [2]:

- La *ciencia de la ingeniería*, que se ocupa de los principios y mecanismos subyacentes de la disciplina.
- Procesos de *diseño*, que en general incluyen una fase de conceptualización y una fase de diseño detallado.
- Aspectos de *gestión y organización*, pues la tecnología que se produce implica tanto a las personas como a las organizaciones.

La ciencia de la ingeniería que nos interesa cuando estudiamos a la Ingeniería de Software es la ciencia de la computación, pues abarca los principios matemáticos de proyectos que involucren programación, y diseño y análisis de sistemas complejos para la automatización de muy diversas actividades [4].

La idea de software en los años 50 era prácticamente el sinónimo del término *programa de computadora*. Sin embargo, el concepto de software va más allá de los programas de computadoras.

> *Definición.* El **software** [3] es el conjunto de los programas de cómputo, procedimientos, reglas, documentación y datos asociados que forman parte de las operaciones de un sistema de computación.

Para poder comprender lo que es el software (y la Ingeniería de Software), es importante examinar las características que lo hacen diferente de otras cosas que el hombre puede construir. Cuando se construye hardware, el proceso se traduce finalmente en una forma física. El software tiene las siguientes características distintas a las del hardware de acuerdo a Pressman [5]:

- El software se desarrolla, no se fabrica.
- El software no se desgasta.
- La mayor parte del software se construye a la medida.

Por lo tanto, software no son sólo programas de computadora en sí, sino también los documentos que los describen, así como cualquier otro artefacto relacionado con el mismo, como los procedimientos para su instalación o modificación, e incluso los datos necesarios para su operación.

1.1.3. ¿Qué es la Ingeniería de Software?

La Ingeniería de Software como ciencia es la aplicación del método científico a la teorización y creación de conocimiento sobre la propia Ingeniería de Software. Está dedicada al estudio de sus actividades, y centrada en generar teorías, modelos explicativos o enunciados descriptivos sobre la práctica de la ingeniería. La IEEE define la Ingeniería de Software como:

> *Definición.* La **Ingeniería de Software** [3] es la aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento de software; es decir, la aplicación de la ingeniería al software.

En esta definición se mencionan tres calificativos que pueden aplicarse a la ingeniería, que son sistematicidad, disciplina y cuantificación, de acuerdo al diccionario Macmillan [6]:

- Decimos que algo es **sistemático** cuando sigue completamente un método.
- Decimos que algo es **disciplinado** cuando está bien organizado y sigue reglas o estándares.
- Decimos que algo es **cuantificable** si es capaz de ser medido o descrito como una cantidad.

La Ingeniería de Software aplica diferentes normas y métodos que permiten obtener mejores resultados, en cuanto al desarrollo y uso del software, mediante la aplicación correcta de estas normas y métodos se puede llegar a cumplir de manera satisfactoria con los objetivos fundamentales de la Ingeniería de Software. Entre los objetivos de la Ingeniería de software están [2]:

- Mejorar el diseño de aplicaciones o software de tal modo que se adapten de mejor manera a las necesidades de las organizaciones o finalidades para las cuales fueron creadas.
- Promover mayor calidad al desarrollar aplicaciones complejas.
- Brindar mayor exactitud en los costos de proyectos y tiempo de desarrollo de los mismos.
- Aumentar la eficiencia de los sistemas al introducir procesos que permitan medir mediante normas específicas, la calidad del software desarrollado, buscando siempre la mejor calidad posible según las necesidades y resultados que se quieren generar.
- Una mejor organización de equipos de trabajo, en el área de desarrollo y mantenimiento de software.
- Detectar a través de pruebas, posibles mejoras para un mejor funcionamiento del software desarrollado

1.1.4. Conceptos básicos de la Ingeniería de Software

Como toda ingeniería, la Ingeniería de Software trata fundamentalmente de actividades llevadas a cabo por personas que producen, usan o modifican artefactos. Esas actividades responden a planes parciales o totalmente prescritos (sistemáticas y disciplinadas).

Definición. Una **actividad** [7] es un conjunto cohesivo de tareas de un proceso.

Por lo tanto, las actividades en la Ingeniería de Software abarcan cualquier acción con un propósito claro dentro de esta ingeniería, lo que incluye actividades de gestión, producción, comunicación y documentación.

Las actividades en general tienen asociadas una serie de prescripciones, es decir, están sujetos a normas que dictan cómo deben hacerse. Estas normas provienen del conocimiento científico, o bien de la experiencia. El término *método* es uno de los más utilizados para referirse a ellas. Según la guía SWEBOK [1], los métodos imponen estructura a la actividad de Ingeniería de Software con el objetivo de hacerla más sistemática y finalmente más exitosa.

> *Definición.* Un **método** [8] describe las características de un proceso o procedimiento ordenado utilizado en la ingeniería de un producto o la elaboración de un servicio.

En la Ingeniería de Software, los métodos determinan el orden y la forma de llevar a cabo las actividades. El término *metodología* hace referencia al estudio de los métodos. También puede utilizarse para hacer referencia a un conjunto coherente de métodos.

> *Definición.* Se denomina **metodología** [7] a un conjunto de prácticas, procedimientos, técnicas y reglas utilizados en la especificación del proceso a seguir.

Las especificaciones, como elemento de información, son una parte fundamental de toda disciplina de ingeniería.

> *Definición.* Una **especificación** [3] es una descripción de manera completa, precisa y verificable los requerimientos, el diseño, el comportamiento u otras características de un sistema o componente y, a menudo, los procedimientos para determinar si se han cumplido estas descripciones.

Un concepto muy utilizado es el *ciclo de vida del software*. La utilidad de este concepto es resaltar que el ciclo de vida no se restringe a las actividades de desarrollo previas al uso del software, sino que también abarcan su evolución y mantenimiento.

> *Definición.* El **ciclo de vida del software** [8] es la evolución del proyecto de software desde el momento de su creación hasta el momento en que deja de usarse, y puede describirse en función de las actividades que se realizan dentro de él.

Todo proyecto de software tiene como objetivo producir software de la mejor calidad posible, que cumpla, y si puede supere las expectativas de los usuarios.

> *Definición.* La **calidad de software** [9] se define como el grado con el que un sistema, componente o proceso cumple con los requerimientos especificados y las necesidades o expectativas del cliente o usuario.

Un término relacionado que también se usa de manera frecuente es el de *proceso*, que no es otra cosa que una secuencia de actividades que comparten un propósito.

> *Definición.* Un **proceso** [3] es una secuencia de pasos llevados a cabo para un propósito específico.

Un proceso es un conjunto de actividades y resultados asociados que producen un producto de software. Diferentes tipos de sistemas necesitan diferentes procesos de desarrollo.

> *Definición.* Un **procedimiento** [3] es un camino a seguir para llevar a cabo una tarea determinada.

Un procedimiento está formado por pasos que tienen la capacidad de ser determinantes, esto supone que los mismos valores de entrada producirán siempre los mismos valores de salida.

1.2. Procesos Fundamentales de la Ingeniería de Software

1.2.1. Requerimientos

Uno de los problemas a los que se enfrenta la Ingeniería de Software es la dificultad para determinar cuáles son los requerimientos de un sistema. Es decir, las funcionalidades que el sistema a construir debe incluir, así como todas las consideraciones adicionales sobre seguridad, rendimiento, restricciones, etc.

Las actividades de la Ingeniería de Software relacionadas con la obtención y gestión de requerimientos[1], consisten en obtener y documentar los requerimientos para desarrollar el sistema.

> *Definición.* Un **requerimiento de software** [3] es la capacidad que debe alcanzar o poseer un sistema o componente de un sistema para satisfacer un contrato, estándar, especificación u otro documento formal.

Cuando se especifican los requerimientos, se almacena información sobre los mismos que permite gestionarlos e interpretarlos.

El conjunto de tareas a realizar en el área de los requerimientos software consisten en [1]:

- Obtención de requerimientos: consiste en capturar el propósito y funcionalidades del sistema desde la perspectiva del usuario.
- Análisis de requerimientos: es el proceso de estudiar las necesidades del usuario para obtener una definición detallada de los requerimientos.
- Especificación de requerimientos: proceso de documentar el comportamiento requerido de un sistema software, a menudo utilizando una notación de modelado u otro lenguaje de especificación
- Validación de requerimientos software: consiste en examinar los requerimientos para asegurarse de que definen el sistema que el cliente y los usuarios desean.

Tradicionalmente se han identificado dos tipos de requerimientos atendiendo a criterios de funcionalidad: *requerimientos funcionales* y *requerimientos no funcionales* [10].

> *Definición*. Los **requerimientos funcionales** [3] son requerimientos que especifican una función que un sistema o componente del sistema debe ser capaz de realizar.

Los requerimientos funcionales incluyen, pero no están limitados a [11]:

- Transferencia de datos.
- Transformación de datos.
- Almacenamiento de datos.
- Recuperación de datos.

[1] Comúnmente, se utiliza la palabra requerimientos, sin embargo la palabra correcta es requisitos, ya que un requerimiento es una solicitud que hace una autoridad, y un requisito es una característica que se debe cumplir.

Definición. Un **requerimiento no funcional** [7] es un requerimiento de software que describe no lo que hará el software, sino cómo lo hará el software.

El estándar IEEE 830 [10] lista 13 requerimientos no funcionales que deben ser incluidos en un Documento de Requerimientos de Software:

- Requerimientos de desempeño.
- Requerimientos de interfaz.
- Requerimientos operacionales.
- Requerimientos de recursos.
- Requerimientos de verificación.
- Requerimientos de aceptación.
- Requerimientos de documentación.
- Requerimientos de seguridad.
- Requerimientos de portabilidad.
- Requerimientos de calidad.
- Requerimientos de confiabilidad.
- Requerimientos de mantenimiento.
- Requerimientos de prevención.

De acuerdo al Consorcio Internacional para la Medición Común de Software (Common Software Measurement International Consortium – COSMIC) [11] , puede darse el caso que algunos requerimientos del sistema o de software que se expresan inicialmente como no funcionales evolucionan a medida que el proyecto avanza en una mezcla de requerimientos que pueden ser implementados en las funcionalidades del software y otros requerimientos que son verdaderamente no funcionales (Figura 0.1).

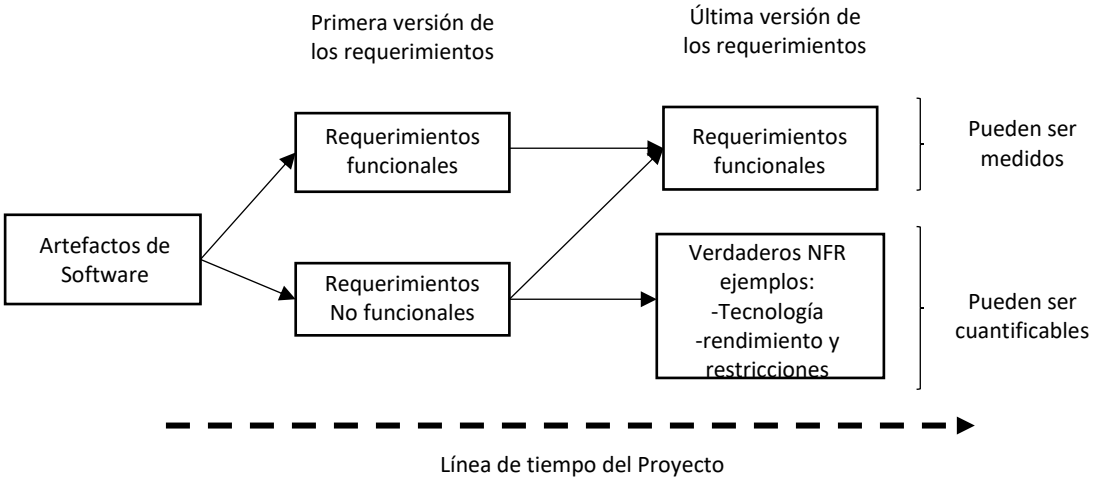


Figura 0.1 - Evolución de Requerimientos Funcionales y No Funcionales

1.2.2. Diseño

En la fase de diseño se pretende obtener una descripción de la "mejor" solución que de soporte a los requerimientos, teniendo no solamente en cuenta aspectos técnicos, sino también aspectos de calidad, costo y tiempo. Se define el término *diseño* de la siguiente manera:

> *Definición.* El **diseño** [3] puede definirse como el proceso para definir la arquitectura, los componentes, las interfaces y otras características de un sistema o un componente, y como el resultado de este proceso.

Se denomina **principios de diseño de software** a los siguientes conceptos fundamentales de diseño [2]:

- Abstracción: Es el proceso de olvidarse de la información para poder tratar las cosas que son diferentes como si fueran iguales.
- Acoplamiento y cohesión: El acoplamiento es la fuerza de las relaciones entre los módulos y la cohesión se refiere a la relación que existe entre estos módulos.
- Descomposición y modularización: la descomposición permite definir componentes de alto nivel en otros de bajo nivel, mientras que la modularización se utiliza para poner diversas funcionalidades en componentes más pequeños.
- Encapsulamiento y ocultación de información: Permite empaquetar información y hacerla invisible. El éxito está en darle una buena función a esa información
- Separación de interfaz e implementación: Define un componente a través de su interfaz gráfica.
- Suficiencia, compleción (calidad) y sencillez.

Uno de los objetivos del diseño es la especificación de componentes o módulos del sistema, y del modo en que estos se comunican sin describir sus detalles internos.

> *Definición.* Un **componente** [3] es una de las partes para formar un sistema. Un componente puede ser hardware o software y puede subdividirse en otros componentes.

> *Definición.* Una **interfaz** [3] es un componente de hardware o software que conecta dos o más componentes con el fin de transmitir información de uno a otro.

Otra propiedad importante en la fase de diseño que hay que tomar en cuenta es la *arquitectura del sistema*, aunque es un concepto muy amplio, se define de la siguiente manera:

> *Definición.* La **arquitectura de un sistema** [12] software es la organización fundamental de dicho sistema plasmada en sus componentes, las relaciones entre éstos y con el entorno, y los principios que guían su diseño e implementación.

1.2.3. Construcción

El término *construcción del software* define un conjunto de actividades que engloban fundamentalmente la codificación, pero también la verificación del código, su depuración y

pruebas. El producto resultante de la construcción de software es el entregable más importante de un proyecto de desarrollo. Para diferenciar el término *construcción del software* de *codificación*, se da la definición de cada uno de la siguiente manera:

> *Definición.* La **construcción del software** [1] es la actividad de combinar los elementos de configuración del software, usando la configuración de datos apropiada, en un programa ejecutable para su distribución a los clientes u otros receptores.

> *Definición.* **Codificar** [3] es el proceso de expresar un programa de computadora en un lenguaje de programación.

La construcción del software está relacionada especialmente con las actividades de diseño, pruebas y mantenimiento. Para construir una pieza de software se necesita un esfuerzo previo en su diseño y otro posterior para comprobar que funciona adecuadamente.

Los principios fundamentales de la construcción de software son los siguientes [2]:

- Minimizar la complejidad.
- Anticipar los cambios.
- Construir para verificar.
- Reutilizar
- Utilizar estándares en la construcción.

1.2.4. Pruebas

Hacer pruebas es una actividad que tiene el objetivo de evaluar y mejorar la calidad del producto, identificando defectos y problemas. Las pruebas de software, al contrario que el resto de actividades, su éxito está en la detección de errores en el proceso de desarrollo y en el software obtenido.

> *Definición.* Una **prueba de software** [3] es el proceso de analizar un componente de software para detectar las diferencias entre las condiciones existentes y los requerimientos (es decir, los errores) y evaluar las características del componente de software.

Durante las pruebas se tiene presente el hecho de que los usuarios ejecutarán todas las funcionalidades del sistema, lo que implica que el software será puesto a prueba, ya sea por los desarrolladores o bien por los usuarios. Por lo tanto, los errores que no sean detectados en las pruebas realizadas durante el desarrollo aparecerán cuando los usuarios utilicen el software.

Todo proceso que permite comprobar el comportamiento de un software con el comportamiento que se espera según sus requerimientos podría ser catalogado como prueba de software, siendo los aspectos fundamentales a verificar en un software su corrección, compleción y seguridad.

1.2.5. Mantenimiento

En la propia definición de la Ingeniería de Software aparece el mantenimiento como una de sus elementos. Las actividades de mantenimiento están orientadas a la modificación o cambio del software mismo, pero para realizar modificaciones, se necesita entender el software que se requiere cambiar, y sólo después se podría hacer la modificación requerida.

Un determinado producto de software está orientado a satisfacer ciertos requerimientos previamente establecidos. El mantenimiento entonces se entiende de manera general como las actividades que producirán cambios en el producto inicialmente acordado. Se define el mantenimiento de la siguiente manera:

> *Definición.* El **mantenimiento** [13] del software es la modificación de un producto de software después de la entrega para corregir fallos, para mejorar su rendimiento u otros atributos, o para adaptar el producto a un entorno modificado.

En esta definición las actividades de mantenimiento de un producto se llevan a cabo sólo después de que dicho producto haya sido entregado y puesto en operación.

La guía SWEBOK [1] considera que el mantenimiento ocurre durante todo el ciclo de vida y coincide con la siguiente definición:

> *Definición.* El **mantenimiento** [14] del software son la totalidad de las actividades necesarias para proporcionar un soporte rentable al sistema software. Estas actividades se desarrollan tanto antes como después de la entrega. Las actividades previas a la entrega incluyen la planificación de las operaciones posteriores a la entrega, la planificación del soporte y la determinación de la logística. Las actividades posteriores a la entrega incluyen la modificación del software, la formación de usuarios y la puesta en marcha de un servicio de soporte técnico y atención al cliente.

El proceso de mantenimiento del software considera las siguientes actividades para los cambios de una pieza de software [13]:

1. Petición de modificación.
2. Identificación y Clasificación.
3. Análisis.
4. Diseño.
5. Implementación.
6. Prueba del sistema.
7. Prueba de aceptación.
8. Entrega.

1.3. Ejercicios

1.3.1. Cuestionario de autoevaluación

1.1. ¿Cuáles son los tres elementos fundamentales de la definición de la Ingeniería de Software?

1.2. De los elementos fundamentales, ¿Cuál es el menos desarrollado?

1.3. En el contexto de la Ingeniería de Software ¿Qué diferencia un proceso de un método?

1.4. Menciona tres técnicas para el mantenimiento del software

1.5. Además de los Procesos Fundamentales, ¿Cuáles son las otras áreas clave que identifica la guía SWEBOK?

1.6. ¿Qué es una herramienta CASE?

1.7. ¿Qué es una metodología ágil?

1.8. ¿Qué es una metodología en cascada?

1.9. ¿Qué es un modelo iterativo?

1.10. Menciona tres marcos de procesos utilizados en la Ingeniería de Software

1.11. Responde verdadero o falso las siguientes afirmaciones:

- Puede darse el caso que algunos requerimientos del sistema que se expresan inicialmente como no funcionales evolucionan a medida que el proyecto avanza en requerimientos funcionales.
- El software se desarrolla, no se fabrica.
- Decimos que algo es cuantificable cuando está bien organizado y sigue reglas o estándares.
- Entre los objetivos de la Ingeniería de Software está el brindar mayor control en los costos de proyectos y tiempo de desarrollo de los mismos.
- El software se desgasta.
- El éxito de las pruebas de software está en la detección de errores en el proceso de desarrollo y en el software obtenido.
- El mantenimiento ocurre durante todo el ciclo de vida del software.
- En la Ingeniería de Software, los métodos determinan el orden y la forma de llevar a cabo las actividades.
- La definición de los roles profesionales en una organización no es un aspecto de la Ingeniería de Software.
- Uno de los objetivos del diseño es la especificación de componentes o módulos del sistema, y del modo en que estos se comunican sin describir sus detalles internos.

2. Introducción a las Métricas de Software

En este segundo capítulo se da una introducción a las Métricas de Software, se abarcan los fundamentos y la definición de los conceptos básicos de éstas. La primera parte del capítulo abarca los fundamentos de medición y la definición de los conceptos, la segunda y tercera parte tratan sobre que se pude medir en el software y algunos tipos de medidas utilizados en la práctica. En la última parte del capítulo se habla sobre el diseño de métodos de medición de software, como diseñar y aplicar un método de medición.

El objetivo de este segundo capítulo es dar una introducción del fundamento de las Métricas de Software, que el alumno entienda para que nos sirven estas métricas, y como diseñar y aplicar un método de medición formal, como los métodos de medición de tamaño funcional existentes (capítulo 3).

Este capítulo termina con una serie de ejercicios propuestos para poner en práctica lo aprendido e investigar más sobre el tema.

La información que se presenta en este capítulo fue recopilada principalmente del libro “Software Metrics and Software Metrology” [15], las definiciones de conceptos fueron recopiladas principalmente del “Vocabulario Internacional de Metrología” (International Vocabulary of Metrology – VIM) [16].

2.1. Fundamentos de medición

2.1.1. ¿Por qué medir?

La Ingeniería de Software es una disciplina que está aún aprendiendo a medir, a estimar y a mejorar la calidad de sus productos y procesos.

> *"medir lo que sea medible, y hacer medible lo que no lo es"*
>
> *Galileo Galilei (1564-1642)*

> *"Una ciencia es tan madura como sus herramientas de medición"*
>
> *Luis Pasteur (1822, 1895)*

En la Ingeniería de Software, a menudo no somos capaces de hacer estimaciones precisas, no alcanzamos la calidad suficiente y nos excedemos en el presupuesto. La medición permite ganar comprensión acerca del proyecto, al proporcionar un mecanismo de evaluación objetiva.

> *"cuando puedes medir aquello de lo que hablas y expresarlo en números, sabes algo acerca de ello; pero cuando no puedes medir, cuando no puedes expresarlo en números, tu conocimiento es pobre e insatisfactorio"*
>
> *William Thomson (1824-1907)*

La medición puede aplicarse al proceso de desarrollo de software con la intención de mejorarlo de manera continua, puede utilizarse para auxiliar en la estimación, control de calidad, valoración de productividad y control del proyecto.

> *"todo lo que se hace se puede medir, sólo si se mide se puede controlar, sólo si se controla se puede dirigir y sólo si se dirige se puede mejorar"*
>
> *Dr. Pedro Mendoza (1962 - presente)*

2.1.2. Conceptos básicos de las Métricas de Software

La Organización Internacional de Metrología Legal (International Organization of Legal Metrology – OIML), en el VIM define a la Metrología de la siguiente manera:

> *Definición.* **Metrología** [16] es la ciencia de las mediciones y sus aplicaciones**.**

Para el estudio de las Métricas de Software, se debe tener claro el significado del término *métrica*. Este término se utiliza como referencia a diferentes conceptos: la cantidad a ser medida, el procedimiento de medición, el resultado de la medición o para modelos entre múltiples mediciones.

> *Definición.* **Métrica** [3] es una medida cuantitativa del grado en que un sistema, componente o proceso posee un *atributo* dado.

Para el estudio de la medición, es importante diferenciar entre medición y medida.

Definición. **Medición** [16] es un proceso que consiste en obtener experimentalmente uno o varios valores de *atributos* de *entidades* del mundo real.

Definición. **Medida** [17] es una variable a la que se le asigna un valor como resultado de una medición.

Definición. **Unidad de medida** [17] es una determinada cantidad definida y adoptada por convenio con la que se comparan otras cantidades del mismo tipo para expresar su magnitud relativa a esa cantidad.

El propósito de un proceso de medición de software es reunir, analizar y reportar datos vinculados con los productos desarrollados y los procesos implementados [17].

Las definiciones anteriores, mencionan los términos de *entidad* y *atributo*.

Definición. Se denomina **entidad** [17] a un objeto que va a ser caracterizado mediante la medición de sus atributos.

Definición. Un **atributo** [3] es una característica de una entidad.

2.1.3. Tipos de escalas de medición

Otro concepto importante en el mundo de la medición es la noción de escala. Al realizar una medición y asignar valores a atributos de entidades, se utilizan *escalas de medición*, donde cada atributo tendrá entonces su tipo de escala determinada la cual debe mantenerse inalterable, lo que es posible modificar son los valores.

Definición. Una **escala de medición** [16] es un conjunto ordenado de valores de utilizado para clasificar magnitudes en orden creciente o decreciente según sus valores cuantitativos.

Cada tipo de escala engloba a todas las escalas que admiten las mismas transformaciones admisibles, es decir, los tipos de operaciones matemáticas que se pueden aplicar a una escala garantizando que se mantiene la condición de representación, como lo expresa Sánchez et al. [2]:

- **Escala nominal**: formada por categorías entre las cuales no existe ningún orden; por ejemplo, la variable sexo que tiene dos categorías (Hombre, Mujer)
- **Escala ordinal**: en la que se definen categorías donde existe una relación de orden entre ellas; por ejemplo, la variable en donde se asignen valores (muy de acuerdo, de acuerdo, indiferente, en desacuerdo y muy en desacuerdo.
- **Escala de intervalo**: donde la distancia entre intervalos es conocida y siempre es la misma; por ejemplo, fechas en un calendario, la temperatura Celsius, etc.
- **Escala de ratio**: tiene un valor inicial de referencia o cero absoluto, y permite definir relaciones coherentes con los valores de la escala lo que permite comparar valores estableciendo proporciones; por ejemplo, la escala Kelvin de medición de temperaturas, el peso, etc.

- **Escala absoluta**: consiste simplemente en la cuenta sin transformación del número de entidades; por ejemplo, contar el número de personas en un proyecto.

2.1.4. Clasificación de medidas

Es importante mencionar que no existe una única clasificación de medidas. Una clasificación habitual consiste en dividir las medidas de un atributo en dos tipos, directas e indirectas [2]:

- **Medidas directas**: las medidas directas de un atributo son aquellas que pueden ser obtenidas directamente de la entidad sin necesidad de ningún otro atributo.
- **Medidas indirectas**: son aquellas que se derivan de una o más medidas de otros atributos. Se definen y calculan, por tanto, a partir de otras medidas.

Otra clasificación diferencia entre medidas objetivas y subjetivas, pues dependiendo de uno y otro tipo de éstas se emplean como base de muy diferentes estudios [2]:

- **Medidas objetivas**: una medida cuyo valor no depende del observador.
- **Medidas subjetivas**: son aquellas en las que la persona que realiza la medición puede introducir factores de juicio en el resultado.

2.2. La medición del software

2.2.1. La Ingeniería de Software como una disciplina

En el campo de la Ingeniería de Software, como ya se ha mencionado, el término métrica es usado como referencia a múltiples conceptos, por ejemplo, la calidad a ser medida, el procedimiento de la medición, el resultado de la medición, etc. Abran [15] menciona que en la Ingeniería de Software el término es aplicado a:

- La medición de un concepto, por ejemplo, la complejidad ciclomática (métrica propuesta por Thomas McCabe en 1976).
- Modelos de calidad, por ejemplo, ISO 9126 – la calidad del producto de software.
- Modelos de estimación, por ejemplo, algorítmicos, no algorítmicos, como son la ecuación de esfuerzo de Halstead en 1977, COCOMO I y II, etc.

Lo que no hay lugar a duda, es que la medición es fundamental en la Ingeniería de Software como una disciplina ingenieril. En las últimas décadas, cientos de llamadas métricas de software han sido propuestas por los investigadores y profesionales, tanto en estudios teóricos y empíricos, para la medición de los productos y procesos de software [15]:

- La mayoría de estas métricas fueron diseñadas con base en la intuición, ya sea por parte de los investigadores o sobre una base empírica, o ambos.
- En el análisis de algunas de ellas, los investigadores han utilizado con mayor frecuencia los conceptos de la teoría de la medición como la base para su investigación analítica.
- En la ciencia, incluyendo la ingeniería, así como administración de empresas y un número significativo de las ciencias sociales, la medición es uno de una serie de instrumentos analíticos. La medición en estas ciencias se basa en un gran cuerpo de conocimiento acumulado durante siglos, incluso milenios, lo que se conoce comúnmente como "metrología".

En la literatura sobre las métricas de software, no hay casi ninguna referencia a los conceptos clásicos de la metrología en las investigaciones sobre la calidad de las métricas propuestas para la Ingeniería de Software. Sólo recientemente algunos de los conceptos relacionados con la metrología han sido introducidos a la Ingeniería de Software, incluyendo la selección del vocabulario ISO sobre metrología como base para la terminología de medición para todas las futuras normas ISO sobre la medición del software [16].

Una de las diferencias de la Ingeniería de Software con otras disciplinas científicas y de ingeniería es su falta de uso general de los datos cuantitativos para la toma de decisiones. Las razones de esto son [15]:

- Un número limitado de medidas de software aceptadas por los practicantes y reconocidas como maduras a través de un estándar.
- Un número pequeño de estudios experimentales rigurosos.

Mientras que en las disciplinas maduras hay:

- Un amplio consenso internacional sobre medidas.
- Métodos de medición establecidos y referencias (etalón).
- Un número significativo de instrumentos de medición para ser utilizados en distintos contextos y bajo ciertas condiciones, muchos de ellos calibrados y certificados contra referencias internacionales.

En las disciplinas maduras, las medidas también se reconocen como un costo necesario de hacer negocios y de llevar a cabo las actividades de ingeniería, así como una necesidad para mejorar la toma de decisiones.

En la Ingeniería de Software, no tenemos ninguno de los anteriores, con la excepción de la reciente adopción de normas ISO para la medición del tamaño funcional.

2.2.2. ¿Qué medir en la Ingeniería de Software?

Una métrica debe poder medir adecuadamente el atributo de la entidad a medir, pero también definir cómo se va a realizar la medición. En la Ingeniería de Software encontramos tres tipos de entidades [2]:

- **Productos**. Cualquier artefacto, entregable o documento que resulta de cualquier proceso del ciclo de vida del software. Algunos ejemplos de productos son el código fuente, especificación de requerimientos, manuales de usuario, etc.
- **Procesos**. Todas las actividades del ciclo de vida del software: requerimientos, diseño, construcción, pruebas, mantenimiento, etc. Algunas de las métricas relacionadas con los procesos son el tiempo invertido en las actividades o el tiempo para reparar un defecto.
- **Recursos**. Cualquier entrada de una actividad. Por ejemplo, el número de personas por proyecto, dinero, esfuerzo, las herramientas utilizadas, oficinas, etc.

Las entidades tienen o se les asignan atributos a medir, Sánchez et al. [2] distinguen entre atributos internos y atributos externos:

- **Atributos internos**. Los atributos internos de una entidad son aquellos que se pueden medir directamente a partir de dicha entidad. Por ejemplo, podemos medir directamente el número de defectos que se han encontrado en una pieza de software.
- **Atributos externos**. Los atributos externos de una entidad son aquellos que los relacionan con el entorno y se deducen en función de atributos internos. Los atributos externos se suelen relacionar con la calidad.

Para efectos del curso consideraremos que los atributos internos del software se pueden dividir en dos factores que son:

- factores técnicos, y
- factores funcionales.

En la Figura 2.1 se muestra para quienes son significativos y algunas características que tienen estos factores.

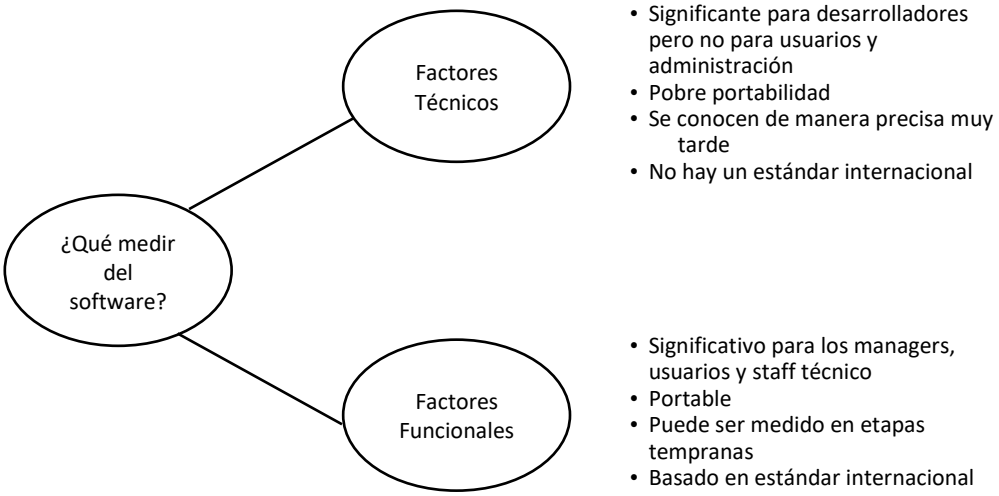


Figura 2.1 - Qué medir en el software

2.3. Tipos de Medidas

2.3.1. Medidas del tamaño

La métrica más antigua para los proyectos de software es la de **líneas de código** (Lines of Code – LOC) [18] introducido por primera vez alrededor de 1960 y se utilizó para estudios económicos, de productividad y de calidad, cuándo el término software era prácticamente sinónimo del término *programa de computadora*.

Contar las líneas de código es una de las métricas más usadas, principalmente por su simplicidad [2]. Sin embargo, aunque esta métrica puede parecer sencilla, su definición y modo de uso debe ser clara para que todos los interesados entiendan y apliquen la métrica del mismo modo. Por ejemplo, considerar las líneas en blanco y/o los comentarios como líneas válidas, etc.

A medida que se crearon lenguajes de programación de nivel superior, los resultados de utilizar LOC comenzó a tener problemas, las métricas de LOC no fueron capaces de medir actividades que no implicaran el uso de código, como requerimientos y diseño que cada vez eran más costosas. Dados los problemas que presentaba utilizar LOC, han aparecido métricas de tamaño más perfeccionadas.

Las métricas de software orientadas a función usan una medida de la funcionalidad del software como un valor de normalización. El estándar ISO/IEC 14143 [19] da la definición de tamaño funcional y define las características de medición de tamaño funcional.

> *Definición*. El **tamaño funcional** [20] se define como un tamaño de software derivado de cuantificar los Requerimientos Funcionales de Usuario.

La métrica orientada a función de mayor uso es el Punto de Función (Function Point – FP), descritos en 1979 por Allan Albrecht, que miden el tamaño funcional del software que proporciona a los usuarios, sin considerar el código fuente como lo hace LOC. Esta métrica se basa en el conteo de funciones de datos (archivos lógicos internos y archivos de interfaz externa) y funciones transaccionales (entradas, salidas y consultas externas) que son ponderadas por un factor de ajuste de hasta +/- 35%. (Consultar más detalle de esta métrica en el capítulo 3.2.5*)* [18].

La Figura 2.2 muestra los resultados del informe anual del Laboratorio de las TI según David Consulting Group [21] que determina las tendencias sobre que métodos utilizó la industria para medir el software en el año 2012. Dentro del ámbito de la muestra se encuentran empresas de diferentes sectores como: Aeroespacial, Educación, Finanzas, Gobiernos, Sanidad, Manufacturas, Servicios, Software y Telecomunicaciones, distribuidas geográficamente en América del Norte, Sudamérica, Europa, Asia, India y Australia.

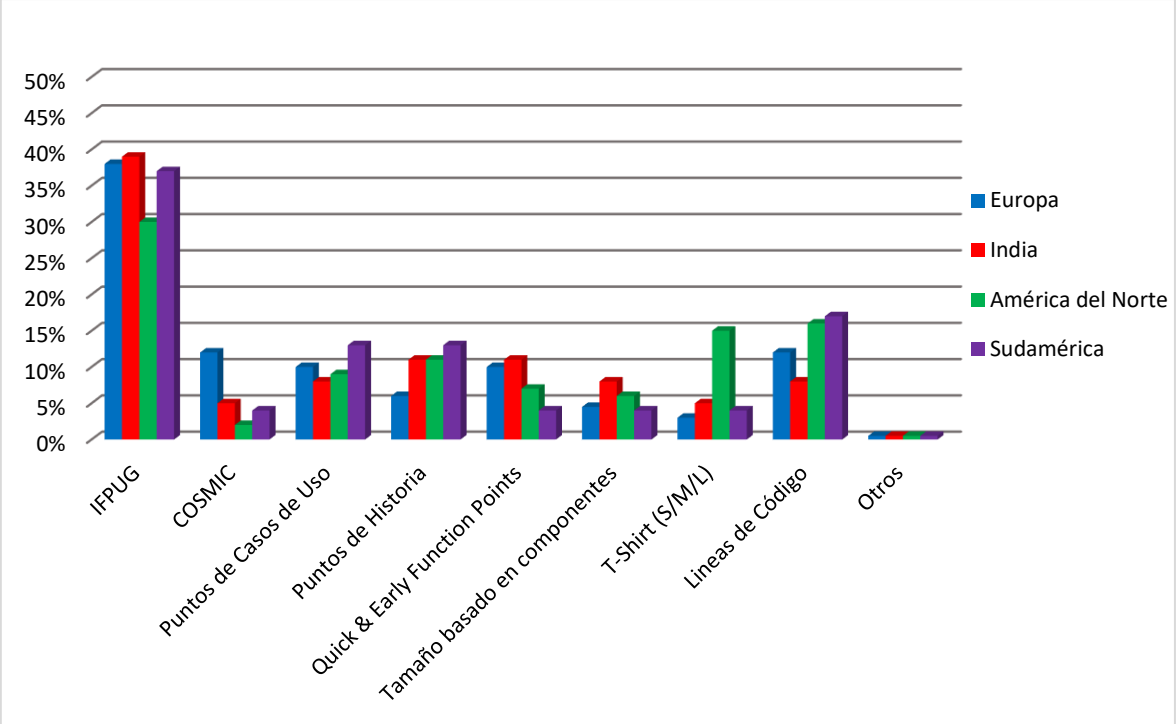


Figura 2.2 - Método de medición utilizado por zona geográfica 2012 (adaptada de [21])

Como los puntos de función, el *caso de uso* se define al principio del proceso del software, lo que permite emplearlo para una estimación antes de iniciar actividades de diseño y construcción. Los casos de uso se utilizan ampliamente como un método para describir los requerimientos, que implican características y funciones del software.

> *Definición*. Un **caso de uso** [22] es todas las formas de utilizar un sistema para lograr un objetivo particular para un usuario en particular.

Basado en el auge de UML y en particular considerando los casos de uso, Gustav Karner desarrolla el método de dimensionamiento **puntos de casos de uso** (use case points – UCP) en 1993 [23]. Esta métrica se basa en lógica y cálculos similares a los puntos de función en concepto, pero no en detalles específicos. Los factores que intervienen en los puntos de casos de uso incluyen factores de complejidad técnica y de ambiente. Una vez calculados, los puntos de casos de uso se pueden utilizar para estimar el esfuerzo y los costos para el desarrollo de un proyecto software [24].

A pesar de que los puntos de casos de uso son una métrica utilizada en las empresas de software, este documento no desarrolla a detalle la aplicación del método debido a que éste no es un estándar internacional como sí lo son los que se desarrollan en el capítulo 3.

2.3.2. Medidas de la complejidad del software

Se sabe desde hace muchos años que la complejidad del código es difícil de mantener y tiene tasas de defectos muy altas [24].

La complejidad ciclomática de McCabe [25] consiste en medir y controlar el número de caminos independientes de un programa. La complejidad ciclomática plantea inmediatamente el siguiente problema: "Cualquier programa con un arco hacia atrás potencialmente tiene un número infinito de rutas".

La medida de la complejidad ciclomática se define en términos de caminos básicos que, cuando son combinados, generarán todos los caminos posibles.

El **número ciclomático V(G)** [25] de un gráfica G con n vértices, e aristas, y p componentes conexas es:

$V(G) = e - n + p$

Otros usos de la complejidad de McCabe son medir la complejidad de la integración de módulos, evitar la dificultad de automatizar las pruebas de un código o incluso medir su fiabilidad.

2.3.3. Medidas relacionadas con el proceso

Existen dos tipos de métricas del proceso, las internas que incluyen el tiempo y el esfuerzo de desarrollo, y las externas que incluyen la estabilidad del proceso o costo [2].

Entre los modelos de mejora de procesos más conocidos se encuentran la guía de Integración de Modelos de Madurez de Capacidades (Capability Maturity Model Integration – CMMI), el Modelo de Procesos para la Industria del Software (MoProSoft), ISO/IEC 15504 (SPICE) e ISO 9000. Algunos de estos modelos, como CMMI y SPICE, agrupan los procesos con sus respectivas métricas en distintos niveles de madurez, con la finalidad de que una organización vaya mejorando sus procesos.

Los modelos CMMI [26] proporcionan una guía de uso para desarrollar procesos. Los modelos CMMI no son procesos ni descripciones de procesos. Los procesos reales utilizados en una organización dependen de muchos factores, incluyendo dominios de aplicación, dominios de estructura y tamaño de la organización.

CMMI se centra en mejorar los procesos en una organización. Contiene elementos esenciales para la eficiencia de los procesos en una o más disciplinas, y describe un camino evolutivo de mejora desde procesos ad hoc e inmaduros a procesos disciplinados y maduros con calidad y eficiencia mejoradas [26].

El concepto de *productividad* es muy utilizado en las medidas relacionadas con el proceso y los recursos.

> *Definición*. La **productividad** [7] se define como la relación entre la cantidad de trabajo obtenido y el esfuerzo invertido.
>
> *productividad = tamaño / esfuerzo*

2.4. Diseño de modelos de medición de Software

2.4.1. Introducción a números, medidas y modelos cuantitativos [15]

Un número en sentido matemático lo aprendemos desde la primaria, conforme avanzamos en la escuela, aprendemos reglas matemáticas, las operaciones básicas, ecuaciones, etc. Cuando tratamos con números, solo en sentido matemático, no hay incertidumbre asociada a estos números.

Un número obtenido como resultado de una medición es un número asignado a un atributo específico de una entidad específica que se expresa generalmente como un número con una unidad de medida y un grado de incertidumbre asociada al procedimiento de medición. Si la incertidumbre de medida se considera despreciable para un determinado fin, el resultado de medida puede expresarse como un único valor medido. Generalmente, un número obtenido de un ejercicio de medición está limitado a las características medidas, por ejemplo, medir la temperatura solo te da información de la temperatura usando una escala de medición (como Celsius) pero el resultado no te dice si está lloviendo o nevando.

Los modelos de decisión y los modelos de evaluación representan reglas sobre cómo interpretar medidas de diferentes tipos. Por ejemplo, un modelo para predecir cuándo dejar de probar un software podría tomar en cuenta el número de errores así como una serie de parámetros adicionales como el tamaño del software, el número de pruebas, periodo de pruebas, etc. Estos modelos son distinguidos por medio de un modelo de contexto de medición.

2.4.2. Modelo de Contexto de Medición

El Modelo de Contexto de Medición [15] establece los distintos pasos del diseño de un método de medición de software, ayuda a comprender la cadena de medición y ayuda a identificar los criterios de verificación necesarios en cada paso para asegurar que los resultados sean sólidos.

El Modelo de Contexto de Medición se conforma de los siguientes tres pasos [15]:

1. **Diseño.** Antes de medir, es necesario decidir que método de medición se va a utilizar (si existe) o diseñar uno.
2. **Aplicación.** Una vez que el método de medición ha sido seleccionado o diseñado, sus reglas son aplicadas al software para obtener un resultado de medición.
3. **Explotación.** El resultado de la medición es explotado en un modelo cuantitativo o cualitativo, usualmente combinado con otro resultado de medición.

El primer paso se refiere al *diseño* del método de medición:

- La entrada a este paso es la descripción del objeto a ser medido.
- La salida es la identificación de los conceptos de medición y la descripción del método de medición para la implementación de la asignación de reglas numéricas.

El segundo paso se refiere a la *aplicación* del método de medición en un contexto específico:

- Las entradas en este paso es el objeto a ser medido y el método de medición.
- La salida no solo es el resultado de la medición, también la calidad y el grado de incertidumbre del resultado de medición.

El tercer paso se refiera a la explotación del resultado de medición:

- Incluye el análisis del resultado cuantitativo del modelo, usualmente para propósitos de evaluación, estadística, comparación o predicción.

La Figura 2.3 ilustra los tres pasos del Modelo de Contexto de Medición con sus entradas, pasos intermedios y las salidas correspondientes, así como la relación entre cada paso.

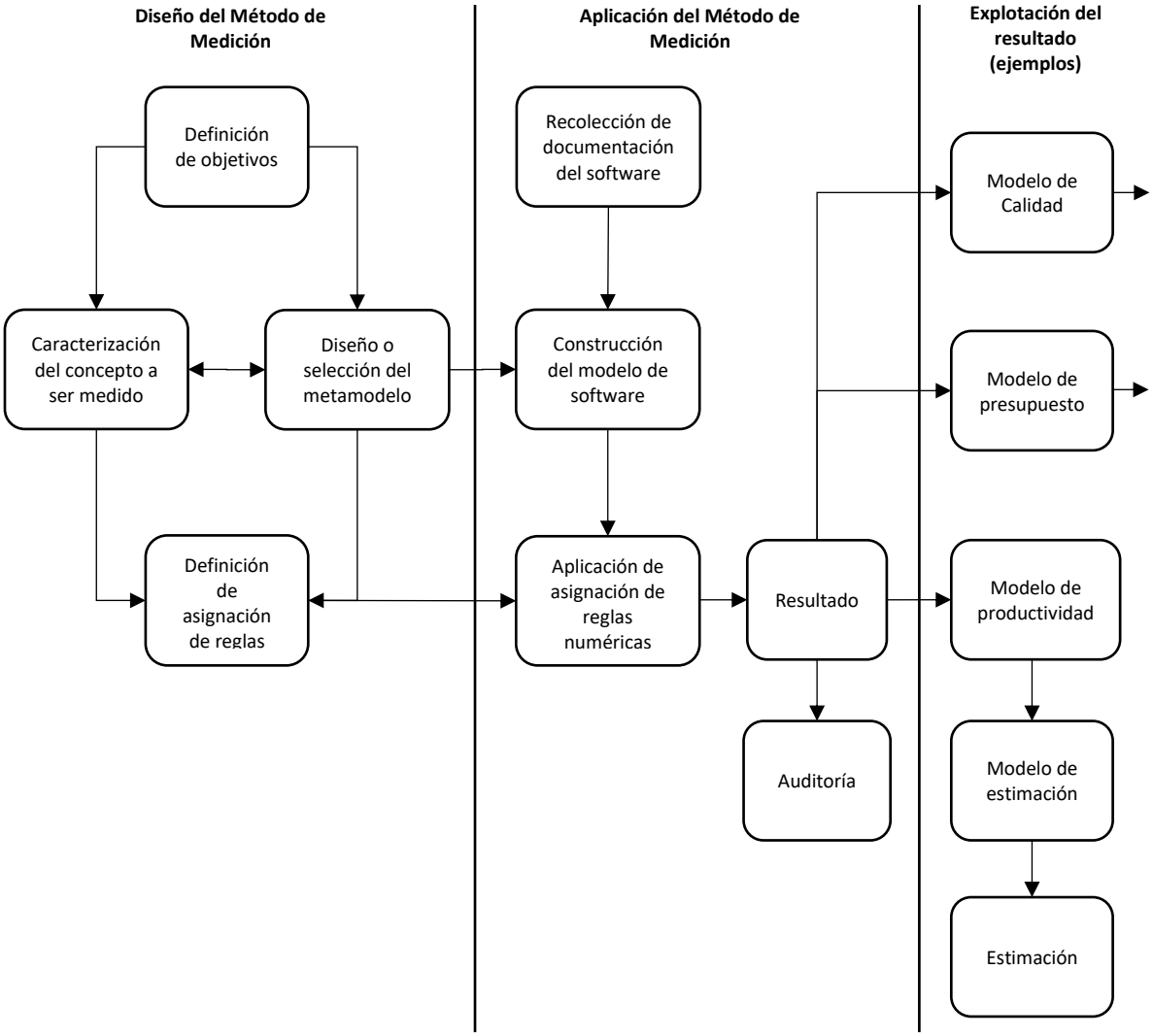


Figura 2.3 - Detalle del Modelo de Contexto de Medición (adaptada de [15])

Los modelos cuantitativos pueden ser muy útiles en la toma de decisiones. Es responsabilidad del ingeniero de software determinar que:

- Los propios modelos cuantitativos son confiables y que proporcionan una base sólida para la toma de decisiones, con restricciones y limitaciones.
- Las entradas a estos modelos son confiables y sus niveles de incertidumbre y precisión son conocidas.
- El diseño de estos modelos es confiable y significativo.

2.5. ¿Cómo medir el Software?

2.5.1. Principio, método y procedimiento de medición

¿Cuándo se mide el software?, ¿se puede medir directamente o a través de un procedimiento de medición y dar como resultado un valor? La respuesta es sí, el Modelo de Contexto de Medición trae consigo todas las actividades necesarias para la aplicación de un método de medición.

- Si el método de medición ya existe, se omite el primer paso (diseñar un método de medición), como sucede con las disciplinas maduras.
- Si el método de medición no existe aún (como sucede frecuentemente en el caso del software), este paso se debe realizar antes de especificar el procedimiento de medición para un contexto de medición particular.

Para obtener suficiente información sobre cómo medir y como tener suficiente grado de confianza en los resultados de la medición, se debe entender esta información como una transición a través de tres niveles: *principio de medición*, *método de medición* y medición a través de un *procedimiento de medición* [16].

El principio de medición da la definición precisa de que es lo que vamos a medir.

> *Definición*. El **principio de medición** [16] se define como el fenómeno que sirve como la base de una medición.

En las disciplinas maduras, el resultado de esta fase es la información obtenida experimentalmente y se hace referencia a las teorías y leyes bien establecidas. El atributo a medir debe ser definido de manera clara y precisa, para que se pueda caracterizar de manera suficientemente clara y sin ambigüedades

> *Definición*. El **método de medición** [16] se define como una descripción genérica de una secuencia lógica de operaciones utilizada en una medición.

Es decir, representar un atributo de una entidad a ser medida mediante el resultado de medición, dando una descripción general de cómo vamos a medir

> *Definición*. El **procedimiento de medición** [16] es una descripción detallada de una medición conforme a uno o más principios de medida y a un método de medida dado, basado en un modelo de medida y que incluye los cálculos necesarios para obtener un resultado de medida.

Un aspecto importante en la medición es describir precisamente el contexto en el que se llevará a cabo el procedimiento de medición. Como mínimo, los parámetros que deben tenerse en cuenta son:

- El propósito del proceso de medición
- Las limitaciones con las que se llevará a cabo la medición

Un método de medición debe ser implementado por un procedimiento de medición para describir una medición de acuerdo con uno o más principios de medición (Figura 2.4).

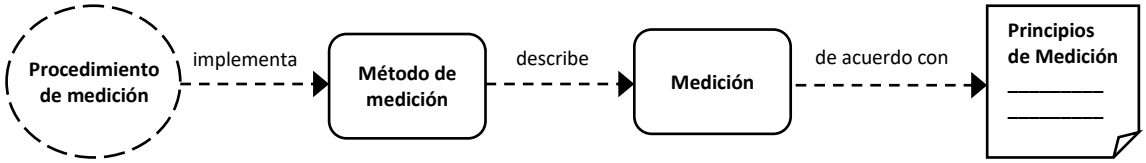


Figura 2.4 - Relaciones entre procedimiento de medición, método de medición, medición y principios de medición.

Un método de medición a su vez:

- Se compone de las operaciones concretas realizadas por medio de los instrumentos de medición y/o acciones prácticas, tales como la selección, cálculo, comparación, etc.
- Es más específico, más detallado, y más relacionado con ambiente y los instrumentos de medición que el método utiliza.

2.5.2. Como diseñar un método de medición

Este documento se describe en el campo de conocimiento del software, por tanto, el enfoque planteado para el diseño de un método de medición es para el software.

El diseño de un método de medición (como se muestra en la Figura 2.3) consiste en cuatro pasos [15]:

1. Determinar los objetivos de la medición. Antes de diseñar un método de medición de software, es importante conocer:
    - Qué queremos medir
    - Cuál es el punto de vista de medición
    - Para qué será utilizado el resultado
2. Caracterización del concepto a ser medido, especificando las entidades y las características (atributos) a considerar.
3. Desarrollo del metamodelo incluyendo las relaciones a través del concepto (entidad y atributo).
    - El meta modelo debe ser descrito genéricamente, es decir, no debe ser específico para una pieza de software en particular y debe ser independiente del contexto de medición
    - El meta modelo también debe describir como reconocer los atributos a tener en cuenta para medir las entidades involucradas en el ejercicio de medición
    - El meta modelo también debe describir el rol que cada atributo juega en la composición del concepto a ser medido y como estos atributos son definidos.
4. Definición de reglas de asignación numérica. Desde el punto de vista matemático, para caracterizar un concepto es necesario definir las reglas de asignación numérica e incluir la selección (o diseño) de una unidad de medida, respetando que las transformaciones de la escala correspondiente sean válidas.

El modelado de estos pasos está basado en una lógica perspectiva y el reconocimiento de que se requieren mejoras iterativas para mejorar la propuesta de diseño inicial de un concepto medible y su correspondiente método de medición.

2.5.3. Aplicación de un método de medición

Una vez que el método de medición ha sido diseñado junto con todos los entregables del diseño, el método puede ser aplicado para medir una pieza de software específica (como se muestra en la Figura 2.3). La "aplicación del método de medición" a "un contexto de medición específico" corresponde al diseño de un procedimiento de medición de un ejercicio de medición (Figura 2.4).

La aplicación de un método de medición se lleva a cabo a través de los siguientes cinco pasos [15]:

Paso 1 - Recolección de la documentación del software. La información para la aplicación del método de medición es recolectada de la siguiente manera:

- Del software a ser medido, cuando el software está disponible para su revisión.
- De la documentación del software, por ejemplo, cuando el software aún no ha sido construido.

Paso 2 - Construcción del modelo de software a ser medido. Una vez que la documentación se ha recolectado, el modelo del software es construido de acuerdo a las reglas de la medición construida, el modelo describe como el software a ser medido es representado por el método de medición.

Paso 3 - Asignación de reglas numéricas. La asignación de las reglas numéricas se aplica al modelo de software derivado de los pasos 1 y 2.

Paso 4 - Presentación de resultados de la medición. Aplicando las reglas de medición hace posible obtener resultados de la medición, de manera que pueden ser evaluados, este resultado generalmente se documenta con:

- La unidad de medida
- La descripción de pasos intermedios de la medición
- La descripción del proceso de medición
- Las medidas
- El procedimiento de medición

Paso 5 - Verificación de resultados de la medición. Los resultados de la medición deben ser validados y auditados, usando los métodos necesarios para cerciorarse de la calidad. Los resultados también pueden ser comparados con otros resultados conocidos con el fin de evaluar su exactitud.

2.5.4. Explotación del resultado de medición

Los resultados de la aplicación de un método de medición pueden ser utilizados de muchas maneras, la Figura 2.3 muestra algunos usos potenciales de un resultado de medición de software [15]:

- En modelos de evaluación, como modelos de calidad.
- En modelos de presupuestos.
- En un proceso de estimación, que en sí se basa en un modelo de productividad y un modelo de estimación.

Los temas de proceso de estimación, modelo de productividad y modelo de estimación se desarrollan a detalle en el capítulo 5.3.

2.6. Ejercicios

2.6.1. Cuestionario de autoevaluación

2.1. ¿Cuál es la diferencia entre medida y métrica?

2.2. ¿Sobre qué entidades fundamentales se realizan mediciones en la Ingeniería de Software?

2.3. ¿Cuál es la diferencia entre las disciplinas maduras e inmaduras?

2.4. ¿Cuáles son los tres pasos que conforman el Modelo de Contexto de Medición?

2.5. ¿Cuál es la métrica orientada a función más utilizada en las empresas de software?

2.6. Menciona tres ejemplos de la explotación de un resultado de medición en la administración de proyectos de software.

2.7. A las entidades se les asignan atributos a medir, ¿cuál es la diferencia entre atributos internos y externos?

2.8. ¿En qué consiste la complejidad ciclomática de McCabe?

2.9. Todas las mediciones tienen asociada una incertidumbre, menciona algunos de los factores a los que puede deberse dicha incertidumbre:

2.10. ¿Cuáles son las siete unidades de medición que reconoce el Sistema Internacional de Unidades?

2.11. Responde verdadero o falso las siguientes afirmaciones:

- La metrología es la ciencia de las mediciones y sus aplicaciones.
- Una unidad de medida es una variable a la que se le asigna un valor como resultado de una medición.
- Siempre que dos medidores apliquen la métrica de líneas de código sobre un código fuente de distinto lenguaje, obtendrán los mismos resultados.
- La efectividad operacional conlleva cualquier número de prácticas que le permiten a una empresa utilizar de mejor manera los insumos de producción.
- El estándar ISO/IEC 15504 - SPICE es un modelo para la mejora y evaluación de los procesos de desarrollo y mantenimiento de sistemas de información y productos de software.
- La productividad se define como la relación entre el esfuerzo invertido y la cantidad de trabajo obtenido (*productividad = esfuerzo / tamaño).*
- Los modelos de decisión y los modelos de evaluación representan reglas sobre cómo interpretar medidas de diferentes tipos, estos modelos son distinguidos por medio un modelo de contexto de medición.

- La barrera de productividad es el máximo valor que una compañía puede entregar sobre un producto o servicio a un costo dado.
- Un ejemplo de escala de ratio es la escala Fahrenheit de medición de temperaturas.
- El propósito de un proceso de medición de software es reunir, analizar y reportar datos vinculados con los productos desarrollados y los procesos implementados.

3. Estándares para la Medición

En este tercer capítulo se da la descripción de algunos estándares y métodos de medición de Software. En la primera parte del capítulo, entre los estándares que se definen, se describe el estándar ISO/IEC 14143 que define los conceptos de medición de tamaño funcional, donde cinco estándares ISO ponen en práctica la cuantificación de estas características de tamaño funcional. El resto del capítulo trata sobre estos cinco estándares para la medición de tamaño funcional.

Una vez que se ha aprendido como diseñar y aplicar un método de medición (capítulo 2), el objetivo de este tercer capítulo es aplicar un método de medición estándar ya existente, para posteriormente explotar el resultado de la medición, por ejemplo, utilizando el resultado como entrada a un modelo de estimación (capítulo 5).

Este capítulo termina con una serie de ejemplos y ejercicios propuestos para poner en práctica lo aprendido y para investigar más sobre el tema.

La información que se presenta en este capítulo fue recopilada principalmente cada uno de los estándares ISO que se menciona en el capítulo, así como el manual de medición de cada uno de los cinco estándares de medición de tamaño funcional y el estándar ISO/IEC 14143 [20] al que dan cumplimiento.

3.1. Estándares para la medición

3.1.1. ISO/IEC 15939:2007 - Measurement process: Proceso de medición

El estándar ISO/IEC 15939:2007 – Proceso de medición [17] define un proceso de medición aplicable a disciplinas de Ingeniería de Software y gestión de sistemas. El proceso se describe a través de un modelo que define las actividades del proceso de medición necesarias para especificar adecuadamente qué información se necesita, cómo se aplicarán las medidas y análisis de resultados y cómo determinar si los resultados son válidos. El proceso de medición es flexible y adaptable a las necesidades de los diferentes usuarios.

El estándar se compone de dos aspectos:

1. El modelo de información de la medición, y
2. El proceso de medición.

El modelo de información de la medición define los términos de uso común relativos a la medición del software y la relación entre las necesidades de información, los tipos de medidas o métricas y las entidades a medir.

El proceso de medición se compone de cuatro actividades fundamentales [15]:

1. Establecer y mantener el compromiso de medición, estableciendo un compromiso empresarial para realizar mediciones.
2. Planificar el proceso de medición, consiste en la selección de las medidas de acuerdo con los objetivos y características de la organización y definir los procedimientos de cómo y cuándo se van a realizar las mediciones.
3. Ejecutar el proceso de medición, realizando y almacenando las mediciones.
4. Evaluación de los resultados obtenidos, evaluando tanto las mediciones como el proceso de medición.

3.1.2. IEEE 1061-1998 - Software Quality Metrics Methodology: Metodología para métricas de calidad del software

Siempre se mide con la idea de mejorar y aumentar la calidad de las entidades involucradas en los procesos del software. Sin embargo, es poco práctico medir todos los atributos de todas las entidades, es por ello debemos determinar qué medir basándonos en metodologías y modelos de calidad bien definidos.

El estándar IEEE 1061-1998 Metodología para métricas de calidad del software [27] describe un modelo de proceso para establecer requerimientos de calidad de software e identificar, implementar, analizar y validar métricas de calidad de software. Estos son definidos por el estándar como:

"*Una función cuyas entradas son datos de software y cuya salida es un valor numérico único que puede ser interpretado como el grado en que el software posee un atributo dado que afecta su calidad*".

El estándar representa un proceso que se ejecuta durante casi todo el ciclo de vida del software. La correspondencia de los pasos de la metodología del estándar con las fases del ciclo de vida del software depende del tipo de métricas de calidad.

El estándar comprende cinco pasos que deben realizarse de manera iterativa [27]:

1. Establecer los requerimientos de calidad del software, puede hacerse durante la fase de análisis de requerimientos.
2. Identificar las métricas de calidad del software, puede hacerse en la fase de diseño.
3. Implementar las métricas de calidad, puede hacerse en la fase de construcción.
4. Analizar los resultados de las métricas de calidad de software, puede hacerse en la fase de construcción, pruebas y fases posteriores.
5. Validar las métricas de calidad de software, es posible cuando se dispone de suficientes datos de calidad para analizar, calibrar y validar las métricas. Esto es útil para aplicar las métricas validadas a fases posteriores del proyecto o a futuros proyectos.

3.1.3. ISO/IEC 14143 - Functional size measurement: Medición de tamaño funcional

El estándar ISO/IEC 14143 [19] , al igual que algunos otros ISO/IEC, se compone de las siguientes partes:

- Parte 1: Definición de conceptos.
- Parte 2: Evaluación de la conformidad de los métodos de medición del tamaño del software.
- Parte 3: Verificación de los métodos de medición del tamaño funcional.
- Parte 4: Modelo de referencia.
- Parte 5: Determinación de dominios funcionales para su uso con medición de tamaño funcional.
- Parte 6: Guía para el uso de ISO/IEC 14143 y estándares internacionales relacionados.

La Medición del Tamaño Funcional (Functional Size Measurement - FSM) es una técnica utilizada para medir el tamaño del software cuantificando los Requerimientos Funcionales de Usuario (Functional Size Requirements – FUR).

Cinco Métodos de Medición del Tamaño Funcional (Functional Size Measurement Method - FSMM) y estándares ISO/IEC se reconocen métodos de medición específicos para poner en práctica la cuantificación de estas características de tamaño funcional [19]:

- ISO/IEC 19761 - COSMIC
- ISO/IEC 20926 - IFPUG
- ISO/IEC 20968 - Mk II
- ISO/IEC 24570 - NESMA
- ISO/IEC 29881 - FiSMA

La medición del software no tiene las características que definen a una disciplina madura, con la excepción de estándares ISO para la medición de tamaño funcional. Por ello, la situación actual de la medición del software se considera inmadura en virtud de que [15]:

- Gran parte de las medidas de software propuestas por la industria son empíricas y no han sido analizadas a detalle, no son lo suficientemente maduras.
- A diferencia de otros campos de la ingeniería y de la ciencia, estas medidas de software carecen de credibilidad para ser utilizadas como base de toma de decisiones.
- El criterio de verificación para medidas de software debería ser fácil de entender, bien definido y conciso.
- Los diseñadores de medidas de software deberían documentar cómo las medidas propuestas cumplen con estos criterios de verificación.

3.1.4. ISO/IEC 25000:2014 - Systems and software Quality Requirements and Evaluation (SQuaRE) -- Guide to SQuaRE: Requerimientos y Evaluación de Calidad de Productos de Software (SQuaRE)

El estándar ISO/IEC 25000 Requerimientos y Evaluación de Calidad de Productos de Software (Systems and software Quality Requirements and Evaluation - SQuaRE) [28] define una guía de normas basadas en ISO/IEC 9126 y en ISO/IEC 14598 para el uso de la nueva serie de estándares internacionales.

El propósito de ISO/IEC 25000 es proporcionar una visión general de los contenidos de SQuaRE, modelos de referencia y definiciones comunes, así como guiar la evaluación de calidad de productos software estableciendo criterios para la especificación de requerimientos de calidad de software, sus métricas y su evaluación [28].

La familia de normas ISO/IEC 25000 (SQuaRE) está compuesta por cinco divisiones [29]:

- ISO/IEC 2500n - Quality Management Division: División para gestión de calidad.
- ISO/IEC 2501n - Quality Model Division: División para el modelo de calidad.
- ISO/IEC 2502n - Quality Measurement Division: División para la medición de la calidad.
- ISO/IEC 2503n - Quality Requirements Division: División para los requerimientos de calidad.
- ISO/IEC 2504n – Quality Evaluation Division: División para la evaluación de la calidad.

Las normas que forman la división de **gestión de calidad** definen todos los modelos, términos y definiciones comunes referenciados por todas las otras normas de la familia 25000. Actualmente esta división se encuentra formada por:

- ISO/IEC 25000 - Guide to SQuaRE: Guía para SQuaRE. Contiene el modelo de la arquitectura de SQuaRE, la terminología de la familia, un resumen de las partes, los usuarios previstos y las partes asociadas, así como los modelos de referencia.

- ISO/IEC 25001 - Planning and Management: Planeación y Gestión. Establece los requerimientos y orientaciones para gestionar la evaluación y especificación de los requerimientos del producto software.

Las normas de la división del **modelo de calidad** presentan modelos de calidad detallados incluyendo características para calidad interna, externa y en uso del producto software.

La calidad interna se relaciona con las propiedades estáticas del software mientras que la calidad externa hace referencia a las características asociadas con la ejecución del software en el hardware y el sistema operativo, es decir la calidad interna/externa puede ser definida como una sola, considerada como la calidad relativa a la construcción del software, mientras que la calidad en uso considera la calidad del sistema en su ambiente operacional para los usuarios y para las tareas específicas que realizan, es decir, está relacionada al uso del software [30].

Esta división se encuentra formada por:

- ISO/IEC 25010 - System and software quality models: Modelos de calidad de sistemas y software. Describe el modelo de calidad para el producto software y para la calidad en uso. Esta Norma presenta las características y subcaracterísticas de calidad frente a las cuales evaluar el producto software.
- ISO/IEC 25012 - Data Quality model: Modelo de calidad de datos. Define un modelo general para la calidad de los datos, aplicable a aquellos datos que se encuentran almacenados de manera estructurada y forman parte de un Sistema de Información.

El modelo SQuaRE supone que la calidad de los procesos presenta dependencia tanto de la calidad interna/externa, como de la calidad en el uso, y estas a su vez, se ven influenciadas por la calidad en el desarrollo de software, tal como se observa en la Figura 3.1.

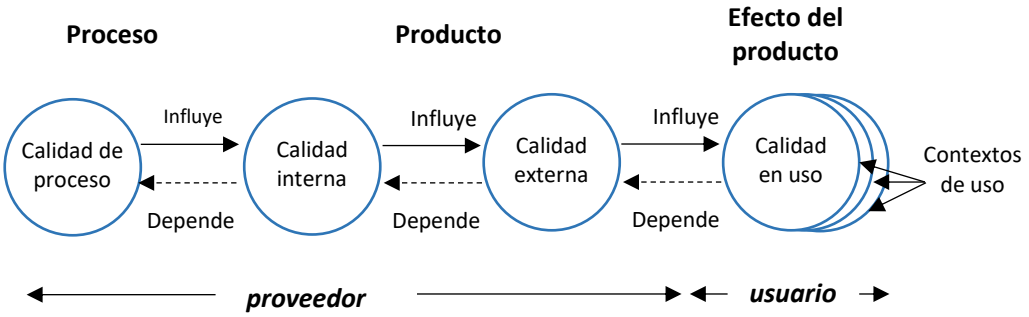


Figura 3.1 – Modelo de referencia de Calidad de Software

El modelo de calidad del producto se encuentra compuesto por ocho características de calidad (Figura 3.2).

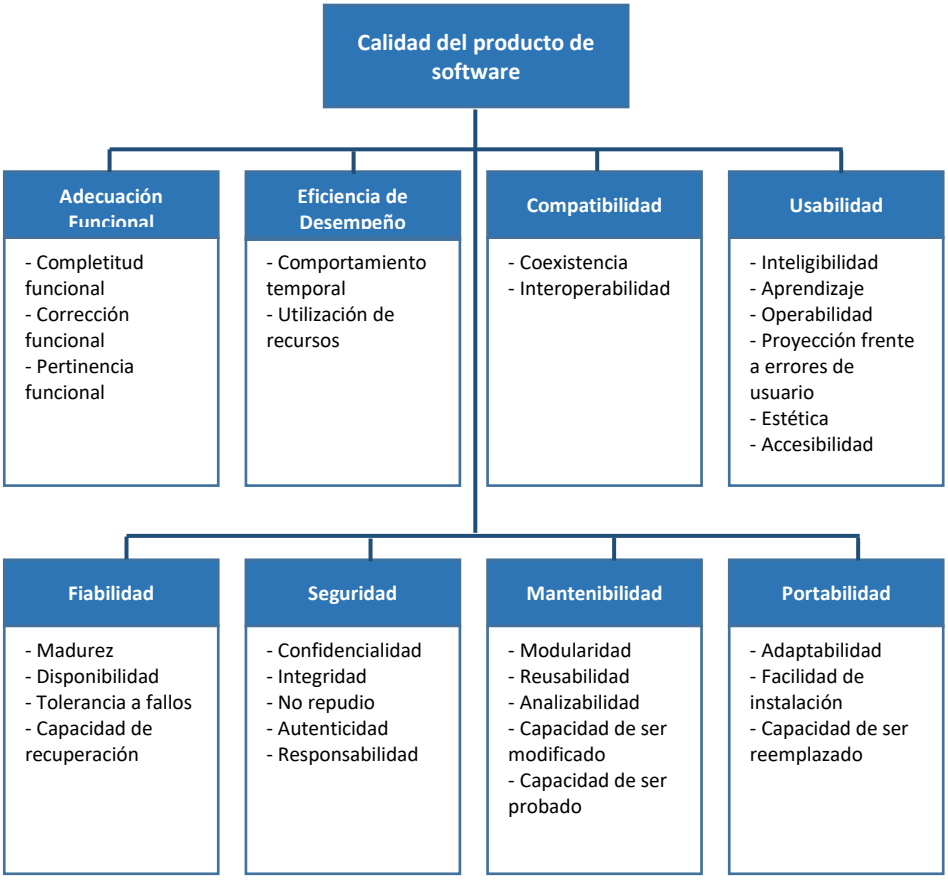


Figura 3.2 – Modelo de calidad del producto

Las normas de la división de la **medición de la calidad** incluyen un modelo de referencia de la medición de la calidad del producto, definiciones de medidas de calidad (interna, externa y en uso) y guías prácticas para su aplicación. Actualmente esta división se encuentra formada por:

- ISO/IEC 25020 - Measurement reference model and guide: Modelo y guía de referencia de medición. Presenta una explicación introductoria y un modelo de referencia común a los elementos de medición de la calidad. También proporciona una guía para que los usuarios seleccionen o desarrollen y apliquen medidas propuestas por estándares ISO.
- ISO/IEC 25021 - Quality measure elements: Elementos de medición de calidad. Define y especifica un conjunto recomendado de métricas base y derivadas que puedan ser usadas a lo largo de todo el ciclo de vida del desarrollo software.
- ISO/IEC 25022 - Measurement of quality in use: Medición de la calidad en uso. Define específicamente las métricas para realizar la medición de la calidad en uso del producto.

- ISO/IEC 25023 - Measurement of system and software product quality: Medición de la calidad del producto del sistema y del software. Define específicamente las métricas para realizar la medición de la calidad de productos y sistemas software.
- ISO/IEC 25024 - Measurement of data quality: Medición de la calidad de los datos. Define específicamente las métricas para realizar la medición de la calidad de datos.

Las normas que forman la división de **requerimientos de calidad** ayudan a especificar requerimientos de calidad que pueden ser utilizados en el proceso de especificación de requerimientos de calidad del producto software a desarrollar o como entrada del proceso de evaluación. Para ello, esta división se compone de:

- ISO/IEC 25030 - Quality requirements: Requerimientos de calidad. Provee de un conjunto de recomendaciones para realizar la especificación de los requerimientos de calidad del producto software.

Las normas de la división de **evaluación de calidad** proporcionan requerimientos, recomendaciones y guías para llevar a cabo el proceso de evaluación del producto software. Esta división se encuentra formada por:

- ISO/IEC 25040 - Evaluation reference model and guide: Modelo y guía de referencia de evaluación. Propone un modelo de referencia general para la evaluación, que considera las entradas al proceso de evaluación, las restricciones y los recursos necesarios para obtener las salidas correspondientes.
- ISO/IEC 25041 - Evaluation guide for developers, acquirers and independent evaluators: Guía de evaluación para desarrolladores, compradores y evaluadores independientes. Describe los requerimientos y recomendaciones para la implementación práctica de la evaluación del producto software desde el punto de vista de los desarrolladores, de los adquirentes y de los evaluadores independientes.
- ISO/IEC 25042 - Evaluation modules: Módulos de evaluación. Define lo que la norma considera un módulo de evaluación y la documentación, estructura y contenido que se debe utilizar a la hora de definir uno de estos módulos.
- ISO/IEC 25045 - Evaluation module for recoverability: Módulo de evaluación para la recuperación. Define un módulo para la evaluación de la subcaracterística Recuperabilidad (Recoverability).

La división de extensión de SQuaRE (ISO/IEC 25050 a ISO/IEC 25099) se reserva para normas o informes técnicos que aborden dominios de aplicación específicos o que puedan ser utilizados para complementar otras normas de la familia SQuaRE.

3.2. Métodos de Medición de Tamaño Funcional (Primera Generación)

3.2.1. Introducción a los métodos de medición de tamaño funcional

En los años 70's Allan J. Albrecht's desarrolló una forma de medición funcional del software independiente de la plataforma de desarrollo y que consideraba el punto de vista del usuario, llamada Análisis de Puntos de Función (Function Points Analysis – FPA), actualmente conocida como Puntos de Función [31], esta técnica de medición considera conceptos vigentes para aquel tiempo cómo archivos lógicos, además tiene un alcance solo para Sistema de Información Gerencial (Management Information System – MIS), en la actualidad existe una gran variedad de dominios de aplicaciones (no solo las MIS) como lo son en tiempo real, sistemas de Planificación de Recursos Empresariales (Enterprise Resource Planning – ERP), Almacenes de Datos (Data Warehouse – DWH), etc.

Más tarde a finales de los años 80's surgió un método llamado Mark II, este método estaba basado en transacciones lógicas y en el modelado de relaciones de entidades, populares en ese tiempo. Estos métodos y otros como: 3D Function Points, Feature Points, FISMA, etc., desarrollados hasta el año 2000, son considerados de la primera generación de FSMM.

La característica es que estos métodos se generaron a partir de estudios estadísticos realizados en alguna empresa (FP en IBM), el análisis expresa en sus resultados la manera de operar de la empresa para los tipos de proyectos analizados. Cuando se trata de utilizar estos métodos en alguna otra empresa o con otro tipo de proyectos, los resultados pueden no ser adecuados ya que la empresa y/o los proyectos operan de manera distinta.

Los criterios definidos como contexto de referencia inicial en el estudio realizado por Albrecht fueron [15]:

- La organización incluye a 450 personas que desarrollan aplicaciones.
- El desarrollo está bajo contrato con clientes de IBM.
- Los desarrolladores y los clientes están dispersos por los Estados Unidos.
- En cualquier momento entre 150 y 170 contratos están abiertos.
- Un contrato medio implica dos o tres personas.
- Cada contrato se realiza en el contexto de un marco de desarrollo específico.
- La mayoría de los contratos se limitan a ciertas fases de la metodología.
- Con base en su experiencia, la fase de diseño representa el 20% de las horas de trabajo, la implementación del 80%.
- Es necesario medir todo el proceso, incluyendo el diseño y los costos incurridos durante el diseño.
- Los proyectos se completaron desde mediados de 1974 hasta principios de 1979.
- El tamaño de los proyectos varía de 500 a 105 000 horas de trabajo.
- Cerca de 1500 contratos para el período, sólo 22 cumplieron los criterios de selección.

Así mismo los criterios que utilizó para aceptación de proyectos que participaron en el estudio fueron [15]:

1. El proyecto debe haber pasado por todas las fases de la metodología (desde la definición de los requerimientos hasta la implementación) y debe haber sido entregado al cliente.
2. Todo el proyecto debe haber sido gestionado adecuadamente con definiciones consistentes de las tareas y los procesos de gestión.
3. Todas las horas de trabajo de IBM y del cliente deben ser conocidas y deben haber sido cuidadosamente medidas.
4. Los factores funcionales deben ser conocidos.

La segunda generación de métodos de FSM está representada por COSMIC (Common Software Measurement International Consortium), este estándar se basa en sólidos principios de Ingeniería de Software y no está basado en estudios estadísticos, se generó considerando como base el estándar ISO/IEC 14143 y la experiencia de los métodos de la primera generación, lo que implica que resuelve gran parte de los problemas presentados en dichos métodos como el manejo de conceptos actualmente no vigentes, el alcance de aplicación de los métodos y una escala de medición más práctica y homogénea, además de tener un dominio de aplicación mayor.

La Figura 3.3 muestra la evolución de los FSMM de primera generación y COSMIC como el primer y único FSMM de segunda generación.

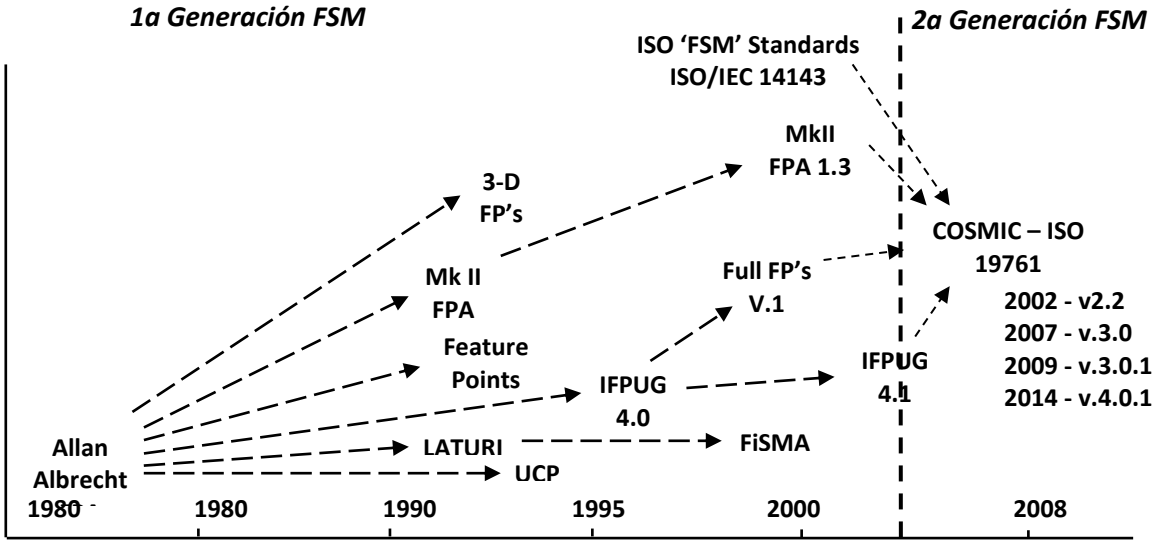


Figura 3.3 - Evolución de los métodos de medición de tamaño funcional (adaptada de Abran [15])

3.2.2. ISO/IEC 20968:2002 - Mk II FPA 1.3.1 [32]

Mk II FPA fue definido por Charles Symons como 'Software Sizing and Estimating: Mk II FPA' publicado en 1991. Este método ha sido mantenido y difundido por la Asociación de Métricas del Reino Unido (United Kingdom Software Metrics Association – UKSMA) en varias versiones.

A finales del 2002, la versión 1.3.1 de Mk II FPA fue reconocida como un estándar a través de ISO/IEC 20968:2002. El método Mk II FPA ha sido el primer método de medición de tamaño funcional reconocido como estándar, debido a que la versión 1.3.1 fue desarrollada conforme al estándar ISO/IEC 14143-1.

Mk II FPA es un método para el análisis cuantitativo y la medición de aplicaciones de procesamiento de información. Cuantifica los requerimientos de procesamiento de información especificados por el usuario para proporcionar un número que expresa el tamaño de una pieza de software.

Mk II FPA es un método que ayuda en la medición de la eficiencia del proceso y la gestión de los costos para las actividades de desarrollo de software, mejoras o mantenimiento. Este método mide el tamaño de una pieza de software independiente de las características técnicas del software. El método Mk II FPA puede ser:

- Aplicado al inicio del proceso de desarrollo de software.
- Aplicado de manera uniforme a lo largo de la vida del software.
- Interpretado en términos de negocio.
- Entendido por los usuarios del software.

MK II FPA es independiente del método de gestión de proyectos que se va a utilizar (por ejemplo, cascada, espiral, incremental) y del método de desarrollo empleado (por ejemplo, orientado a objetos, Ingeniería de la Información, etc.).

Los puntos de función Mk II pueden usarse para medir el tamaño funcional de cualquier aplicación de software que se pueda describir en términos de transacciones lógicas, compuestas por un único componente de entrada, proceso y salida, que son desencadenadas por la ejecución de un evento por el usuario. Luego, se asigna un peso a los elementos identificados, y finalmente, se suman los pesos de los elementos identificados para obtener el tamaño funcional.

Las reglas de dimensionamiento fueron diseñadas para ser aplicadas a piezas de software en el dominio de aplicaciones de negocio, en el que el componente de procesamiento de cada transacción tiende a ser procesos de almacenamiento o recuperación de datos.

El método puede ser aplicado al software de otros dominios, pero el usuario debe tener en cuenta que las reglas de medición no tienen en cuenta algoritmos complejos que se encuentran típicamente en software científico y de ingeniería, ni las reglas para tomar en cuenta software de tiempo real. Para aplicar Mk II FPA a estos otros dominios puede ser posible diseñar ampliaciones o nuevas interpretaciones de las reglas del método.

3.2.3. ISO/IEC 24570:2005 - NESMA FPA VER.2.1

El método Nesma fue fundado por la Asociación Holandesa de Usuarios de Métricas de Software (Netherlands Software Metrics Users Association – NESMA) en 1989 [33]. El método se basó en la versión 3.4 del manual del Grupo Internacional de Usuarios de Puntos

de Función (International Function Point User Group – IFPUG), entregando guías concretas, consejos y ejemplos para aplicar el método de medición definido por IFPUG.

La razón de la creación del método fue debido a que los usuarios de NESMA no estaban de acuerdo con algunas de las guías de medición del manual de IFPUG, que eran percibidas como "demasiado técnicas". Más tarde, IFPUG adhirió cada vez más a las ideas de NESMA.

En el 2004, la versión 2.0 de NESMA fue reconocida como estándar por ISO/IEC 24570:2004 [34], debido a que su definición es conforme al estándar ISO/IEC 14143-1.

La guía de medición de FPA according to NESMA and IFPUG [35] indica que las reglas de medición entre ambos métodos son casi las mismas, usan la misma terminología, tienen los mismos criterios para identificar procesos elementales y funciones de datos, y diferencian los cinco tipos de funciones de usuario.

A pesar de ser muy similares, NESMA tiene los siguientes métodos adicionales para medir el tamaño del software [19]:

- FPA estimado (Estimated FPA), también llamado FPA de alto nivel (High Level FPA)
- FPA indicativo (Indicative FPA)

El método FPA de alto nivel y el método FPA indicativo no necesitan requerimientos de usuario detallados. El tamaño funcional usando estos métodos es muy cercano al tamaño funcional usando el método FPA detallado. Es por eso que estos dos métodos son muy adecuados para ser aplicados en etapas tempranas del ciclo de vida de desarrollo de software o en caso de que el tamaño funcional debe medirse rápidamente [36].

3.2.4. ISO/IEC 29881:2010 - FiSMA 1.1 FSMM

El método FiSMA [37] fue desarrollado por un grupo de trabajo de la Asociación Finlandesa de Medición de Software (Finnish Software Measurement Association – FiSMA), para reemplazar el método de FPA. Las razones para construir un nuevo método fueron las limitaciones y las complicadas reglas de interpretación para determinar el tipo de componente funcional de FPA.

La primera versión de FiSMA fue publicada en 1991 con el nombre original de "método Laturi", desde entonces, el método ha evolucionado sobre la base de la investigación científica y las empresas [38]. En el 2008, la versión 1.1 del manual de medición de FiSMA fue reconocido como estándar por ISO/IEC 29881:2008 [39].

FiSMA es aplicable para medir una pieza de software en cualquier dominio funcional. Además, el método no tiene limitaciones respecto a la calidad de la pieza de software a ser medida [37].

Cuando los practicantes están familiarizados con otros FSMM, al estudiar FiSMA se puede observar que este método mide especificaciones más concretas de los requerimientos funcionales de usuario y adopta un enfoque diferente a cualquiera de los otros métodos.

El proceso de medición FiSMA consta de dos partes principales: medición de servicios de interfaces de usuario final y medición de servicios indirectos. Si una de estas dos partes no existe en la pieza de software, entonces el proceso consiste sólo en medir los servicios que están presentes [37].

3.2.5. ISO/IEC 20926:2009 - IFPUG FSMM [18]

IFPUG se basa en el método FPA definido por Albretch. En 1986, fue fundado IFPUG que se encarga de la difusión y de la creación de manuales de uso del método, publicando varias versiones de manuales (3.4, 4.0, 4.1, 4.1.1, y 4.2).

En el 2003, el manual de medición versión 4.1 fue reconocido como estándar por ISO/IEC 20926. Los objetivos de IFPUG se basan en el análisis de software mediante la cuantificación de la funcionalidad del software basado principalmente en el diseño lógico. Con esto en mente, los objetivos de análisis de puntos función son los siguientes:

- Medir funcionalidad donde el usuario solicita y recibe.
- Medir el desarrollo de software y mantenimientos, independientemente de la tecnología utilizada.

Es necesario considerar los criterios definidos como contexto de referencia inicial en el estudio realizado por Albrecht. Esto muchas veces se pasa por alto, sin embargo, es muy relevante tener estas consideraciones ya que los resultados de la medición utilizando el método FP serán relevantes en la medida que estos criterios se cumplan.

El método IFPUG sugiere que la identificación de elementos que contribuyen al tamaño funcional debe ser realizada a partir de uno o varios de los siguientes componentes: requerimientos de usuario, modelos de datos, modelos de procesos, ventanas o pantallas, y reportes de la pieza de software. La Figura 3.4 muestra el diagrama del proceso de medición de Puntos de Función.

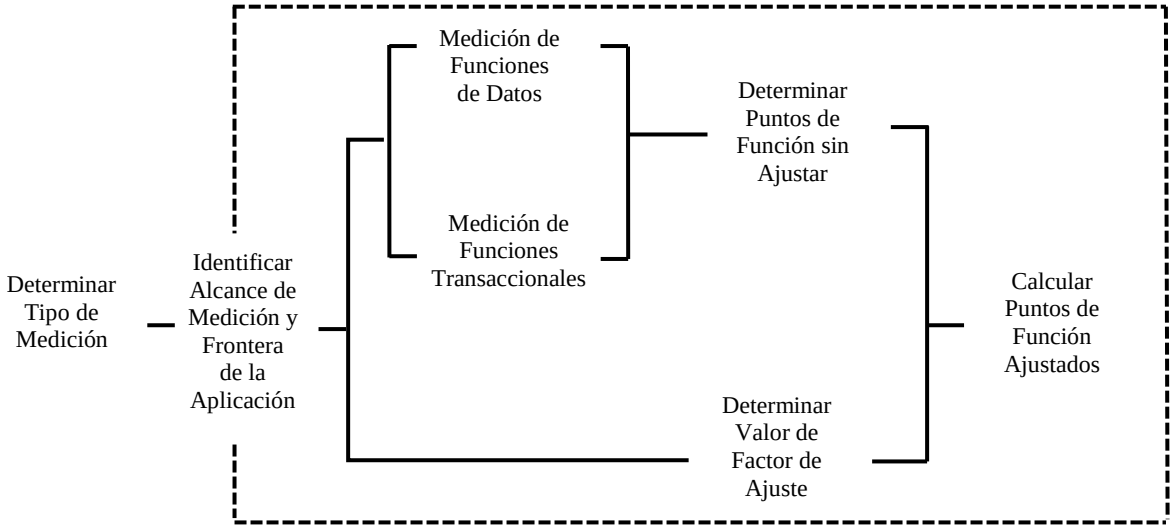


Figura 3.4 - El proceso de medición del método IFPUG

El proceso de medición visto a un nivel de abstracción alto contempla dos partes:

- El cálculo de Puntos de Función no Ajustados (Unadjusted Function Points – UFP)
- El Valor de Factor de Ajuste de los FP (Value Adjustement Factor – VAF)

Para el cálculo de UFP se deben de identificar los elementos funcionales:

- **EI** (Entradas Externas - External Inputs),
- **EO** (Salidas Externas - External Outpu),
- **EQ** (Consultas Externas - External Inquiries),
- **ILF** (Archivos Lógicos Internos - Internal Logic Files) y
- **EIF** (Archivos de Interfaz Externa - External Interface Files).

Esta identificación se realiza en base a reglas definidas por el manual, sin embargo, tienen un grado de incertidumbre al ser clasificaciones subjetivas. Esta clasificación implica el manejo de escalas nominales.

Una vez identificados los elementos funcionales y con base a tablas (Tabla 3.1, Tabla 3.2, Tabla 3.3) de peso se obtiene para cada elemento funcional su complejidad (baja, media y alta). Para realizar esto se deben de identificar los subcomponentes funcionales:

- **DET's** (Tipo de Elemento de Datos - Data Element Type) y
- **RET's** (Tipo de Elemento de Registro - Record Element Type),

Los DET's son campos de datos únicos identificados por el usuario, los RET's son los subgrupos de datos en el mismo almacenamiento y que son identificables por el usuario.

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

Tabla 3.1 - Cálculo de complejidad de ILF, EIF

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

Tabla 3.2 - Cálculo de complejidad de EI

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

Tabla 3.3 - Cálculo de complejidad de EO, EQ

El siguiente paso es obtener el peso en UFP para cada elemento funcional con base en su complejidad, para realizar esto se utilizan también tablas de peso, de manera resumida se muestra una matriz de contribución en la siguiente tabla.

	Baja	Media	Alta	Totales
ILF	___ x 7	___ x 10	___ x 15	_____
EIF	___ x 5	___ x 7	___ x 10	_____
EI	___ x 3	___ x 4	___ x 6	_____
EO	___ x 4	___ x 5	___ x 7	_____
EQ	___ x 3	___ x 4	___ x 6	_____
			UFP =	_____

Tabla 3.4 - Matriz de contribución

Una vez calculados los UFP se debe de realizar el cálculo del Valor de Factor de Ajuste (Value Adjustment Factor - VAF), este factor de ajuste modifica los UFP en un ±35% dependiendo de cómo influyen las siguientes 14 características generales del sistema (General System Characteristics - GSCs) que califican a la aplicación como un todo en requerimientos no funcionales:

GSC	Describe el grado mediante el cual:
1. Comunicación de Datos	La aplicación se comunica directamente con el procesador.
2. Procesamiento de Datos Distribuidos	La aplicación transfiere datos entre componentes físicos de la aplicación.
3. Rendimiento	Se considera el tiempo de respuesta y desempeño como parte del desarrollo de la aplicación.
4. Configuración de Uso Intensivo	Se consideran restricciones de recursos de cómputo como parte del desarrollo de la aplicación.
5. Nivel de Transacciones	La velocidad de las transacciones de negocio influye en el desarrollo de la aplicación.
6. Entrada de Datos en Línea	Los datos son introducidos o recuperados a través de transacciones interactivas (transacciones en línea).
7. Eficiencia del Usuario Final	Se consideran factores humanos y facilidad de uso al usuario.
8. Actualización en Línea	Los ILFs son actualizados en línea.
9. Procesamiento Complejo	La lógica de proceso influye en el desarrollo de la aplicación.
10. Reutilización	La aplicación y el código han sido diseñados, desarrollados y soportados para ser utilizados en otras aplicaciones.
11. Facilidad de Instalación	Conversión de ambientes anteriores y facilidad de instalación influyen en el desarrollo de la aplicación.

12. Facilidad de Operacional	La aplicación considera aspectos operacionales como procesos de recuperación, de inicio, de respaldo.
13. Múltiples Sitios	La aplicación ha sido desarrollada para ambientes diferentes tanto de hardware como de software.
14. Facilidad de Cambio	La aplicación ha sido desarrollada para ser mantenida o modificada tanto en su lógica de proceso como en la estructura de datos.

Tabla 3.5 - Características Generales del Sistema

El manual provee reglas de clasificación para asignar los valores de 0 a 5 para cada una de las 14 GSC's, por ejemplo 0 si no existe y 5 si es el valor máximo, esta clasificación de 0 a 5 representa un conjunto ordenado dónde cada elemento desde el 0 hasta el 5 es considerado más grande que el anterior.

La evaluación del Grado de Influencia (Degree of Influence - DI) de cada una de estas 14 GSC's se realiza con base a juicio de experto. Al sumar estos 14 grados de influencia se obtiene el Grado Total de Influencia (TDI):

$$TDI = \sum DI$$

$$VAF = (TDI \times 0.01) + 0.65$$

El último paso del método de medición de IFPUG consiste en calcular los Puntos de Función Ajustados (Adjusted Function Points – AFP) mediante la siguiente formula:

$$AFP = VAF \times UPF$$

Consideraciones importantes del método de medición:

- El método de IFPUG mide puntos de función, pero ¿Qué es un punto de función?, es decir un metro describe una misma distancia siempre, un punto de función varía ya que se obtiene de distintos orígenes (ILF, ELF, EI, EO, EQ) con diferentes complejidades.
- Podemos observar que las tablas de peso, en cualquiera de las variables el último rango está abierto, esto significa que un software que tenga más elementos funcionales del último valor del rango no importa cuantos más sean (1 o 1,000) el valor será el mismo. Esto no refleja adecuadamente la realidad, ya que si se está midiendo un software con 1 DET más o con 1,000 DET's más el resultado sería el mismo.
- El estándar ISO/IEC 14143 no incluye el uso del factor de ajuste, esto es porque hay estudios que han demostrado que esto solo sobre estima el valor obtenido. Por otro lado, y quizá más importante es que el tamaño funcional de un software [20] no tiene que ver con las características de ambiente, calidad, desempeño, experiencia o el proceso de desarrollo en el cual se realiza dicho software.

3.3. ISO/IEC 19761:2011 - COSMIC FSMM [11]

3.3.1. Introducción al método de medición COSMIC

En 1999 COSMIC publicó la primera versión oficial del método de medición COSMIC-FFP v2.0, iniciando la segunda generación de métodos de medición de tamaño funcional. En el 2003, la versión 2.2 fue publicada y reconocida como estándar mediante la norma ISO/IEC 19761. Durante el 2007 se publicó la versión 3.0 de COSMIC, que se diferencia de la anterior porque agrega una fase para definir la estrategia de medición, la cual contempla la identificación de los usuarios funcionales de la aplicación, dejando de lado la relevancia del punto de vista desde donde se realiza la medición.

COSMIC fue diseñado para dar cumplimiento al estándar ISO/IEC 14143, ha sido diseñado con la industria y por la industria. El método COSMIC está diseñado para ser aplicable para medir la funcionalidad del software en los siguientes dominios funcionales:

- Aplicaciones Software de Gestión, las cuales son comúnmente utilizadas para la administración de negocios.
- Software en tiempo real, el cual interacciona con su entorno físico y mantiene o controla acontecimientos que están sucediendo en el mundo real.
- Híbridos de los anteriores, tales como controladores de dispositivos.
- Algunos tipos de software científico / ingeniería

COSMIC no ha sido diseñado para medir el tamaño funcional de una pieza de software, o de sus partes que:

- Se caracterizan por algoritmos matemáticos complejos u otras normas específicas complejas.
- Procesos de variables continuas tales como sonidos de audio o imágenes de video, como pueden encontrarse en un juego de computadora, instrumentos musicales y similares.

Sin embargo, se han realizado mediciones en estos ámbitos obteniendo buenos resultados.

El tamaño funcional medido por el método COSMIC está diseñado para depender solo de los FUR del software que va a ser medido y ser independiente de cualquier requerimiento o restricciones relativas a la aplicación de los FUR.

El método COSMIC se basa en sólidos principios de Ingeniería de Software. Estos principios se resumen en dos modelos, el *Modelo Genérico de Software* y el *Modelo Contextual de Software*.

El **Modelo Contextual de Software** se aplica para que el software que se va a medir y la medición requerida sean definidos sin ambigüedad, con ello se garantiza que los resultados pueden ser entendidos e interpretados consistentemente.

Los principios del **Modelo Genérico de Software** definen cómo los FUR del software que se va a medir deben ser modelados para que puedan ser medidos.

3.3.2. El proceso de medición COSMIC

El proceso de medición COSMIC se compone de tres fases (Figura 3.5):

1. La *Fase de Estrategia de Medición*, en la cual el propósito y alcance de la medición es definido, aplicando los principios del Modelo Contextual de Software.
2. La *Fase de Representación de la Medición*, en la cual se aplican los principios del Modelo Genérico de Software
3. La *Fase de Medición*, en la que se miden los tamaños reales.

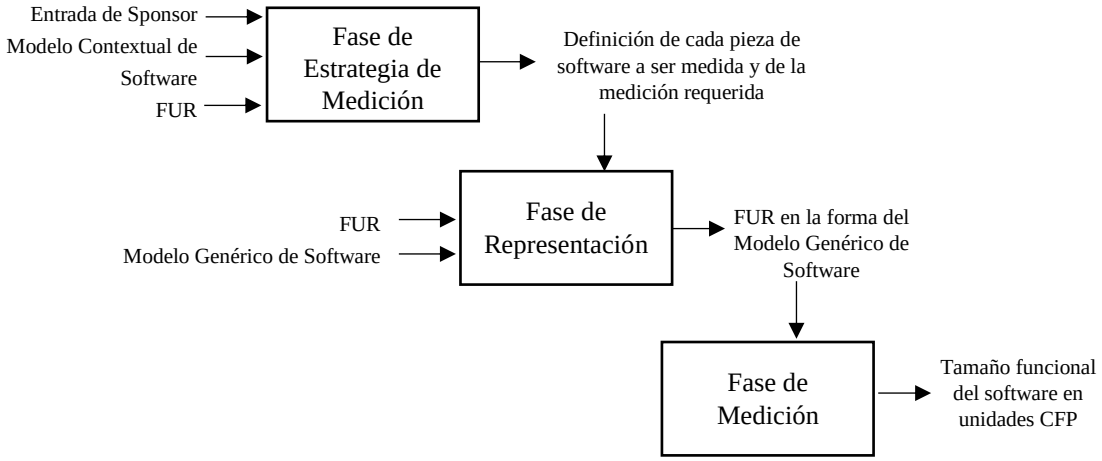


Figura 3.5 - El proceso de medición de COSMIC

3.3.3. Fase de estrategia de medición

En esta fase se describen los parámetros clave que deben ser considerados antes de realmente comenzar a medir. Estos son el *propósito* de la medición, el *alcance*, los *usuarios funcionales* y el *nivel de granularidad*. La determinación de estos parámetros ayuda a responder a las preguntas ¿qué tamaño debe medirse?, ¿qué tan precisa queremos la medición?, etc.

La Figura 3.6 muestra que la determinación de los parámetros puede necesitar alguna iteración.

> *Definición*. El **propósito de la medición** es una declaración que define por qué una medición es necesaria, y para qué se utilizará el resultado.

El medidor de una pieza de software debe decidir:

- *Cuándo* medir, antes, durante o después del desarrollo.
- *Qué* medir, por ejemplo, todo el software que se entrega en un proyecto, o excluir el software reutilizado.
- *Cuáles artefactos* utilizar para derivar los FUR a ser medidos, por ejemplo, una especificación de requerimientos.

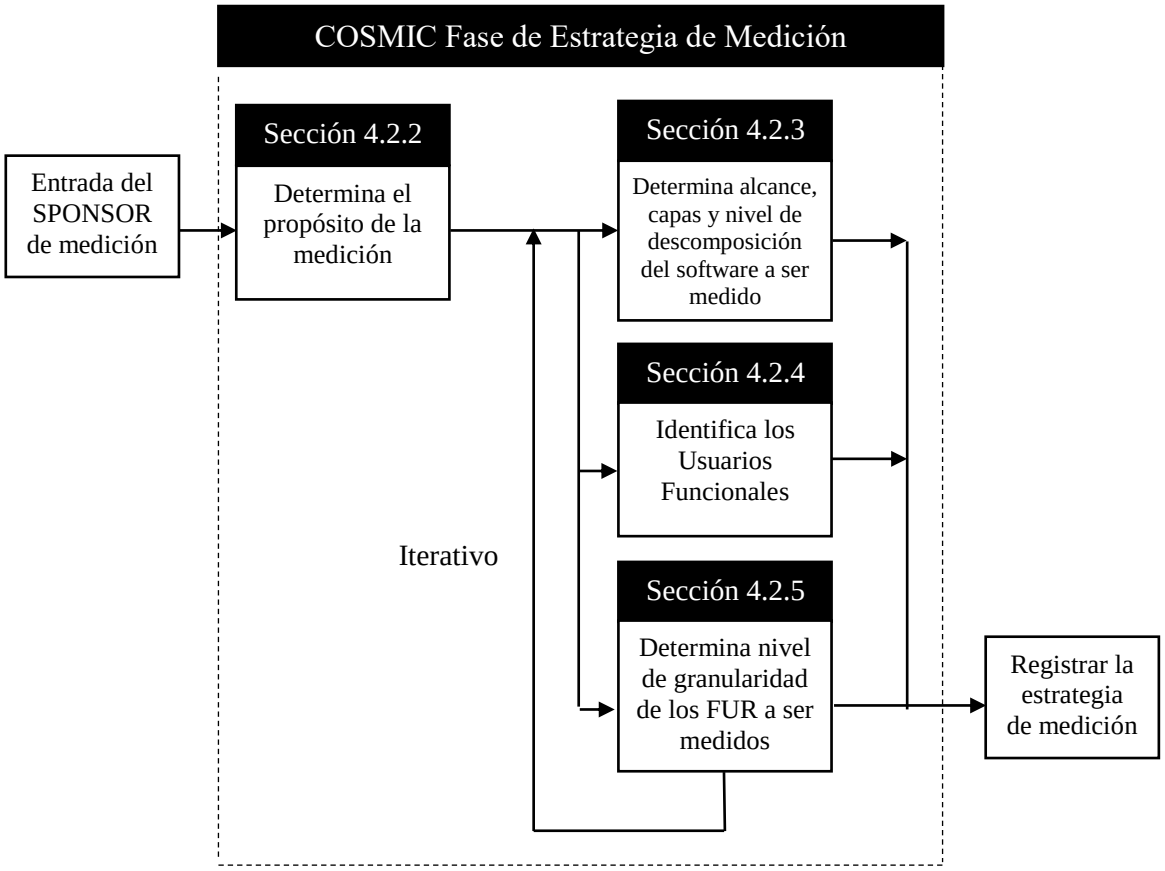


Figura 3.6 - El proceso para determinar una estrategia de medición

El propósito ayuda a los medidores a determinar:

- El *alcance* que debe medirse y por lo tanto los aparatos necesarios para la medición.
- Los *usuarios funcionales*.
- El momento en el ciclo de vida del proyecto en el que la medición se llevará a cabo.
- La precisión necesaria de la medición.

Definición. El **alcance de la medición** es el conjunto de los FUR que deben incluirse en un determinado ejercicio de medición de tamaño funcional.

Reglas del alcance de la medición:

1. El alcance de cualquier pieza de software a ser medida se deriva del propósito de la medición.
2. El alcance de cualquier medición no se extenderá por más de una capa de software que se desea medir.

Definición. Una **capa** es una partición funcional de una arquitectura de un sistema de software.

En una arquitectura de software definida, cada capa debe cumplir con los siguientes principios:

1. El software en una capa proporciona un conjunto de servicios que es coherente de acuerdo con algún criterio definido, y ese software puede utilizarse en otras capas sin saber cómo se implementan estos servicios.
2. La relación entre el software en cualquiera de sus capas se define por una regla de correspondencia que puede ser:
    a. *jerárquica*, es decir, el software en la capa A tiene permitido utilizar los servicios proporcionados por el software en la capa B, pero no al revés.
    b. *bidireccional*, es decir, el software en la capa A tiene permitido utilizar el software en la capa B, y viceversa.
3. El Software en una capa intercambia grupos de datos con el software en otra capa a través de sus respectivos procesos funcionales.
4. El Software en una capa no necesariamente utiliza todos los servicios funcionales proporcionados por un software en otra capa.
5. El Software en una capa de una arquitectura de software definida puede ser dividido en otras capas de acuerdo a diferentes arquitecturas de software definidas.

Una medición puede referirse a dos o más piezas *semejantes* de software, que se definen de la siguiente manera:

> *Definición*. Dos piezas de software son **semejantes** una a la otra si se encuentran en la misma capa.

> *Definición*. El **nivel de descomposición** de una pieza de software es cualquier nivel resultante de dividir una pieza de software en componentes, dividir dichos componentes en subcomponentes, y después dividir estos subcomponentes en sub-sub componentes, etc.

> *Definición*. Un **usuario funcional** es un tipo de usuario que es un emisor y/o destinatario de los datos de los FUR de una pieza de software.

La identificación de usuarios funcionales tiene las siguientes reglas:

1. Los usuarios funcionales de una pieza de software a ser medida, deben ser derivados del propósito de la medición.
2. Para los usuarios funcionales que son funcionalmente idénticos según los FUR, se identifica un solo tipo de usuario funcional.
3. Cuando el propósito de una medición de una pieza de software está relacionado con el esfuerzo para desarrollar o modificar el software, entonces los usuarios funcionales deben ser todos aquellos emisores y/o receptores de datos hacia/desde la nueva funcionalidad o la modificada, como es requerido por sus FUR.

Después de haber identificado los usuarios funcionales, es entonces fácil de identificar la *frontera*.

> *Definición*. La **frontera** es una interfaz conceptual entre el software que se está midiendo y sus usuarios funcionales.

Definición. Un **almacén persistente** es un almacén que permite a un proceso funcional almacenar un grupo de datos más allá de la vida del proceso funcional y/o del cual un proceso funcional puede recuperar un grupo de datos almacenado por otro proceso funcional, o almacenado por una ocurrencia anterior del mismo proceso funcional, o almacenado por otro proceso.

En el modelo COSMIC, el almacén persistente es un concepto que existe solamente dentro de la frontera del software que se está midiendo, no es considerado como un usuario funcional del software que se está midiendo (Figura 3.7).

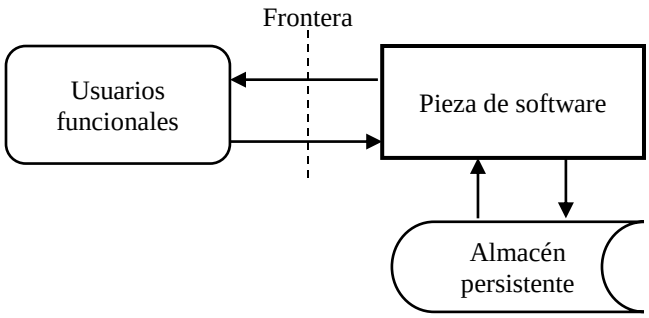


Figura 3.7 - Diagrama de contexto para la medición de una pieza de software

Definición. El **nivel de granularidad** es cualquier nivel de expansión de la descripción de una pieza de software de tal manera que, a cada aumento del nivel de expansión, la descripción de la funcionalidad de la pieza de software se encuentra en un nivel de detalle mayor y uniforme.

Una medición precisa de tamaño funcional de una pieza de software requiere que sus FUR sean conocidos a nivel de granularidad en la que se pueden identificar sus procesos funcionales y sus movimientos de datos.

Definición. El **nivel de granularidad de un proceso funcional** es un nivel de granularidad de la descripción de una pieza de software en el que los usuarios funcionales:

- Son seres humanos individuales o dispositivos de ingeniería o elementos de software (y no grupos de estos).
- Detectan ocurrencias únicas de eventos a los que la aplicación software debe responder (y no cualquier nivel en el cual se han definido grupos de eventos).

3.3.4. Fase de representación

La representación, segunda fase del proceso de medición (Figura 3.8), establece los pasos del proceso para mapear los FUR y los artefactos de software disponibles a la forma requerida por el Modelo Genérico de Software de COSMIC.

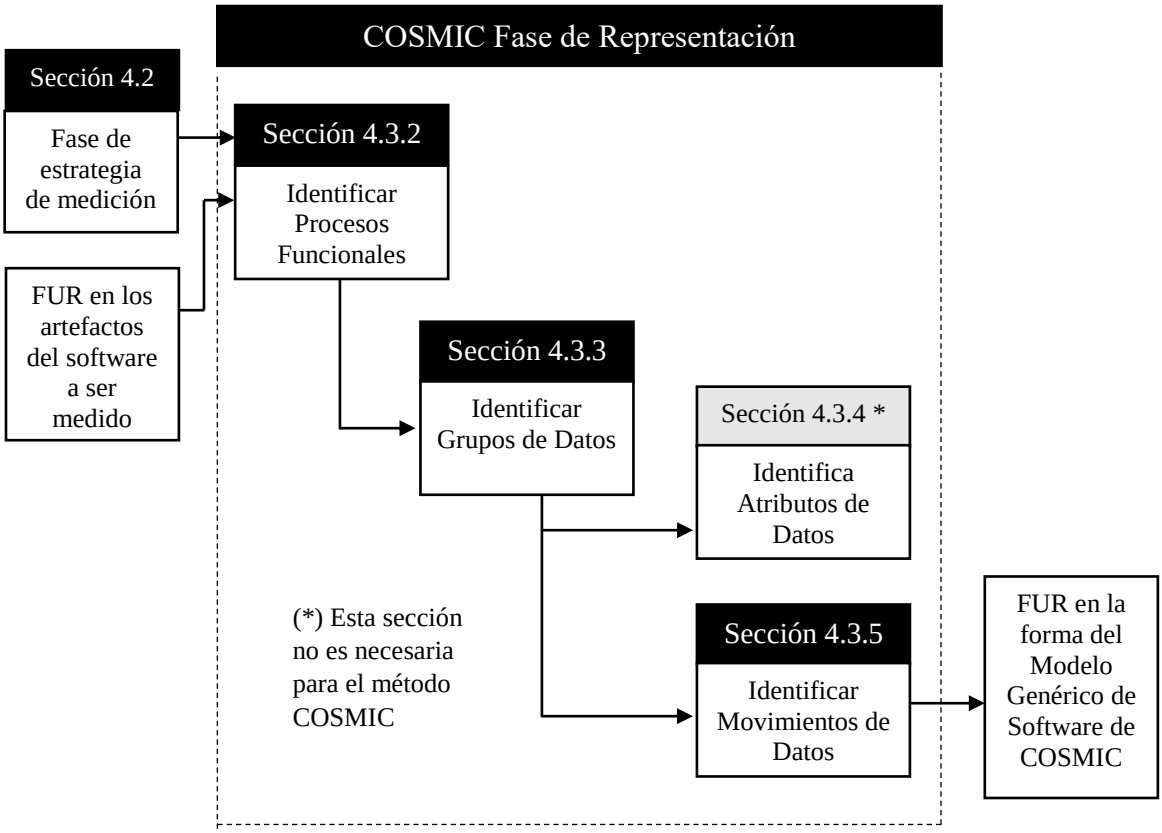


Figura 3.8 - Método general del proceso de representación COSMIC

Definición. Un **evento desencadenante** es un evento que genera que un usuario funcional del componente de software desencadene uno o más procesos funcionales.

Un evento desencadenante debe estar reconocido en los FUR del software que se está midiendo. Un evento hace que uno o más usuarios funcionales generen uno o más grupos de datos, cada uno de los cuales posteriormente serán movidos por una Entrada desencadenante. Un evento desencadenante no puede ser subdividido y sucede o no sucede.

Definición. Un **proceso funcional** es un conjunto de movimientos de datos, que representa una parte elemental de los FUR para el software que se está midiendo, que es único dentro de estos FUR y que se puede ser definido de manera independiente de cualquier otro proceso funcional en estos FUR.

- Un proceso funcional puede tener sólo una entrada desencadenante. Cada proceso funcional inicia el procesamiento con la recepción de un grupo de datos movidos por el movimiento de datos de una entrada desencadenante enviado por un usuario funcional.
- El conjunto de todos los movimientos de datos de un proceso funcional es el conjunto que se necesita para cumplir con los FUR para todas las posibles respuestas a la entrada desencadenante.

Definición. La **entrada desencadenante** es el movimiento de entrada de datos de un proceso funcional que mueve un grupo de datos generados por un usuario funcional que el proceso funcional necesita para iniciar el procesamiento.

La relación entre un evento desencadenante, el usuario funcional y el movimiento de datos de entrada que desencadena un proceso funcional se muestra en la Figura 3.9. La interpretación de esta figura es: un evento desencadenante ocasiona que un usuario funcional genere un grupo de datos que se mueve mediante una entrada desencadenante para iniciar el proceso funcional.

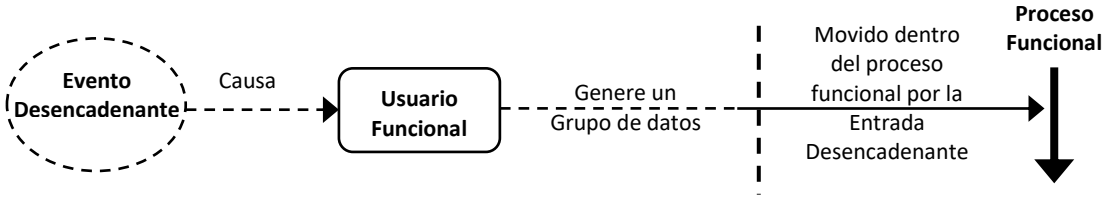


Figura 3.9 - Relaciones entre evento desencadenante, usuario funcional y proceso funcional

El proceso de identificación de los procesos funcionales, después de que los usuarios funcionales se han identificado y dados los FUR para el software que se está midiendo consisten en cuatro pasos:

1. Identificar en el mundo de los usuarios funcionales los eventos por separado a los que el software que se está midiendo debe responder (eventos desencadenantes) y que están en los FUR.
2. Identificar qué usuario(s) funcional(es) del software puede responder a cada evento desencadenante.
3. Identifique la entrada (o entradas) desencadenante(s) que cada usuario funcional puede iniciar en respuesta al evento.
4. Identificar el proceso funcional iniciado por cada entrada desencadenante.

Se utilizan las siguientes reglas para comprobar que los procesos funcionales candidatos han sido adecuadamente identificados:

1. Un proceso funcional pertenecerá íntegramente al alcance de la medición de una pieza de software en una, y sólo una, capa.
2. Cualquiera entrada desencadenante de una pieza de software que se está midiendo puede iniciar sólo un proceso funcional en ese software.
3. Un proceso funcional comprenderá al menos dos movimientos de datos, una Entrada y además una Salida o una Escritura. No hay límite superior para el número de movimientos de datos en un proceso funcional.
4. Un proceso funcional en ejecución se debe considerar terminado cuando se han satisfecho los FUR's para la respuesta a su entrada desencadenante. Una pausa durante la tramitación por razones técnicas no se considerará como la terminación del proceso funcional.

Definición. Un **objeto de interés** es cualquier cosa del mundo del usuario funcional que se identifica en los FUR, sobre la que se requiere que el software procese y/o

almacene datos. Puede ser cualquier cosa física, así como cualquier objeto conceptual o parte de un objeto conceptual.

Definición. Un **grupo de datos** es un conjunto único, no vacío, no ordenado y no redundante de atributos de datos donde cada atributo de datos incluido describe un aspecto complementario del mismo objeto de interés.

Una vez identificado, cada grupo de datos candidato debe cumplir con lo siguiente:

- Cada grupo de datos identificado deberá ser único y distinguible a través de su colección única de atributos de datos.
- Datos cualesquiera que aparezca en pantallas de entrada o salida o en informes y que no están relacionados con un objeto de interés para un usuario funcional no deberían ser identificados indicando un movimiento de datos, por tanto no deberían medirse.

En el método COSMIC no es obligatorio identificar los atributos de los datos. Sin embargo, la comprensión del término de *atributo de datos* es necesario para comprender los cambios en la medición, donde un FUR que cambia un atributo de datos puede resultar en un movimiento de datos debiendo ser medido.

Definición. Un **atributo de datos** es la pieza más pequeña de información, dentro de un grupo de datos identificados, que tiene un significado desde la perspectiva de los requerimientos funcionales del software.

Definición. Un **movimiento de datos** es un componente funcional base que mueve un único grupo de datos.

- Hay cuatro tipos de movimiento de datos.
- Cada tipo de movimiento de datos se considera que incluye ciertas manipulaciones de datos asociadas.

Definición. Una **Entrada (Entry – E)** es un movimiento de datos que mueve un grupo de datos desde un usuario funcional a través de la frontera hacia el proceso funcional donde se necesita.

Un candidato a movimiento de datos de entrada debe cumplir con los siguientes **principios**:

- Una Entrada deberá mover un único grupo de datos describiendo un solo objeto de interés de un usuario funcional a través de la frontera y hacia un proceso funcional del que la entrada forma parte. Si la entrada a un proceso funcional comprende más de un grupo de datos, cada uno describiendo un objeto de interés diferente, se identifica una entrada por cada grupo de datos que entra.
- Una Entrada no deberá sacar datos a través de la frontera, ni leer ni escribir datos del/hacia el almacenamiento persistente.

Las siguientes **reglas** ayudan a confirmar la condición de un candidato a movimiento de datos de entrada:

1. El grupo de datos de una entrada desencadenante puede constar de sólo un atributo de datos que simplemente informa al software que un evento ha ocurrido.
2. Las marcas de tiempo (ticks) de reloj que están desencadenando eventos serán siempre externos al software que está siendo medido. Se debe notar que no hay ninguna diferencia si el evento desencadenante es generado periódicamente por el hardware o por otra parte del software fuera de los límites del software medido.
3. Salvo que sea necesario en un proceso funcional especifico, obtener el tiempo desde el reloj del sistema, este no será considerado como causa de una entrada.
4. Si una ocurrencia de un evento específico desencadena la entrada de un grupo de datos que comprende hasta 'n' atributos de un objeto de interés en particular y los FUR permiten que otras ocurrencias del mismo evento puedan desencadenar una entrada de un grupo de datos que tiene valores de atributos de sólo un subconjunto de 'n' atributos del objeto de interés, entonces será identificada una entrada, compuesto por todos 'n' atributos.
5. Cuando se identifican las entradas en una pantalla que permite a los usuarios funcionales humanos introducir los datos de entrada en los procesos funcionales, se debe analizar sólo las pantallas que están llenos de datos. Ignore cualquier pantalla con formato pero de otra manera 'en blanco' a excepción de posibles valores por defecto, y no hacer caso de todos los campos y otros encabezados que permiten a los usuarios humanos comprender los datos de entrada requeridos.

Definición. Una **Salida (Exit – X)** es un movimiento de datos que mueve un grupo de datos desde un proceso funcional a través de la frontera hacia el usuario funcional que los requiere.

Un candidato a movimiento de datos de salida debe cumplir con los siguientes **principios**:

- Una Salida deberá mover un único grupo de datos describiendo un solo objeto de interés desde el proceso funcional del que la salida forma parte a través de la frontera hacia un usuario funcional. Si la salida de un proceso funcional comprende más de un grupo de datos, hay que identificar una salida para cada grupo de datos que sale.
- Una Salida no introducirá datos a través de la frontera, ni leer ni escribir datos del/hacia el almacenamiento persistente.

Las siguientes **reglas** pueden ser útiles para confirmar el estado de un movimiento de datos de salida candidato:

1. Una consulta que emite el texto fijo se modela como que tiene una sola salida por la salida de texto fijo.
2. Si la salida de un proceso funcional mueve un grupo de datos que comprende hasta 'n' atributos de datos de un objeto particular de interés y los FUR permiten que el proceso funcional pueda tener una ocurrencia de una salida que mueve un grupo de datos que tiene valores para un sub-conjunto de solamente 'n' atributos del objeto de interés, entonces se identifica una salida, que comprende todos los 'n' atributos de datos.
3. Cuando se hace la identificación de salidas, ignorar todos los campos y encabezados que permiten a los usuarios humanos comprender los datos de salida.

Definición. Un **mensaje de error/confirmación** es una salida emitida por un proceso funcional a un usuario humano funcional que, o bien confirma solamente que los datos han sido aceptados, o solamente que hay un error en los datos introducidos.

Cualquier Salida que incluya indicaciones de errores, pero que no está destinado a un usuario funcional humano, no es considerada como un mensaje de error/confirmación. Los mensajes de error/confirmación tienen las siguientes **reglas**:

1. Una salida se identificará para contabilizar todos los tipos de mensajes de error/confirmación emitidos por un proceso funcional del software que se mide por todas las causas posibles de acuerdo a su FUR. Si los FUR del proceso funcional no requieren ningún tipo de mensaje de error/confirmación a emitir, no identificar ninguna salida correspondiente.
2. Si un mensaje a un usuario funcional humano proporciona datos además de confirmar que los datos introducidos han sido aceptados, o que los datos introducidos es por error, entonces estos datos adicionales deben ser identificados como un grupo de datos movido por una salida, además de la salida de error/confirmación.
3. Todos los demás datos, emitidos o recibidos por el software que se mide, hacia/desde su hardware o usuarios funcionales software deben ser analizados de acuerdo con los FUR como salidas o entradas, respectivamente, de acuerdo con las reglas normales COSMIC, independientemente de si o no los valores de los datos indican una condición de error.
4. Las lecturas y escrituras se consideran que incluyen cualquier informe de condiciones de error. Por lo tanto ninguna entrada al proceso funcional que sea mide se debe identificar para cualquier indicación de error recibido como resultado de una lectura o escritura de datos persistentes.

Definición. Una **Lectura (Read – R)** es un movimiento de datos que mueve un grupo de datos desde un almacén persistente en el proceso funcional que los requiere.

Un candidato a movimiento de datos de lectura debe cumplir con los siguientes **principios**:

- Una lectura deberá mover un único grupo de datos describiendo un solo objeto de interés del almacén persistente a un proceso funcional del cual la lectura forma parte. Si el proceso funcional debe recuperar más de un grupo de datos del almacén, se debe identificar una lectura para cada grupo de datos que es recuperado.
- Una lectura no recibirá ni sacará datos a través de la frontera ni escribirá datos al almacenamiento persistente.
- Durante un proceso funcional, el movimiento o manipulación de constantes o variables que son internas del proceso funcional y que sólo se pueden cambiar por un programador, o mediante los resultados intermedios en un cálculo, o de los datos almacenados en un proceso funcional sólo resultantes de la aplicación, más que de los FUR, no se considerará como un movimiento de datos de lectura.

- Una lectura siempre incluye cualquier funcionalidad de solicitud de lectura (así, un movimiento de datos separado nunca será contado para cualquier funcionalidad de solicitud de lectura).

Las siguientes **reglas** pueden ser útiles para confirmar el candidato como movimiento de datos de lectura:

1. Identifica una lectura cuando, según los FUR, el software que se está midiendo debe recuperar un grupo de datos desde el almacenamiento persistente.
2. No identificar una lectura cuando los FUR del software que se está midiendo especifican algún usuario funcional de software o hardware como la fuente de un grupo de datos, o como los medios de recuperación de un grupo de datos almacenados.

Definición. Una **Escritura (Write – W)** es un movimiento de datos que mueve un grupo de datos a un almacén persistente que se encuentra dentro de un proceso funcional.

Un candidato como movimiento de datos de escritura debe cumplir con los siguientes **principios**:

- Una Escritura deberá mover un único grupo de datos describiendo un solo objeto de interés del proceso funcional del que la escritura forma parte hacia el almacén persistente. Si el proceso funcional debe pasar más de un grupo de datos al almacén persistente, identificar una escritura por cada grupo de datos que se mueve al almacén persistente.
- Una escritura no recibirá ni sacará datos de la frontera, ni leerá los datos.
- Un requerimiento para borrar un grupo de datos de un almacén persistente se medirá como una sola Escritura.
- Lo siguiente no podrá considerarse como movimientos de datos de Escritura:
  - El movimiento o manipulación de los datos que no existía en el inicio de un proceso funcional y que no se ha hecho persistente cuando el proceso funcional es completado.
  - Creación o actualización de variables o resultados intermedios que son internos al proceso funcional.
  - Almacenamiento de los datos por un proceso funcional resultante sólo de la aplicación, en lugar de que estén establecidos en los FUR.

Las siguientes **reglas** pueden ser útiles para confirmar el estado de un movimiento de datos de escritura candidato:

- Identificar una escritura en que, de acuerdo a los FUR, el software que se está midiendo debe mover un grupo de datos hacia el almacenamiento persistente.
- No identificar una escritura cuando los FUR del software que se está midiendo especifica algún usuario funcional de software o hardware como el destino del grupo de datos o como medio de almacenamiento del grupo de datos.

La Figura 3.10 muestra los cuatro tipos de movimientos de datos (entrada, salida, lectura y escritura) y su relación con los usuarios funcionales (humanos, dispositivos de hardware u otro software) y el almacenamiento persistente.

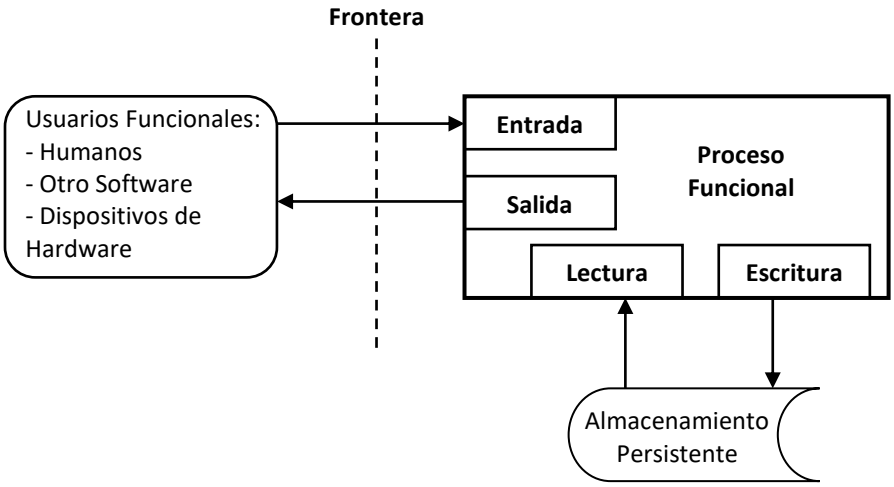


Figura 3.10 - Los cuatro tipos de movimientos de datos y su relación con los usuarios funcionales y el almacenamiento persistente.

> *Definición*. La **manipulación de datos** es todo lo que le sucede a los datos, distinto de un movimiento de datos en o de un proceso funcional, o entre un proceso funcional y el almacén persistente.

Todas las manipulaciones de datos en un proceso funcional serán asociadas con los cuatro tipos de movimiento datos (E, X, R, y W).

> *Definición*. Un **comando de control** es un comando que le permite al usuario funcional humano controlar el uso del software pero que no involucra ningún movimiento de datos sobre el objeto de interés del software que se está midiendo.

Un comando de control no es un movimiento de datos porque el comando no mueve datos acerca de un objeto de interés. Solo en aplicaciones con interfaces de usuario existen comandos de control, los cuales serán ignorados ya que no implican ningún movimiento de datos sobre un objeto de interés. Ejemplos de ello son comandos de ‘página arriba / abajo, presionar la tecla Tab o tecla Enter, hacer clic 'OK' para confirmar una acción anterior, al pulsar un botón para continuar, etc.

3.3.5. Fase de medición

> *Definición*. **Unidad de medida COSMIC**, 1 CFP (Punto de Función COSMIC) que se define como el tamaño de un movimiento de datos.

Los resultados de una medición COSMIC se deberían anotar como ‘x CFP (v.y)’, donde:

- ‘x’ representa el valor numérico del tamaño funcional.
- v.y representa la identificación de la versión estándar del método COSMIC usado para obtener el tamaño funcional.

Un **cambio funcional** de un software es interpretado en el método COSMIC como cualquier combinación de sumas de nuevos movimientos de datos, de modificaciones o eliminaciones de los movimientos de datos existentes incluyendo la manipulación de datos asociada. La necesidad de un cambio de software puede surgir de cualquier:

- Nuevo FUR (agregar funcionalidad)
- Desde un cambio en el FUR (agregar, modificar o eliminar funcionalidad)

Un movimiento de datos se considera modificado funcionalmente si al menos aplica una de las siguientes consideraciones:

- El grupo de datos movido es modificado
- La manipulación de datos asociada es modificada.

Un grupo de datos es modificado si al menos aplica una de las siguientes consideraciones:

- Uno o más atributos nuevos son agregados al grupo de datos
- Uno o más atributos existentes son modificados o eliminados del grupo de datos

Al modificar un movimiento de datos se tienen en cuenta las siguientes reglas:

- Un cambio CFP se medirá, independientemente de la cantidad de modificaciones en el movimiento de datos.
- Si un grupo de datos debe ser modificado, los movimientos de datos moviendo el grupo de datos modificado, cuya funcionalidad no se ve afectada por la modificación del grupo de datos, no serán identificados como movimientos de datos modificados.

Los movimientos de datos modificados no tienen influencia sobre el tamaño funcional ya que estos existen tanto antes como después de que las modificaciones han sido realizadas.

Reglas generales de agregación de resultados de la medición:

a) Para cualquier proceso funcional, el tamaño funcional de cada movimiento de datos individual debe ser agregado en un único valor de tamaño funcional en unidades de CFP para luego sumar todos juntos.

$$Tamaño(proceso\ funcional_i) = \Sigma\ tamaño(Entradas_i) + \Sigma\ tamaño(Salidas_i) + \Sigma\ tamaño(Lecturas_i) + \Sigma\ tamaño(Escrituras_i)$$

b) El tamaño de una pieza de software dentro de alcance definido se obtendrá sumando los tamaños de sus procesos funcionales.

$$Tamaño(pieza\ software) = \Sigma\ tamaño(procesos\ funcionales)$$

c) Para cualquier proceso funcional, el tamaño funcional de los cambios en sus Requerimientos Funcionales de Usuario se sumarán al tamaño de movimientos de datos que han sido añadidos, modificados o eliminados en el proceso funcional para dar un tamaño del cambio en unidades de CFP.

$$Tamaño(Cambio(proceso\ funcional_i)) = \Sigma\ tamaño(movimientos\ datos\ añadidos_i) + \Sigma\ tamaño(movimientos\ datos\ modificados_i) + \Sigma\ tamaño(movimientos\ datos\ eliminados_i)$$

d) El tamaño de cualquier cambio en una pieza de software dentro de un alcance definido se obtendrá sumando los tamaños de todos los cambios de todos los procesos funcionales.

Tamaño(Cambios(pieza software)) = Σ tamaño(Cambios(procesos funcionales))

e) Después del cambio funcional de una pieza de software, su nuevo tamaño total es igual al tamaño original, más el tamaño funcional de todos los movimientos de datos agregados, menos el tamaño funcional de todos los movimientos de datos eliminados. Los movimientos de datos modificados no tienen influencia sobre el tamaño de la pieza de software ya que estos existen tanto antes como después de que las modificaciones han sido realizadas.

Tamaño(proceso funcionali) = Tamaño(proceso funcionali)
+ Σ tamaño(movimientos datos añadidosi)
– Σ tamaño(movimientos datos eliminadosi)

f) Los tamaños de piezas de software o de cambios de piezas de software podrán sumarse sólo si se mide al mismo nivel de granularidad de los FUR el proceso funcional.

g) Los tamaños de piezas de software y/o cambios en los tamaños de piezas de software dentro de una capa o de diferentes capas serán sumados sólo si tiene sentido hacerlo, a efectos de la medición.

h) El tamaño de una pieza de software se obtiene sumando los tamaños de sus componentes proporcionados
- el tamaño de las contribuciones de los movimientos de datos inter-componentes son eliminadas y
- sólo una Salida se identifica para todos los mensajes de error/confirmación emitidos por un proceso funcional a un usuario funcional humano.

i) Si el método COSMIC se extiende localmente, entonces el tamaño medido por la extensión local debe ser informado por separado y no puede ser añadido al tamaño obtenido por el método estándar, medido en CFP.

3.3.6. Extendiendo el método COSMIC

El método COSMIC fue diseñado para medir software rico en movimiento de datos. Al igual que todos los demás FSMM, COSMIC no fue diseñado para medir de forma explícita la funcionalidad de manipulación de datos. En cambio, el método asume que los tipos de movimiento de datos contabilizan la funcionalidad de manipulación de datos asociados. El método también no considera la influencia del número de atributos de datos por el movimiento de datos en el tamaño funcional de software. Cuando sea necesario extender el método COSMIC se debe tener en cuenta lo siguiente:

1. Cualquiera que desee refinar el método COSMIC al introducir una sub-unidad de medida es libre de hacerlo, pero debe dejar claro que el tamaño resultante medido no se expresa en el estándar de puntos de función COSMIC.

2. Cuando se necesita más precisión en la medición de los movimientos de datos, entonces se puede definir una sub-unidad de medida.
3. Si se considera necesario para tener en cuenta manipulación de datos asociada a los movimientos de datos, contabilizar el número de atributos de un grupo de datos, algoritmos complejos, entre otros, puede organizarse un estándar local para estas funcionalidades.
4. Un resultado de una medición COSMIC utilizando extensiones locales debería ser anotado como 'x CFP (v. y) + z Local FP', donde:
   - 'x' representa el valor numérico obtenido al agregar todos los resultados de medidas individuales de acuerdo al método estándar COSMIC.
   - 'v.y' representa la identificación de la versión estándar del método COSMIC usado para obtener el tamaño funcional.
   - 'z' representa el valor numérico obtenido al agregar todos los resultados individuales de medida obtenidos de las extensiones locales al método COSMIC.

3.4. Ejemplo: Caso de estudio "Administración de avisos de venta de vehículos"

3.4.1. Caso de uso: Iniciar Sesión

Flujo de eventos

- Flujo básico

Acción que realiza el usuario	Acción que realiza el sistema
1. Ingresa al módulo de inicio de sesión.	2. Muestra en la pantalla de inicio de sesión, un formulario para capturar la siguiente información: • Nombre de usuario • Contraseña
3. Captura el nombre de usuario y contraseña.	4. Valida el nombre de usuario y contraseña. En caso de que los datos sean incorrectos, muestra mensaje de error.
	5. Muestra la pantalla de inicio del usuario.
Fin del flujo básico	

3.4.2. Medición con COSMIC del Caso de uso: Iniciar sesión

Fase de estrategia:

- **Propósito:** determinar el tamaño funcional del caso de uso Iniciar Sesión como propósito de ejemplo.
- **Alcance:** global, ya que la medición considera toda la funcionalidad del caso de uso.
- Identificación de capas: se considera una sola capa global de aplicación para la medición del caso de uso.
- **Identificación de usuarios funcionales:** Se identifica un solo usuario funcional genérico, el cual interactúa con el sistema.
- **Nivel de granularidad**: de las descripciones textuales del caso de uso, se identifica un nivel de granularidad adecuado para realizar la medición.
- **Diagrama de contexto**:

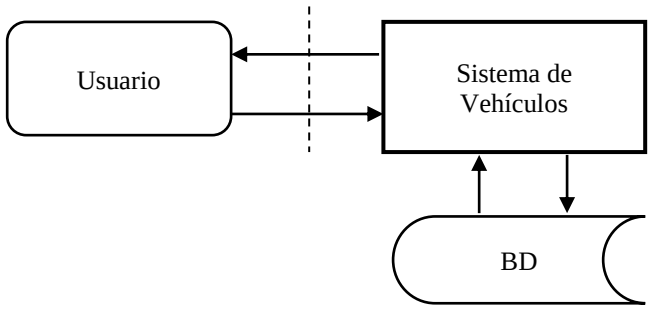


Diagrama 3.1- Diagrama de contexto del sistema

Fase de Representación:

- **Identificación de procesos funcionales**: de las descripciones textuales del caso de uso, se identifica un solo evento desencadenante donde un usuario desea ingresar al sistema, identificando un solo proceso funcional que atiende a este evento desencadenante.

Evento desencadenante	Proceso funcional
Usuario desea ingresar al sistema	Iniciar sesión

- **Identificación de objetos de interés y grupos de datos**: De las descripciones textuales del caso de uso, se identifica un solo objeto de interés llamado "Usuario" el cual tiene un grupo de datos llamado "Datos de usuario".

Objeto de interés	Grupos de Datos
Usuario	Datos de Usuario

- **Identificación de movimientos de datos** de los procesos funcionales: La identificación de movimientos de datos se describe en la siguiente tabla.

Proceso Funcional	**Evento Desencadenante**	**Descripción de Subprocesos**	**Grupo de Datos**	**Mov. de datos**	**CFP**	**Σ CFP**
Iniciar sesión	Usuario desea ingresar al sistema	Usuario ingresa al módulo de inicio de sesión.	N/A	N/A	0	
		Sistema muestra en la pantalla de inicio de sesión, un formulario para capturar la siguiente información: •Nombre de usuario •Contraseña	N/A	N/A	0	
		Usuario captura nombre y contraseña.	Datos de Usuario	E	1	
		Sistema valida nombre y contraseña.	Datos de Usuario	R	1	
		Sistema muestra la pantalla de inicio.	N/A	N/A	0	
		Sistema muestra mensaje de error en caso de que los datos sean incorrectos.	Mensajes de error/ confirmación	X	1	
						3
				Total:	**3 CFP v4.0.1**	

Fase de Medición:

La fase de medición consiste en la asignación de reglas numéricas, dicha asignación se muestra en la tabla presentada en la identificación de movimientos de datos de la fase de representación.

La siguiente tabla muestra el total de movimientos de datos, total por procesos funcionales y total por casos de uso.

Caso de Uso	Proceso Funcional	Entradas	Salidas	Lecturas	Escrituras	Total por PF	Total por CU
Iniciar Sesión	Iniciar Sesión	1	1	1	0	3	3
Total		1	1	1	0	3	3

3.4.3. Caso de uso: Administración de avisos de venta de vehículos

Flujo de eventos

- Flujo básico

Acción que realiza el usuario	Acción que realiza el sistema
1. Ingresa al menú principal y solicita la opción "Venta de vehículos".	2. Muestra pantalla con los siguientes campos de captura: • NIV • NRPV Muestra la opción • Buscar • Venta de Vehículo (Continúa con el flujo AO01 Registrar aviso de venta)
3. Ingresa alguno de los siguientes campos: • NIV • NRPV Y selecciona la opción Buscar	4. Valida la información capturada y muestra la siguiente información: Detalle del aviso de venta: • Fecha de movimiento • Tipo de movimiento (1 = venta) • Fecha de entrega o recepción • Factura (archivo adjunto) • NIV • NRPV • Número de Placa • Modelo • Marca • Número de puertas • Nombre completo del dueño • CURP • RFC • Teléfono Muestra las siguientes opciones: • Actualización (Continúa con el flujo AO02 Actualización de avisos de venta de vehículos).

	• Cancelación de venta (Continúa con el flujo AO03 Cancelación de avisos de venta de vehículos) Si el NIV o NRPV son inválidos muestra mensaje de error.
Fin del flujo básico	

- Flujo alterno AO01: Registrar aviso de venta

Acción que realiza el usuario	Acción que realiza el sistema
	1. Valida la información capturada y muestra la siguiente información: Detalle del vehículo: • NIV • NRPV • Número de Placa • Modelo • Marca • Número de puertas • Nombre completo del dueño • CURP • RFC • Teléfono Muestra los siguientes campos: • Fecha de movimiento (otorgado por el sistema = fecha actual) • Tipo de movimiento (otorgado por el sistema 1 = venta) • Fecha de entrega o recepción(campo de captura) • Factura (campo de selección de archivo) Muestra las siguientes opciones: • Registrar • Cancelar • Si el NIV o NRPV son inválidos muestra mensaje de error.
2. Ingresa la fecha y selecciona la factura para adjuntar. Selecciona la opción Registrar.	3. Valida que la fecha de entrega sea posterior a la fecha actual y que la factura esté en formato PDF. Si cumple con las validaciones guarda la información del aviso y muestra mensaje de confirmación, en caso contrario muestra un mensaje de error.
Fin del flujo alterno AO01	

- Flujo alterno AO02: Actualización de avisos de venta de vehículos

Acción que realiza el usuario	Acción que realiza el sistema
	1. Muestra pantalla con los siguientes campos: • Fecha de movimiento (otorgado por el sistema = fecha actual) • Tipo de movimiento (otorgado por el sistema 1 = venta) • Fecha de entrega o recepción (campo habilitado) • Factura (campo de selección de archivo) Muestra las siguientes opciones: • Actualizar • Cancelar
4. Modifica la fecha de entrega y/o selecciona la factura para adjuntar. Selecciona la opción Actualizar.	5. Valida que la fecha de entrega sea posterior a la fecha actual y que la factura esté en formato PDF. Si cumple con las validaciones actualiza la información del aviso y muestra mensaje de confirmación, en caso contrario muestra un mensaje de error.
Fin del flujo alterno AO02	

- Flujo alterno AO03: Cancelación de avisos de venta de vehículos

Acción que realiza el usuario	Acción que realiza el sistema
	1. Muestra el siguiente mensaje: "¿Deseas cancelar el aviso de venta del vehículo con NIV: *NIV del vehículo*?" Muestra las siguientes opciones: • Aceptar • Cancelar
2. Selecciona la opción Aceptar	3. Elimina el aviso de venta del vehículo y muestra el siguiente mensaje: "aviso de venta del vehículo con NIV: *NIV del vehículo* ha sido cancelado"
Fin del flujo alterno AO03	

3.4.4. Medición con COSMIC del Caso de uso: Administración de avisos de venta de vehículos

Fase de estrategia:

- **Propósito**: determinar el tamaño funcional del caso de uso Administrar avisos de venta de vehículos como propósito de ejemplo.
- **Alcance**: global ya que la medición considera toda la funcionalidad del caso de uso.
- **Identificación de capas**: se considera una sola capa global de aplicación para la medición del caso de uso.
- **Identificación de usuarios funcionales**: se identifica un solo usuario funcional genérico, el cual interactúa con el sistema.

- **Nivel de granularidad**: de las descripciones textuales del caso de uso, se identifica un nivel de granularidad adecuado para realizar la medición.
- **Diagrama de contexto**:

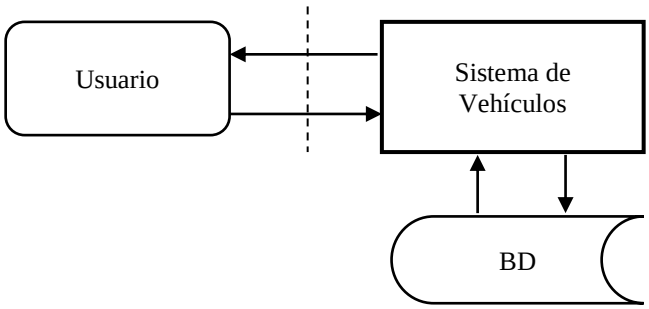


Diagrama 3.2- Diagrama de contexto del sistema

Fase de Representación:

- **Identificación de procesos funcionales**: de las descripciones textuales del caso de uso, se identifican cuatro eventos desencadenantes donde un usuario desea buscar, registrar, modificar y cancelar un aviso de venta de vehículo, identificando cuatro procesos funcionales que atienden a cada evento desencadenante.

Evento desencadenante	Proceso funcional
Usuario desea consultar la información de un aviso de venta	Buscar aviso de venta
Usuario desea registrar un aviso de venta	Registrar aviso de venta
Usuario desea actualizar la información de un aviso de venta	Actualizar aviso de venta
Usuario desea cancelar un aviso de venta	Cancelar aviso de venta

- **Identificación de objetos de interés y grupos de datos**: De las descripciones textuales del caso de uso: Administrar avisos de venta de vehículos, se identifican los siguientes objetos de interés y grupos de datos.

Objeto de interés	Grupos de Datos
Vehículo	Datos de Vehículo
Aviso de Venta	Datos de Aviso de Venta
Factura	Datos de Factura

- **Identificación de movimientos de datos** de los procesos funcionales: La identificación de movimientos de datos se describe en la siguiente tabla.

Proceso Funcional	Evento Desencadenante	Descripción de Subprocesos	Grupo de Datos	Mov. de datos	CFP	Σ CFP
Buscar aviso de venta	Usuario desea consultar la información de un aviso de venta	Usuario ingresa al menú principal y solicita la opción "Venta de vehículos"	N/A	N/A	0	
		Sistema muestra pantalla con los siguientes campos de captura: •NIV •NRPV	N/A	N/A	0	
		Usuario captura NIV o NRPV y selecciona Buscar	Datos de Vehículo	E	1	
		Sistema valida la información capturada y muestra el detalle del aviso de venta, del vehículo y del dueño del vehículo.	Datos de Aviso de Venta, Datos de Factura	R	2	
			Datos de Aviso de Venta, Datos de Factura	X	2	
		Si el NIV o NRPV son inválidos el sistema muestra mensaje de error.	Mensajes de error/ confirmación	X	1	
						6
Registrar aviso de venta	Usuario desea registrar un aviso de venta	Usuario ingresa al menú principal	N/A	N/A	0	
		Sistema muestra pantalla con los siguientes campos de captura: •NIV •NRPV	N/A	N/A	0	
		Usuario captura NIV o NRPV y selecciona Venta de Vehículos	Datos de Vehículo	E	1	
		Sistema valida la información capturada y muestra el detalle del vehículo y del dueño del vehículo.	Datos de Vehículo	R	1	
			Datos de Vehículo	X	1	
		Sistema Muestra los campos: •Fecha de movimiento (otorgado por el sistema) •Tipo de movimiento (otorgado por el sistema) •Fecha de entrega o recepción (campo de captura)	Datos de Aviso de Venta	X	1	

Proceso Funcional	Evento Desencadenante	Descripción de Subprocesos	Grupo de Datos	Mov. de datos	CFP	Σ CFP
		•Factura (campo de selección de archivo)				
		Usuario ingresa la fecha y selecciona la factura para adjuntar. Selecciona la opción Registrar.	Datos de Aviso de Venta, Datos de Factura	E	2	
		Sistema valida que la fecha de entrega sea posterior a la fecha actual y que la factura esté en formato PDF. Si cumple con las validaciones guarda la información del aviso.	Datos de Aviso de Venta, Datos de Factura	W	2	
		Si el NIV o NRPV son inválidos, o si la fecha o el formato del PDF son inválidos el sistema muestra mensaje de error.	Mensajes de error/ confirmación	X	1	
						9
Actualizar aviso de venta	Usuario desea actualizar la información de un aviso de venta	Usuario selecciona la opción Actualización	Comando	E	1	
		Sistema Muestra los siguientes campos: •Fecha de movimiento (otorgado por el sistema) •Tipo de movimiento (otorgado por el sistema) •Fecha de entrega o recepción (campo habilitado) •Factura (campo de selección de archivo)	Datos de Aviso de Venta	X	1	
		Usuario ingresa la fecha y/o selecciona la factura para adjuntar. Selecciona la opción Actualizar.	Datos de Aviso de Venta, Datos de Factura	E	2	
		Sistema valida que la fecha de entrega sea posterior a la fecha actual y que la factura esté en formato PDF. Si cumple con las validaciones guarda la información del aviso.	Datos de Aviso de Venta, Datos de Factura	W	2	

Proceso Funcional	Evento Desencadenante	Descripción de Subprocesos	Grupo de Datos	Mov. de datos	CFP	Σ CFP
		Si la fecha o el formato del PDF son inválidos el sistema muestra mensaje de error.	Mensajes de error/ confirmación	X	1	
						7
Cancelar aviso de venta	Usuario desea cancelar un aviso de venta	Usuario selecciona la opción Cancelar aviso de venta	Comando	E	1	
		Sistema muestra el siguiente mensaje: "¿Deseas cancelar el aviso de venta del vehículo con NIV: NIV del vehículo?"	Datos de Vehículo	X	1	
		Usuario selecciona aceptar	N/A	N/A	0	
		Sistema elimina el aviso de venta del vehículo	Datos de Aviso de Venta	W	1	
		Sistema muestra el siguiente mensaje: "aviso de venta del vehículo con NIV: NIV del vehículo ha sido cancelado"	Datos de Vehículo	X	1	
						4
				Total:	**26 CFP v4.0.1**	

Fase de Medición

La fase de medición consiste en la asignación de reglas numéricas, dicha asignación se muestra en la tabla presentada en la identificación de movimientos de datos de la fase de representación.

La siguiente tabla muestra el total de movimientos de datos, total por procesos funcionales y total por casos de uso.

Caso de Uso	Proceso Funcional	Entradas	Salidas	Lecturas	Escrituras	Total por PF	Total por CU
Administrar avisos de ventas	Buscar aviso de venta	1	3	2	0	6	26
	Registrar aviso de venta	3	3	1	2	9	
	Actualizar aviso de venta	3	2	0	2	7	
	Cancelar aviso de venta	1	2	0	1	4	
Total		8	10	3	5	26	26

3.5. Ejemplo: Medición de un aplicativo con distintos métodos de medición.

3.5.1. Caso de estudio: Compra-Venta de Autos

Una empresa de compra-venta de autos usados, ha decidido ofrecer en la red el servicio de venta de autos. Para ello ha pedido a una empresa desarrolladora de software, la ejecución del proyecto. De antemano, la empresa desarrolladora, prevé que los requerimientos y restricciones funcionales, serán de complejidad promedio y el desarrollo será en C#.

Como parte de los requerimientos funcionales, el cliente ha indicado lo siguiente:

1. Un posible comprador de vehículo podrá acceder a la página Web del distribuidor para conocer el precio y comprar un auto del inventario.
2. La aplicación Web utilizará la información que ya se mantiene dentro del Sistema de Inventario de Vehículos (SIV). Este sistema mantiene los vehículos y los posibles accesorios de cada uno de ellos. El modelo de datos que maneja es el siguiente:

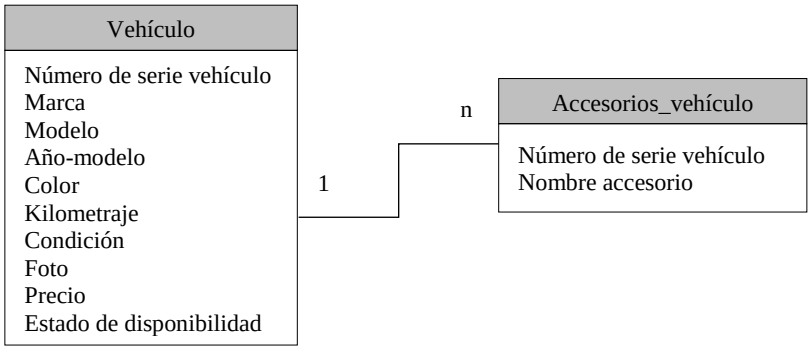
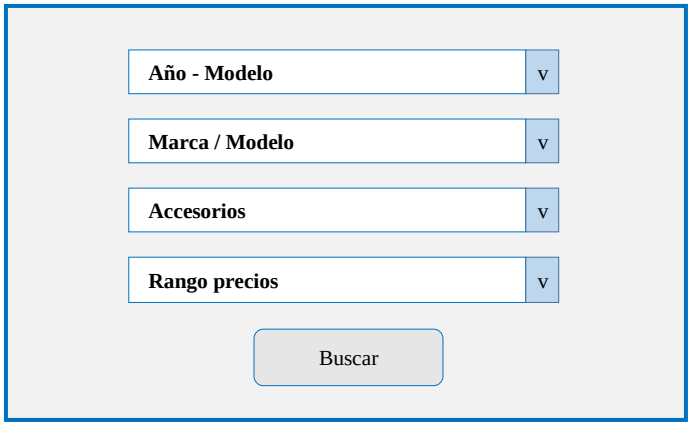


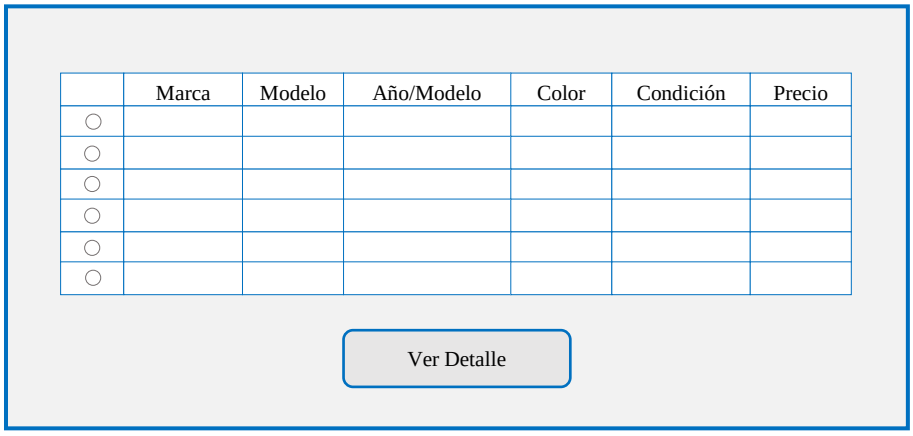
Diagrama 3.3 - Modelo de datos SIV

3. Un comprador potencial, podrá buscar un vehículo de dos formas: podrá ver todo el inventario, o filtrar con base a una serie de características de los vehículos. La pantalla de consulta será de la forma siguiente:



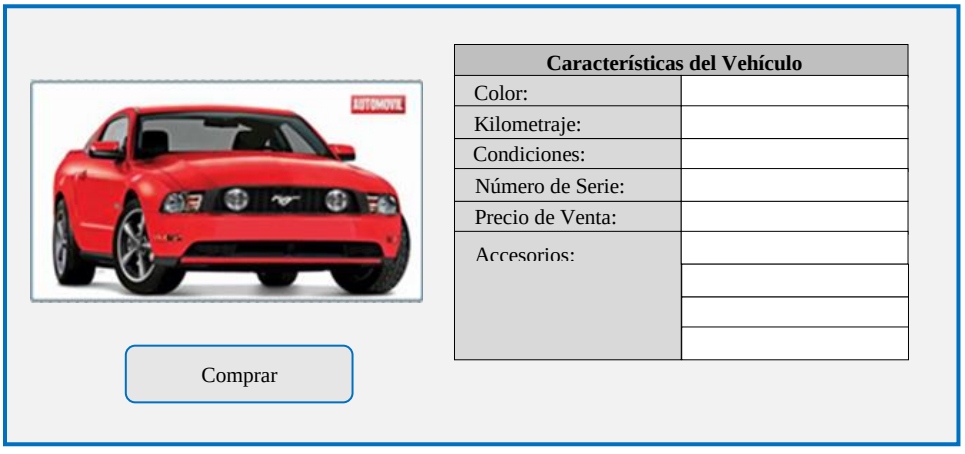
Pantalla 3.1 - Parámetros de consulta

4. Los drop-downs del Año-Modelo, Marca/Modelo y Accesorios, serán llenados a partir de la información almacenada en el Sistema de Inventario de Vehículos. Adicionalmente, el drop-down de "Rango de Precios" ofrecerá 5 rangos de precio. Cada uno será calculado a partir del valor mínimo y máximo del precio de los autos que se tienen en inventario, y luego dichos valores serán concatenados para mostrarse de la manera siguiente:
    - Entre 50,000 y 100,00
    - Entre 100,001 y 150,000
    - Entre 150,001 y 200,000
    - …
5. Con base en los parámetros de consulta, será mostrado un listado con los vehículos disponibles en el inventario. Si no hay vehículos, el mensaje "No hay vehículos con esas características", se desplegará.



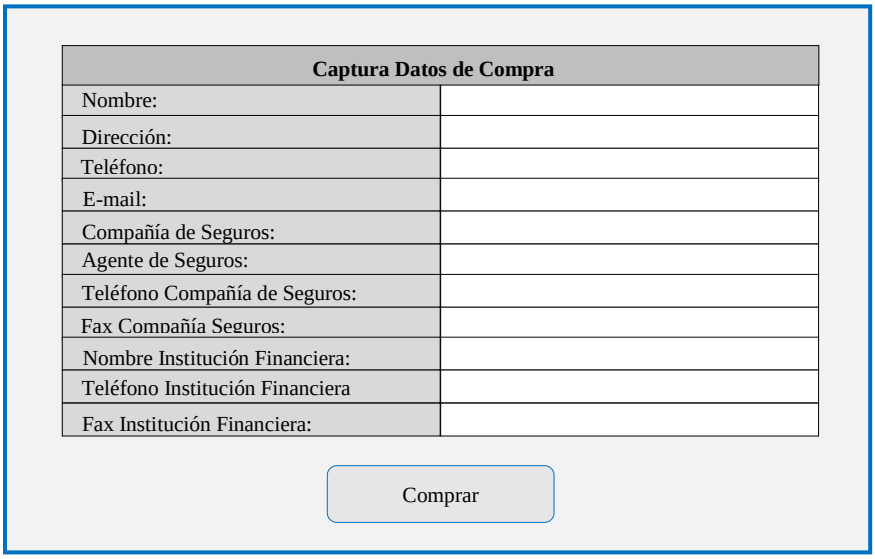
Pantalla 3.2 - Listado de vehículos

6. El comprador podrá entonces verificar el listado, y si se ve un vehículo que le llame la atención, podrá seleccionarlo y hacer una consulta detallada, la cual se mostrará en la pantalla siguiente:



Pantalla 3.3 - Detalle del vehículo

7. Si el posible comprador, decide comprar el vehículo, entonces presionará el botón "Comprar", que lo llevará a la siguiente pantalla, en dónde registrará sus datos, los de su aseguradora y los de su institución financiera. Tal y como se muestra en la pantalla siguiente:



Pantalla 3.4 - Captura de datos para compra

8. Al término de esta operación, presionará el botón: "Comprar", lo cual hará que el auto seleccionado quede como "comprado" en el inventario de vehículos. La información de compra se guardará en el almacenamiento de compras, y por último, se enviará un correo de confirmación al comprador, informando: el número de compra, nombre del comprador, nombre de la aseguradora, nombre de la financiera, marca del vehículo comprado, modelo, año-modelo, color, accesorios y precio del vehículo. Los datos quedarán almacenados de la siguiente manera:

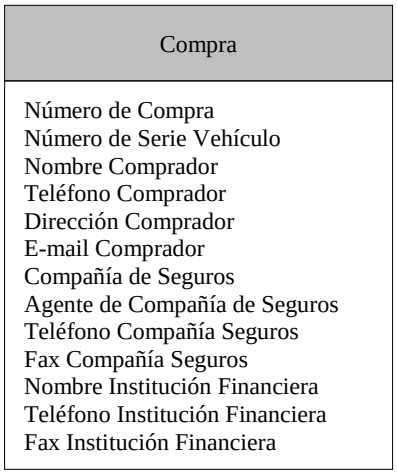


Diagrama 3.4 - Datos de compra

3.5.2. Medición con IFPUG del caso de estudio: Compra-Venta de Autos

Cálculo de FP no ajustados (UFP)

- **Tipo de conteo**: Nuevo Desarrollo
- **Propósito**: Determinar el conteo para posteriormente calcular el esfuerzo y costo del proyecto.
- **Frontera del sistema**:

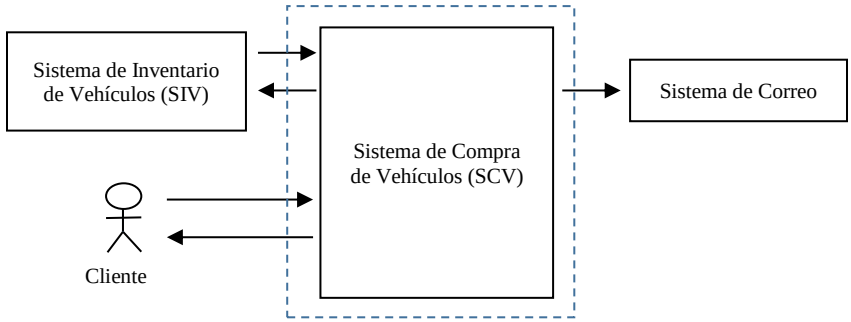


Diagrama 3.5 - Frontera del sistema

- **Identificación de FTR´s**

SIV	- Vehículo - Accesorio_vehículo	EIF
SCV	- Compra	ILF
SCorreo	- Correo	ILF

  - Complejidad EIF
    - Vehículo (Numero serie, Marca, Modelo, Año-modelo, Color, Kilometraje, Condición, Foto, Precio, Disponibilidad, Nombre accesorio)
      - Para 1 RET y 11 DET’s la complejidad es Baja.

  - Complejidad ILF
    - Compra (Número compra, Número serie, Nombre comprador, Dirección comprador, Tel comprador, E mail, Compañía seguros, Agente compañía seguros, Teléfono compañía seguros, Fax compañía seguros, Nombre inst. financiera, Teléfono inst. financiera, Fax inst. Financiera)
      - Para 1 RET y 13 DET’s la complejidad es Baja.
    - Correo (Número compra, Nombre comprador, Nombre de la aseguradora, Nombre de la financiera, Marca del vehículo, Modelo, Año-modelo, Color, Accesorios, Precio)
      - Para 1 RET y 10 DET’s la complejidad es Baja.

- **Identificación de Transacciones**
  - Pantalla 1 (Drop-downs):
    - Año-modelo (flecha y campo)
      - Para 2 DET's y 1 FTR la complejidad para EQ es Baja.
    - Marca-modelo (flecha y campo)
      - Para 2 DET's y 1 FTR la complejidad para EQ es Baja.
    - Accesorios (flecha y campo)
      - Para 2 DET's y 1 FTR la complejidad para EQ es Baja.
    - Rango de precios (flecha y campo, se requiere cálculo)
      - Para 2 DET's y 1 FTR la complejidad para EO es Baja.
  - Pantalla 2 (Marca, Modelo, Año-modelo, Precio, Color, Condición, Radio button, Botón "Ver Detalle"). Se obtienen todos los datos de un FTR (Vehículo).
    - Para 8 DET's y 1 FTR la complejidad para EQ es Baja.
  - Pantalla 3 (Color, Kilometraje, Condición, Numero serie, Precio, Accesorios (aunque se repite), Botón "Comprar", Foto). Se obtienen todos los datos de un FTR (Vehículo).
    - Para 8 DET's y 1 FTR la complejidad para EQ es Baja.
  - Pantalla 4 (Nombre comprador, Dirección comprador, Tel comprador, E mail, Compañía seguros, Agente compañía seguros, Teléfono compañía seguros, Fax compañía seguros, Nombre inst. financiera, Teléfono inst. financiera, Fax inst. Financiera, Botón "Comprar", Número compra procesamiento del mail, Marca vehículo, Modelo, Año-modelo, Color, Accesorios, Precio). Se insertan los datos en Compra y se actualiza un Vehículo, = 2 FTR's)
    - Para 19 DET's y 2 FTR la complejidad para EI es Alta.

- **Cálculo de PF no ajustados (UFP)**

Nombre	Tipo	Complejidad	UFP
Consultar listado año-modelo	EQ	Baja	3
Consultar listado marca-modelo	EQ	Baja	3
Consultar listado accesorios	EQ	Baja	3
Mostrar listado rango precios	EO	Baja	4
Vehículo	EIF	Baja	5
Compra	ILF	Baja	7

Correo	ILF	Baja	7
Consultar listado vehículos	EQ	Baja	3
Consultar detalle vehículos	EQ	Baja	3
Insertar compra	EI	Alta	6
Total			44

- **Cálculo del valor de factor de ajuste (VAF)**

Característica general del sistema	Valor (Caso 1)	Valor (Caso 2)
Comunicación de Datos	5	0
Procesamiento de Datos Distribuidos	5	0
Rendimiento	5	0
Configuración de Uso Intensivo	5	0
Nivel de Transacciones	5	0
Entrada de Datos en Línea	5	0
Eficiencia del Usuario Final	5	0
Actualización en Línea	5	0
Procesamiento Complejo	5	0
Reutilización	5	0
Facilidad de Instalación	5	0
Facilidad de Operacional	5	0
Múltiples Sitios	5	0
Facilidad de Cambio	5	0
SUMA (TDI)	70	0
VAF = (TDI x 0.01) + 0.65	1.35	0.65

- **Resultado de PF ajustados (AFP)**

  Una vez calculados los puntos de función sin ajustar y el valor de factor de ajuste se procede a calcular los puntos de función ajustados. En este ejemplo se calcularon dos valores de factor de ajuste con el propósito de ilustrar cuál es el rango en el que pueden estar los puntos de función al ajustarse. Resultado con factor de ajuste ± 35%:

Total UFP	44
AFP C1	59.4
AFP C2	28.6

3.5.3. Medición con COSMIC del caso de estudio: Compra-Venta de Autos

Fase de estrategia:

- **Propósito**: determinar el tamaño funcional para posteriormente calcular el esfuerzo y costo del proyecto.
- **Alcance**: global, ya que la medición considera toda la funcionalidad descrita en los requerimientos.
- **Identificación de capas**: se considera una sola capa global de aplicación.
- **Identificación de usuarios funcionales**:
  - Cliente (envía y recibe información)
  - SIV (envía y recibe información)
  - Sistema de correo (recibe información)
- **Nivel de granularidad**: de las descripciones textuales, se identifica un nivel de granularidad adecuado para realizar la medición.
- **Diagrama de contexto**:

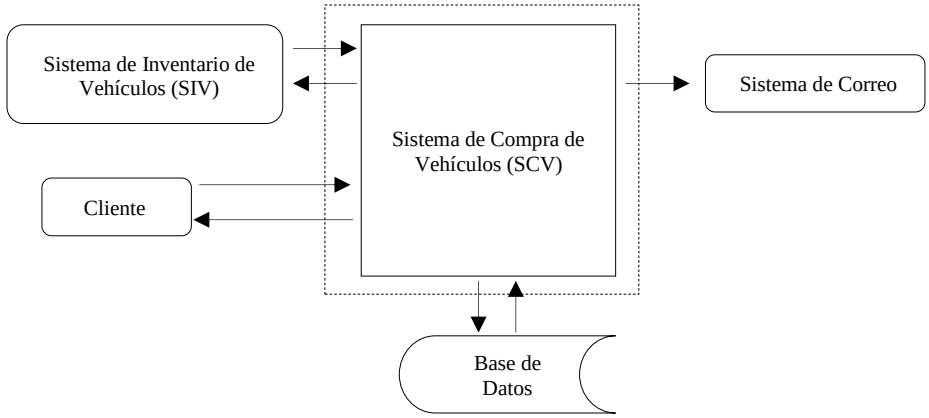


Diagrama 3.6 - Diagrama de Contexto de SCV

Fase de Representación:

- **Identificación de procesos funcionales**: de las descripciones textuales, se identifican los siguientes eventos desencadenantes y procesos funcionales.

Evento desencadenante	Proceso funcional
Cliente desea consultar listado vehículos	Consultar lista de vehículos
Cliente desea ver el detalle de un vehículo	Ver detalle de vehículo
Cliente desea comprar un vehículo	Comprar vehículo

- **Identificación de objetos de interés y grupos de datos**: De las descripciones textuales, se identifican los siguientes objetos de interés y grupos de datos.

Objeto de interés	Grupos de Datos
Vehículo	Datos de Vehículo
Accesorios de Vehículo	Datos de Accesorio de vehículo
Compra de Vehículo	Datos de Compra

- **Identificación de movimientos de datos** de los procesos funcionales: La identificación de movimientos de datos se describe en la siguiente tabla.

Proceso Funcional	Evento Desencadenante	Descripción de Subprocesos	Grupo de Datos	Mov. de datos	CFP	Σ CFP
Consultar lista de vehículos	Cliente desea consultar listado vehículos	Cliente accede a la página y la opción de búsqueda	Comando	E	1	
		Sistema solicita información a SIV para llenar filtros de búsqueda (drop-downs)	Vehículo	X	1	
		Sistema recibe la información de SIV para llenar filtros de búsqueda (drop-downs)	Año/Modelo, Marca/Modelo, Accesorio, Rango precio	E	4	
		Sistema muestra los filtros de búsqueda (drop-downs)	Año/Modelo, Marca/Modelo, Accesorio, Rango precio	X	4	
		Cliente selecciona los filtros para buscar un vehículo, sino selecciona nada se buscan todos los vehículos, presiona el botón “buscar”	Parámetros de Vehículo	E	1	
		Sistema envía los filtros seleccionados a SIV.	Parámetros de Vehículo	X	1	
		Sistema recibe de SIV la lista de vehículos que cumplen con los filtros seleccionados.	Vehículo	E	1	
		Sistema muestra lista de vehículos que cumplen con los filtros seleccionados.	Vehículo	X	1	
		Si no se encuentran resultados, el sistema muestra “No hay vehículos con esas características”.	Mensaje de error	X	1	
						15
Ver detalle de vehículo	Cliente desea ver el detalle de un vehículo	Cliente selecciona vehículo y presiona “Ver Detalle”	Vehículo	E	1	

		Sistema solicita a SIV el detalle del vehículo seleccionado	Vehículo	X	1	
		Sistema recibe la información de SIV del vehículo	Vehículo, Accesorio	E	2	
		Sistema muestra el detalle del vehículo seleccionado	Vehículo, Accesorio	X	2	
						6
Comprar Vehículo	Cliente desea comprar un vehículo	Cliente selecciona la opción “Comprar”	N/A	N/A	0	
		Sistema muestra el formulario de captura de compra	N/A	N/A	0	
		Cliente captura los detalles de la compra y selecciona “Comprar”	Compra	E	1	
		Sistema envía información a SIV para que el auto quede como “comprado”	Compra	X	1	
		Sistema guarda la información de compra en el almacenamiento de compras.	Compra	W	1	
		Sistema envía correo de confirmación al comprador.	Correo de Compra	X	1	
						4
Total:				25 CFP v4.0.1		

Fase de Medición:

La fase de medición consiste en la asignación de reglas numéricas, dicha asignación se muestra en la última columna de la tabla presentada en la identificación de movimientos de datos de la fase de representación.

La siguiente tabla muestra el total de movimientos de datos, total por procesos funcionales y total por casos de uso.

Proceso Funcional	Entradas	Salidas	Lecturas	Escrituras	Total	%
Consultar lista de vehículos	7	8	0	0	15	52.4%
Ver detalle de vehículo	3	3	0	0	6	28.6%
Comprar Vehículo	1	2	0	1	4	19%
Total	11	13	0	1	25	100%

3.5.4. Comparación entre medición IFPUG y COSMIC del caso de estudio: Compra-Venta de Autos

Al tener distintas unidades de medida, IFPUG el Punto de Función y COSMIC el Punto de Función COSMIC, no es posible determinar con qué método se obtuvo mayor tamaño funcional. Sin embargo, para determinar el esfuerzo requerido para desarrollar el proyecto se debe utilizar un modelo de estimación para cada una de las mediciones que reciba como entrada el tamaño funcional y devuelva como resultado el esfuerzo requerido para llevar a cabo el proyecto. Debido a que no se cuenta con estos modelos de estimación, no se puede determinar el esfuerzo del proyecto.

Por otra parte, con estas dos mediciones se puede hacer el análisis de a cuantos Puntos de Función equivale un Punto de Función COSMIC y viceversa.

	Ajuste Mínimo	Tamaño Funcional	Ajuste Máximo
Medición IFPUG	28.6 FP	44 FP	59.4 FP
Medición COSMIC	-	25 CFP	-
1 CFP equivale a	1.144 FP	1.76 FP	2.376 FP
1 FP equivale a	0.87 CFP	0.57 CFP	0.42 CFP

Cabe mencionar que los resultados obtenidos son únicamente como referencia y solo son válidos para propósitos de ejemplos. Para poder ser utilizados en entornos reales se debe contar con una muestra lo suficientemente grande para poder obtener resultados significativos y estadísticamente válidos.

3.6. Ejercicios

3.6.1. Cuestionario de autoevaluación

3.1. ¿Cuál fue el primer método basado en una medición de tamaño funcionalidad?

3.2. ¿Cuantos y cuales métodos de medición de tamaño funcional y estándares ISO ponen en práctica la cuantificación de tamaño funcional según ISO/IEC 14143?

3.3. ¿Cuál fue el primer método de medición de tamaño funcional reconocido como estándar?

3.4. ¿Cuál es la unidad de medida del método COSMIC y cuál del método IFPUG?

3.5. ¿La medición del método COSMIC e IFPUG depende de la tecnología que se utiliza para desarrollar la pieza de software?

3.6. ¿Cuáles son las diferencias entre los métodos de 1ª y segunda generación?

3.7. ¿Cuál es la definición de “calidad de software”?

3.8. El método de medición de IFPUG y NESMA es muy similar, sin embargo, NESMA tiene dos métodos adicionales para medir el tamaño del software, ¿cuáles son estos métodos?

3.9. El proceso de medición FiSMA consta de dos partes principales, ¿cuáles son?

3.10. ¿Cuáles son los dos objetivos principales del método de medición de IFPUG?

3.11. ¿Cuáles son las tres fases que componen el proceso de medición del método COSMIC?

3.12. Responde verdadero o falso las siguientes afirmaciones:

- El método COSMIC se basa en sólidos principios de Ingeniería de Software. Estos principios se resumen en dos modelos, el Modelo Genérico de Software y el Modelo Contextual de Software.
- El valor de factor de ajuste de IFPUG ajusta los puntos de función hasta un +/- 30%.
- El valor de factor de ajuste de IFPUG está basado en 14 características generales del sistema que miden la funcionalidad general de la aplicación.
- Los movimientos de datos que existen en el método de medición de COSMIC son: Entradas, Salidas, Lecturas, Consultas Externas y Escrituras).
- El tamaño mínimo de un proceso funcional en el método COSMIC es 1 CFP.
- Según el método COSMIC, el alcance de la medición es el conjunto de los FUR que deben incluirse en un determinado ejercicio de medición de tamaño funcional.
- Si el método COSMIC se extiende localmente, entonces el tamaño medido por la extensión local debe ser añadido al tamaño obtenido por el método estándar, medido en CFP.

- En COSMIC, un proceso funcional deberá incluir al menos un movimiento de datos de Entrada (E) y un movimiento de Escritura (W) o Salida (X).
- El tamaño mínimo de un cambio en COSMIC es 2 CFP.

3.6.2. Caso de uso: Administrar Empleados

Realizar la medición de tamaño funcional utilizando el método COSMIC del caso de uso: Administrar Empleados.

Flujo de Eventos

- Flujo Básico

<table>
<tr><th>Acción que realiza el actor</th><th>Acción que realiza el Sistema</th></tr>
<tr><td rowspan="2">1. Selecciona Administrar Empleados.

3. Selecciona Buscar empleado

5. Ingresa criterio(s) de búsqueda.</td><td>2. Muestra las opciones
• Buscar empleado
• Agregar empleado <u>AO01 Agregar empleado.</u>

4. Muestra pantalla de búsqueda de empleado con los siguientes filtros de búsqueda:
• CURP
• Nombre
• Primer apellido
• Segundo apellido
El sistema muestra la opción “Buscar”.

6. Valida que los datos ingresados tengan el formato y contexto correcto. De lo contrario muestra mensaje de error.

7. Valida que el empleado buscado exista en la base de datos. De lo contrario muestra mensaje de error.

8. Muestra detalle de empleado(s) encontrados junto con las certificaciones que ha tomado y las opciones:
• Modificar empleado <u>AO02 Modificar empleado.</u>
• Eliminar empleado <u>AO03 Eliminar empleado.</u></td></tr>
<tr><td>Fin del flujo básico.</td></tr>
</table>

Flujos Alternos

- AO01 Agregar empleado

Acción que realiza el actor	Acción que realiza el Sistema
2. Ingresa datos requeridos. **3.** El usuario selecciona opción “Guardar”	**1.** El sistema muestra pantalla de “Agregar”, con los siguientes campos: Datos de empleado:

Acción que realiza el actor	Acción que realiza el Sistema
	• CURP • Nombre • Primer apellido • Segundo apellido • Sexo • Fecha de Nacimiento • Correo electrónico • País • Entidad Federativa • Municipio o alcaldía • Localidad • Colonia • Calle • No. Exterior • No. Interior • Código postal • Teléfono particular • Teléfono celular Datos de certificaciones (se puede agregar más de una certificación) • Nombre de certificación • Fecha de inicio • Fecha de fin • Duración • Adjuntar la constancia El sistema muestra la opción “Guardar”. **4.** El sistema valida que los datos tengan el formato y contexto correcto. De ser correctos guarda los datos ingresados y genera la clave del empleado; de lo contrario muestra un mensaje de error. **5.** De ser correcto, el sistema muestra mensaje “Guardado”.
	Fin del flujo AO01 Agregar empleado.

- AO02 Modificar empleado

Acción que realiza el actor	Acción que realiza el Sistema
1. Selecciona opción modificar. **3.** Modifica datos necesarios del empleado existente. **6.** Confirma la ejecución del proceso a efectuar.	**2.** Habilita los datos del empleado y certificaciones y muestra la opción “Guardar”. **4.** Valida que los datos ingresados tengan el formato y contexto correcto. De lo contrario muestra mensaje de error. **5.** Muestra un mensaje de confirmación de modificación.

Acción que realiza el actor	Acción que realiza el Sistema
	7. Actualiza la información del empleado.
	Fin del flujo AO02 Modificar empleado.

- AO03 Eliminar empleado

Acción que realiza el actor	Acción que realiza el Sistema
1. Selecciona opción eliminar. **3.** Acepta confirmación de eliminación	**2.** El Sistema muestra mensaje de confirmación de eliminación: ¿Desea eliminar empleado? Con las opciones de: • Aceptar. • Cancelar **4.** Elimina el registro del empleado. **5.** Muestra mensaje de eliminación exitosa.
	Fin del flujo AO03 Eliminar empleado.

4. Aproximación de Tamaño Funcional

En el capítulo 3 se revisaron los estándares de medición de tamaño funcional haciendo principal hincapié en el método COSMIC. Sin embargo, como se establece en el método NESMA, la aplicación del método de medición estándar no es la única manera para determinar el tamaño funcional de una pieza de software, en particular en NESMA se definen FPA estimado, también llamado FPA de alto nivel y FPA indicativo, que no necesitan requerimientos de usuario detallados. Estos dos métodos son muy adecuados para ser aplicados en etapas tempranas del ciclo de vida de desarrollo de software o en caso de que el tamaño funcional debe medirse rápidamente.

En este cuarto capítulo se da la descripción de algunos métodos de aproximación de tamaño funcional basados en el método COSMIC bajo ciertas características.

En la primera parte del capítulo se describe que es la aproximación de tamaño funcional y porque en algunos casos se necesita aproximar en vez de medir. La segunda parte del capítulo describe algunos métodos de aproximación de tamaño funcional propuestos por COSMIC. La tercera parte del capítulo muestra tres estudios presentados en las conferencias especializadas en medición y estimación de software para comparar y analizar los resultados obtenidos.

Este capítulo termina con una serie de ejemplos y ejercicios propuestos para poner en práctica lo aprendido y para investigar más sobre el tema.

La información que se presenta en la primera y segunda parte del capítulo fue recopilada de la “Guía de Aproximación de Tamaño Funcional del Método COSMIC” [40], la información de los estudios presentados en las conferencias de IWSM MENSURA fueron recopilados desde la página iwsm-mensura.org.

4.1. Introducción a la Aproximación de Tamaño Funcional

4.1.1. ¿Por qué aproximar el tamaño funcional?

Una medición precisa de tamaño funcional de una pieza de software con el método COSMIC, requiere que sus FUR sean conocidos a nivel de granularidad en la que se pueden identificar sus procesos funcionales y sub-procesos de movimiento de datos. Si algunos requerimientos deben medirse antes de que se hayan definido con suficiente detalle para una medición precisa, los requerimientos se pueden medir utilizando un método de aproximación. Estos métodos definen cómo los requerimientos pueden ser medidos en niveles de granularidad superiores.

Según Desharnais et al. [41], cuando el software carece de documentación es no es posible aplicar a detalle todas las reglas de medición y los medidores deben utilizar técnicas de aproximación para medir los requerimientos.

Existen tres principales situaciones en las cuales se necesita una aproximación de tamaño funcional [40]:

1. Cuando se necesita rápidamente una medida del tamaño y una medida del tamaño aproximado es aceptable si se puede medir mucho más rápido que con un método de medición estándar. Esto se conoce como "rapid sizing".
2. En etapas tempranas en la vida de un proyecto antes de que los requerimientos se hayan especificado con suficiente detalle para una medición precisa del tamaño. Esto se conoce como "early sizing".
3. En general, cuando la calidad de la documentación de los requerimientos no es lo suficientemente buena para una medición precisa del tamaño.

Steve McConnell describió tres niveles para determinar el tamaño de software:

- Juzgar: Los expertos pueden dar una aproximación del tamaño, basado en la experiencia. Esta opinión es el medio menos preciso de aproximación. La precisión se puede reforzar si el juicio de experto puede estar vinculada a información concreta.
- Calcular: Si no hay suficiente información disponible en el nivel de detalle deseado, contar algo que está disponible y luego calcular la respuesta mediante el uso de los datos de calibración.
- Contar: Si la información detallada está disponible, la forma más exacta de determinar el tamaño es contar.

La Tabla 4.1 muestra un ejemplo entre contar, calcular y juzgar respondiendo a la pregunta, ¿Cuántas personas hay en este momento en un determinado estadio de futbol?

Contar	Calcular	Juzgar
41,392 tickets escaneados	Anillo superior 5 secciones de 1,500 asientos	45% lleno de una capacidad de 114,500 personas
24,802 en palcos	Anillo medio 16 secciones de 2,000 asientos	

1,593 staff	Palcos 25,000 asientos	
Total **67,787** personas	Total **64,500** personas	Total **51,500** personas

Tabla 4.1 - Ejemplo entre contar, calcular y juzgar.

Un método de medición lo que hace es contar, mientras que un método de aproximación calcula o aproxima y en algunos casos juzga. Además, los métodos de aproximación utilizan escalamiento o una combinación entre escalamiento y clasificación.

> *Definición*. El **escalamiento** [40] consiste en contar o medir cada requerimiento y multiplicar el conteo o medición por un número, el "factor de escala" para determinar su tamaño funcional. Un factor de escala está determinado por un proceso de calibración.

> *Definición*. La **clasificación** [40] consiste en clasificar cada requerimiento y asignar un tamaño al mismo (es decir, se aplica un factor de escala) que representa el tamaño funcional.

Cualquier aproximación al tamaño funcional es el resultado de una compensación entre la facilidad y rapidez de la medición frente a la pérdida de precisión. Antes de seleccionar un método de aproximación, el medidor debe primero comprobar si los factores de escalamiento y/o clasificación se pueden aplicar al método de aproximación. Sin embargo, como Santillo [42] declaró "el tamaño funcional de un software a ser desarrollado puede ser medido con precisión (solamente) después de la fase de especificación: esta fase se completa a menudo relativamente tarde en el proceso de desarrollo".

Los métodos de aproximación de tamaño funcional basados en el método COSMIC que se presentan en la sección 4.2, con excepción del método de Estimación de Proyectos en Contextos de Incertidumbre (Estimation of Projects in Contexts of Uncertainty – EPCU), requieren calibración local, es decir, se necesita tener un conjunto de procesos funcionales medidos (casos de uso, procesos generales o macro procesos según sea el caso) para calcular los distintos factores de escala que aplican para cada método de aproximación.

4.2. Métodos de Aproximación de Tamaño Funcional [40]

4.2.1. Proceso Funcional Promedio (Average Functional Process – AFP)

Este es un método simple para la obtención de un tamaño aproximado de una pieza de software. Se puede utilizar cuando la especificación de los requerimientos no es lo suficientemente buena para una medición precisa del tamaño funcional. Los siguientes pasos muestran como calibrar y aplicar el método de aproximación:

1. Muestreo y cálculo del tamaño del proceso funcional promedio
   1.1. Identificar una muestra de requerimientos cuyos procesos funcionales y movimientos de datos se han definido en detalle, con características similares a los requerimientos del software a ser medido.
   1.2. Identificar los procesos funcionales de estos requerimientos de la muestra.
   1.3. Medir el tamaño de los procesos funcionales de estos requerimientos de la muestra con precisión utilizando el método COSMIC estándar.
   1.4. Determinar el tamaño promedio (en unidades CFP) de los procesos funcionales de estos requerimientos incluidos en la muestra (proceso funcional promedio), entonces el promedio es el factor de escala para este método.
2. Aproximación utilizando el promedio calculado de la muestra
   2.1. Identificar y contar todos los procesos funcionales de los requerimientos del software a ser medido.
   2.2. El tamaño funcional aproximado de los requerimientos del software a ser medido es estimado multiplicando el número de procesos funcionales por factor de escala.

4.2.2. Clasificación de Tamaño Fijo (Fixed Size Classification – FSC)

El método depende de la identificación de una clasificación de tamaño de los procesos funcionales en la pieza de software a medir. Un tamaño correspondiente, o factor de escala, se asigna a cada proceso funcional.

Los requerimientos deben ser analizados para identificar los procesos funcionales y clasificar cada uno de ellos según su tamaño en una de tres o más clases de tamaño llamadas, por ejemplo, pequeño, mediano y grande. La Tabla 4.2 muestra un ejemplo de un conjunto de clases de tamaño. Las filas de la tabla muestran las tres clases de tamaños posibles y el número total de CFP que debe asignarse a un proceso funcional en cada grupo.

Clasificación	Tamaño (CFP)
Pequeño	5
Mediano	10
Grande	15

Tabla 4.2 - Ejemplo de Fixed Size Classification

Si es necesario, la tabla podría ampliarse para agregar uno o más tamaños, tales como “Muy Grande” de 20 CFP. Cuando está bien calibrado, este método debe dar un tamaño

funcional más preciso que el tamaño medio de Average Functional Process ya que tiene más rangos de clasificación.

4.2.3. Bandas de Igual Tamaño (Equal Size Bands – ESB)

Este método de aproximación puede ser refinado para dar un resultado más preciso si se dispone de datos suficientes para el tamaño de una calibración precisa y pertinente a la necesidad local de medición. En el método “Equal Size Bands”, los procesos funcionales se clasifican en bandas de tamaño, donde los límites de las bandas se eligen en el proceso de calibración de manera que el tamaño total de todos los procesos funcionales en cada banda es el mismo para cada banda, por ejemplo, si la elección es tener tres bandas, el tamaño total de todos los procesos funcionales en cada banda contribuirá 33% con el tamaño total del software que se está midiendo.

Frank Vogelezang (presidente de COSMIC) informó sobre el uso de este método para early sizing, después de haber llevado a cabo una calibración usando mediciones en 37 desarrollos de aplicaciones de negocios, cada una de tamaño total mayor a 100 CFP. Se decidió utilizar cuatro bandas de tamaño para hacer una distinción entre los procesos relativamente pequeños, procesos medianos, grandes y muy grandes. Los tamaños promedio de cada banda se muestran en la siguiente tabla.

Banda	Tamaño Promedio de un Proceso Funcional	% del Tamaño Funcional total	% del número total de Procesos Funcionales
Pequeño	4.8	25%	40%
Mediano	7.7	25%	26%
Grande	10.7	25%	19%
Muy Grande	16.4	25%	15%

Tabla 4.3 - Ejemplo de Equal Size Bands con 37 aplicaciones de negocio

Para medir el tamaño de una nueva pieza de software se identifican los procesos funcionales de la nueva pieza, y se clasifican respecto a las bandas definidas (como "Pequeño", "Medio", "Grande" o "Muy Grande"). En el siguiente paso, los tamaños promedio de cada banda se utilizan para multiplicar el número de procesos funcionales de la nueva pieza de software para obtener el tamaño total aproximado.

4.2.4. Caso de Uso Promedio (Average Use Case – AUC)

El principio de la aproximación de Average Use Case es similar a Average Functional Process pero en un nivel más alto de granularidad, el caso de uso.

La calibración local podría determinar que un caso de uso comprende, en promedio, x procesos funcionales, cada uno con un tamaño promedio de proceso funcional. Por lo tanto el tamaño promedio de casos de uso de acuerdo con esta definición local, es igual al producto del promedio de procesos funcionales por caso de uso por promedio de proceso funcional, por ejemplo, si un caso de uso tiene en promedio 3.5 procesos funcionales y el tamaño

promedio de un proceso funcional es 8, entonces el tamaño promedio de casos de uso es 3.5x8 = 28 CFP.

Para un nuevo proyecto, el tamaño funcional total es igual a multiplicar el número de casos de uso del nuevo proyecto por el tamaño promedio de casos de usos.

Con esta calibración, identificar el número de casos de uso en una etapa temprana del desarrollo de un proyecto, provee las bases para determinar una estimación de tamaño funcional en unidades CFP. La incertidumbre de éste método de aproximación debería ser mayor al de Average Functional Process, ya que el factor de escala de tamaño promedio de casos de uso es el producto de dos factores de escala que también fueron estimados.

4.2.5. Temprano y Rápido (Early & Quick)

El método de aproximación Early & Quick es una adaptación del método de Puntos de Funcion Early & Quick. El método fue originalmente propuesto en 1997 por IFPUG con el fin de ayudar a los medidores a dimensionar sistemas de software a partir de la información no detallada, por lo general con una reducción de esfuerzo y tiempo en comparación con la que se necesita para una medición estándar.

El método de aproximación Early & Quick combina escala y clasificación. Se permite el uso de diferentes niveles de granularidad para diferentes ramas del sistema en diferentes niveles de descomposición. El tamaño total aproximado es la suma del tamaño de las aproximaciones de los componentes individuales.

El método de aproximación Early & Quick se basa en la capacidad del medidor para clasificar una parte de los requerimientos en una categoría funcional particular. La Tabla 4.4 permite al medidor asignar un valor medio de CFP para ese elemento (esto se aplica para el software en cada capa identificada de la arquitectura de software por separado, de acuerdo con el método COSMIC estándar). Cada función puede ser categorizada, con el fin de aumentar la magnitud y la disminución del número de elementos que lo componen, como un proceso funcional, un proceso típico, proceso general o macro-proceso.

a) En el método de aproximación Early & Quick un proceso funcional (PF) es el proceso más pequeño, autónomo y con características significativas. No es posible, desde el punto de vista del usuario, proceder a la descomposición de un proceso funcional sin violar los principios de relevancia, autonomía y coherencia del sistema. Un proceso funcional puede ser “Pequeño”, “Mediano”, “Grande” o “Muy Grande”, dependiendo de su número estimado de movimientos de datos.
b) Un proceso típico (PT) es un conjunto de las cuatro operaciones básicas del usuario: crear, recuperar, actualizar y eliminar en los datos que describen un particular objeto de interés. Estos procesos típicos se encuentran frecuentemente en software de aplicaciones de negocio.
c) Un proceso general (PG) es un conjunto de procesos funcionales medianos y puede ser pensado como un subsistema de la aplicación. Un PG puede ser “Pequeño”,

"Mediano" o "Grande", basado en el número estimado de procesos funcionales que contiene.

d) Una macro-proceso (MP) es un conjunto de procesos generales medianos y puede ser pensado como un subsistema relevante del sistema general de información de la organización del usuario. Un MP puede ser "Pequeño", "Mediano" o "Grande", basado en el número estimado de procesos generales que contiene.

Cada nivel se construye sobre la base del nivel anterior. La Tabla 4.4 muestra una referencia apropiada que permite al medidor asignar un valor medio de CFP para ese elemento.

Tipo	Nivel	Rangos / Equivalencia con COSMIC	Mínimo CFP	Más probable	Máximo CFP
Proceso Funcional	Pequeño	1-5 movimientos de datos	2	3.9	5
	Mediano	5-8 movimientos de datos	5	6.9	8
	Grande	8-14 movimientos de datos	8	10.5	14
	Muy Grande	14+ movimientos de datos	14	23.7	30
Proceso Típico	Pequeño	CRUD (procesos pequeños/medianos) CRUD + lista (procesos pequeños)	15.6	20.4	27.6
	Mediano	CRUD (procesos medianos/grandes) CRUD + lista (procesos medianos) CRUD + lista + reporte (procesos pequeños)	27.6	32.3	42
	Grande	CRUD (procesos grandes) CRUD + lista (procesos medianos/grandes) CRUD + lista + reporte (procesos medianos)	42	48.5	63
Proceso General	Pequeño	6-10 PF´s genéricos	20	60	110
	Mediano	10-15 PF´s genéricos	40	95	160
	Grande	15-20 PF´s genéricos	60	130	220
Macro Proceso	Pequeño	2-4 PG´s genéricos	120	285	520
	Mediano	4-6 PG´s genéricos	240	475	780
	Grande	6-10 PG´s genéricos	360	760	1,300

Tabla 4.4 - Valores de referencia para las categorías funcionales del método de aproximación Early & Quick

Asignar cada parte de los requerimientos a una categoría específica más alta que el nivel de proceso funcional es bastante subjetiva. La precisión del método es fuertemente dependiente de la formación y la capacidad de los medidores que lo utilizan para comprender las categorías en los niveles más altos de granularidad.

4.2.6. Temprano y Veloz (Early & Speedy – EASY)

El método de aproximación EASY fue introducido por primera vez en el 2012, como una ejemplificación de una Técnica Rápida de Aproximación de Medición de Software (Software Measurement Approximation Rapid Technique - SMART) para medir requerimientos poco claros. El método SMART, en cualquier función, el medidor es libre de asumir uno o más posibles valores basados en su comprensión de los requerimientos que describen la funcionalidad.

Se debe tener en cuenta que el valor más probable no es necesariamente siempre el "mediano". Esto es diferente de cualquier enfoque "promedio", donde los valores promedios o medianos se toman siempre como los valores más probables. Sin embargo, el método SMART puede llevar mucho tiempo, para que el medidor pueda asignar más de un valor posible a cada función a ser medida, y por otra parte una probabilidad correspondiente a cada valor por función.

La Tabla 4.5 muestra las distribuciones de probabilidad típicos de valores aproximados para la mayoría de los casos comunes en el dominio de negocios.

Clasificación de PF	Nivel de especificación	CFP (min)	CFP	CFP (max)	CFP Aproximado	Probabilidad
PF pequeño	Poco desconocido	2 (10%)	3 (75%)	5 (15%)	3.2	>80%
PF pequeño	Desconocido (No FUR)	2 (15%)	4 (50%)	8 (35%)	5.1	<50%
PF mediano	Poco desconocido	5 (10%)	7 (75%)	10 (15%)	7.25	>80%
PF mediano	Desconocido (No FUR)	5 (15%)	8 (50%)	12 (35%)	8.95	<50%
PF grande	Poco desconocido	8 (10%)	10 (75%)	12 (15%)	10.1	>80%
PF grande	Desconocido (No FUR)	8 (15%)	10 (50%)	15 (35%)	11.45	<50%
PF complejo	Poco desconocido	10 (10%)	15 (75%)	20 (15%)	15.25	>80%
PF complejo	Desconocido (No FUR)	10 (15%)	18 (50%)	30 (35%)	21	<50%

Tabla 4.5 - Distribuciones de probabilidad de valores de aproximación en el dominio de negocios.

4.2.7. Estimación de Proyectos en Contextos de Incertidumbre (Estimation of Projects in Contexts of Uncertainty – EPCU)

El Dr. Francisco Valdés Souto (miembro del Comité de Prácticas de Medición COSMIC), trabajó en la creación del modelo EPCU desde 2006, anteriormente llamado “Estimación Difusa de Software (EDSW)”.

El modelo EPCU encuentra su principal aplicación en entornos de “información imperfecta”, es decir, cuando las decisiones se basan principalmente en variables cualitativas y en el juicio de experto, como lo es el caso de estimaciones o evaluación de algunos aspectos de los proyectos de software, para su operación el modelo EPCU toma en cuenta:

- Las variables lingüísticas utilizadas por los expertos para describir el contexto de estimación/evaluación, que normalmente se basan en información vaga o ambigua.
- La forma en que los expertos combinan estos valores lingüísticos para tomar decisiones.

El modelo EPCU es un mecanismo que permite realizar estimaciones de proyectos y evaluaciones en distintos ámbitos, por ejemplo, planeación estratégica, en los que usualmente se manejan variables subjetivas que resulta complicado medir de manera precisa (entornos de información imperfecta) y se requieren mecanismos formales para tomar decisiones.

El modelo EPCU, trabaja sobre un marco matemático de Lógica Difusa que permite manejar adecuadamente la incertidumbre.

Con la finalidad de simplificar el uso del modelo EPCU, se han desarrollado distintos mecanismos relacionados a distintos contextos EPCU que han sido definidos por expertos de distintas empresas y probados en entornos reales.

El método EPCU Approximation Approach considera dos variables cualitativas de entrada que son las que definen el tamaño funcional aproximado de un caso de uso o proceso funcional:

- Tamaño del Caso de Uso
- Grado de presencia de Objetos de Interés

La información solicitada de los casos de uso o procesos funcionales se divide en dos partes:

1. Dar una clasificación cualitativa con base en la percepción para cada caso de uso o procesos funcional, considerando cuatro valores para la variable "Tamaño del Caso de Uso" los cuales son: *Pequeño, Promedio, Grande* y *Muy grande,* y tres valores para la variable "Grado de presencia de Objetos de Interés" los cuales son: *Pocos, Promedio* y *Muchos*.
2. Asignar a cada variable ("Tamaño del Caso de Uso" y "Grado de presencia de Objetos de Interés") un valor numérico en un rango de 0 a 5 (se puede incluir un decimal), dónde el 0 es el mínimo valor y el 5 es el máximo valor, para cada caso de uso o proceso funcional.

Una vez definidas estas variables por cada caso de uso o proceso funcional, los valores numéricos asignados se dan como entrada al método EPCU Approximation Approach para generar el tamaño aproximado en unidades CFP por cada caso de uso o proceso funcional. Con base en los datos del estudio inicial, es posible generar un tamaño aproximado mínimo y máximo cuando se aproxima a nivel de procesos funcionales considerando un error

promedio de 46,7% y una desviación estándar de 23.4% y cuando se aproxima a nivel de casos de uso considerando un error promedio de 43.4% y una desviación estándar de 34.3%.

Aplicando este modelo a la aproximación de tamaño funcional se generan ventajas interesantes, como que no se requieren datos históricos en virtud de que no requiere calibrar localmente. La Tabla 4.6 muestra los métodos de aproximación que requieren calibración local y consideraciones especiales.

	Necesita calibración local	Nivel de granularidad de los requerimientos	Consideración
Average Functional Process	X	Proceso Funcional	Esta aproximación es válida mientras existan razones suficientes para suponer que la muestra en la que se calcula el tamaño del proceso funcional promedio es representativo del software cuyo tamaño funcional se aproxima.
Fixed Size Classification	X	Proceso Funcional	Esta aproximación es válida mientras existan razones suficientes para suponer que la clasificación de tamaño asignada es representativa del software cuyo tamaño funcional se aproxima.
Equal Size Bands	X	Proceso Funcional	Este método se recomienda para el dimensionamiento aproximado de software en el que la distribución de los tamaños de proceso funcionales es sesgada. Para aplicaciones de negocio este método tiene poco valor añadido sobre el método de Average Functional Process o Fixed Size Classification.
Average Use Case	X	Caso de Uso	Esta aproximación es válida mientras existan razones suficientes para suponer que la muestra en la que se calcula el tamaño del caso de uso promedio es representativo del software cuyo tamaño funcional se aproxima.
Early & Quick	X	Enfoque Multinivel	La precisión del método depende en gran medida del entrenamiento y la capacidad de los profesionales que lo utilizaron para entender las categorías en los niveles más altos de granularidad. Este enfoque de aproximación combina enfoques de escala y clasificación.
Easy	X	Proceso Funcional	La precisión es directamente proporcional al nivel de granularidad del proceso funcional analizado.
EPCU		Proceso Funcional y Casos de Uso	No requiere calibración local (menos costoso) y es útil cuando no hay datos históricos disponibles.

Tabla 4.6 - Métodos de aproximación que requieren calibración local y consideraciones especiales

4.3. Estudios sobre mecanismo de aproximación basados en EPCU

4.3.1. Aproximación de tamaño funcional utilizando Procesos Funcionales, sin utilizar datos históricos

Es conocido que las técnicas más utilizadas para aproximar el tamaño de una pieza de software, en etapas tempranas del desarrollo, involucran datos históricos. Sin embargo, reunir datos históricos es un reto en sí mismo.

El propósito de este estudio fue utilizar el método EPCU Approximation Approach y el método de ESB para aproximar el tamaño funcional de una pieza de software de referencia como lo es el C-Registration System, considerando los CPF aproximados y compararlos con el valor real de la medición de dicha pieza de software.

El C-Registration System [43] es un sistema de referencia que consiste en un conjunto de requerimientos totalmente definidos (estables), previamente conocidos para la aproximación del tamaño funcional. Sin embargo, como el rendimiento de los métodos de aproximación suele evaluarse con los proyectos terminados, el tamaño funcional para el C-Registration System es 107 CFP.

Se desarrolló un experimento con 9 practicantes, los cuales no estaban familiarizados con el método COSMIC, no tenían datos históricos para calibrar el método ESB, ni participaron en la definición del contexto EPCU o conocen el modelo EPCU. La única información que tenían era los requerimientos del caso de estudio del C-Registration System.

En el experimento, los practicantes utilizaron el método EPCU dentro de un contexto previamente definido, la variable de salida se definió utilizando un rango continuo de valores posibles con un límite superior "natural", o de corte, a 16.4 CFP. Los practicantes también aplicaron el método ESB utilizando la calibración de las bandas de tamaño según el estudio realizado por Vogelezang [40].

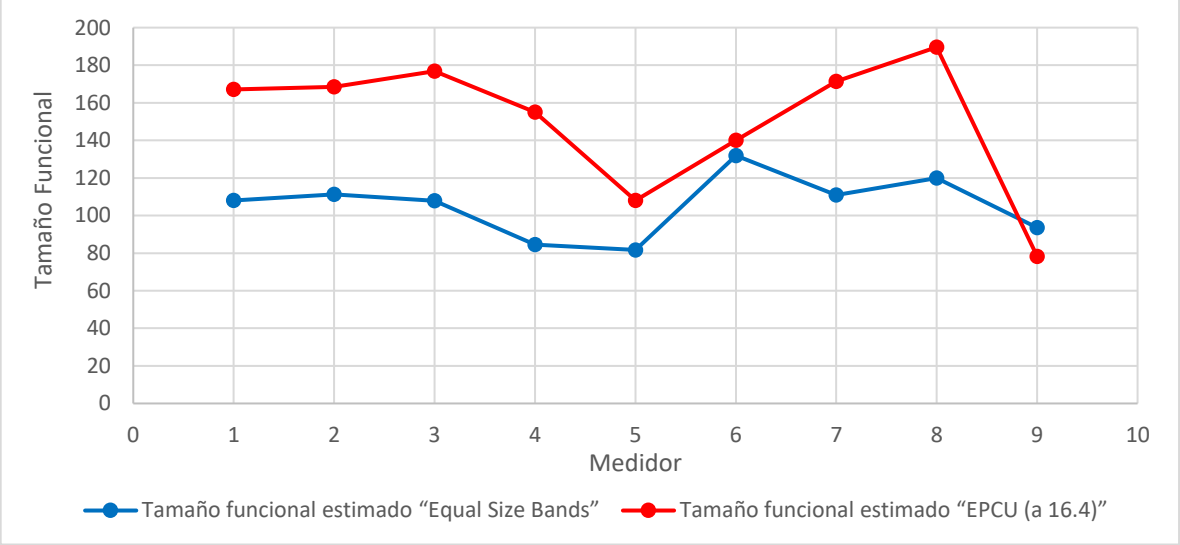
La Tabla 4.7 muestra los resultados obtenidos por nueve practicantes al aplicar los métodos ESB y EPCU al sistema CRS.

La aproximación utilizando el método EPCU muestra una Media de la Magnitud del Error Relativo (Mean Magnitude of Relative Error – MMRE) = 47%, con una Desviación Estándar de la Magnitud del Error Relativo (Standard Deviation of Magnitude of Relative Error – SDMRE) = 23%. En comparación con los resultados obtenidos con ESB (MMRE = 11%, SDMRE = 9%), este MMRE es superior al 36%, y su SDMRE también más alta, al 14%.

La Gráfica 4.1 muestra la dispersión de los resultados de cada practicante al aproximar el tamaño funcional utilizando ESB y EPCU.

Practicante	Tamaño funcional medido "COSMIC"	Tamaño funcional aproximado "Equal Size Bands"	MRE	Tamaño funcional aproximado "EPCU (a 16.4)"	MRE	Diferencia
1	107	108.1	1%	167.1	56%	55%
2	107	111.2	4%	168.4	57%	53%
3	107	107.9	1%	176.9	65%	64%
4	107	84.6	21%	155.1	45%	24%
5	107	81.7	24%	108.1	1%	23%
6	107	131.9	23%	140.1	31%	8%
7	107	111	4%	171.3	60%	56%
8	107	119.9	12%	189.6	77%	65%
9	107	93.5	13%	78.3	27%	14%
MMRE			11%		47%	
SDMRE			9%		23%	

Tabla 4.7 - Análisis de MRE de ESB y EPCU con la medición. Estudio 2012 [44].



Gráfica 4.1 - Dispersión de aproximaciones de ESB y EPCU. Estudio 2012 [44].

Es posible ver en la Gráfica 4.1 que el método ESB y EPCU se comportan de manera similar. Como señala Vogelezang [40], "una medición de tamaño rápido será aceptable si se puede producir más rápido y puede proporcionar una aproximación fiable de la medición de tamaño funcional detallada".

Teniendo en cuenta el experimento desarrollado, podemos concluir que el uso del método EPCU es muy útil cuando no hay datos históricos disponibles, además de ser menos costoso que ESB, que requiere datos históricos.

4.3.2. Aproximación de tamaño funcional utilizando Casos de Uso, sin utilizar datos históricos

De igual manera que el estudio llevado a cabo en el 2012, el propósito de este estudio es utilizar el método EPCU Approximation Approach y el método de ESB para aproximar el tamaño funcional de una pieza de software no de referencia, sino de industria para simular de mejor manera una situación real, por lo que se realizó a nivel de casos de uso, comparando los resultados con el valor real de la medición de dicha pieza de software.

El proyecto ALFA es un proyecto real de industria para una institución federal mexicana y fue utilizado en el experimento con 8 practicantes. Los practicantes que participaron en el experimento no estaban familiarizados con el método COSMIC, no tenían datos históricos para calibrar ESB, ni participaron en la definición del contexto EPCU o conocían el modelo EPCU, la única información que tenían era la lista de casos de uso identificados y su experiencia con el proceso de negocio relacionado con el proyecto.

El experimento en este trabajo fue diseñado para probar el método EPCU en etapas tempranas en el ciclo de vida de un proyecto: sólo los nombres de los casos de uso de un proyecto de software fueron compartidos con los participantes a través de un formulario de encuesta.

En el experimento, los practicantes utilizaron el método EPCU dentro de un contexto previamente definido, la variable de salida se definió utilizando un rango continuo de valores posibles con un límite superior "natural", o de corte, a 16.4 CFP. Los practicantes también aplicaron el método ESB utilizando la calibración de las bandas de tamaño según el estudio realizado por Vogelezang [40].

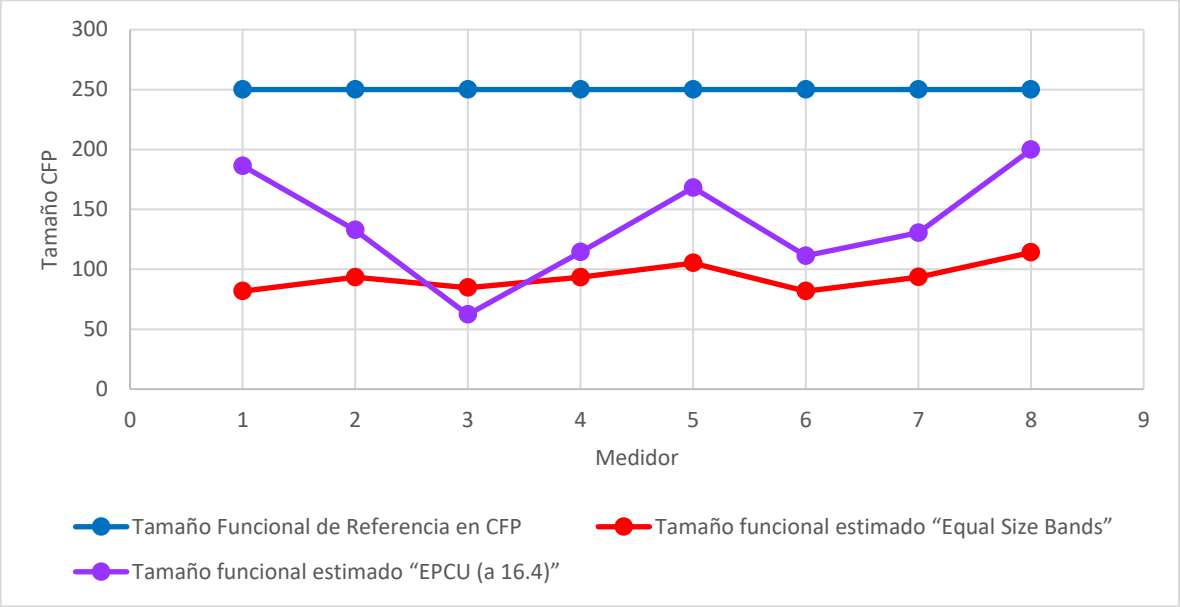
Tabla 4.8 muestra los resultados obtenidos por ocho practicantes al aplicar los métodos ESB y EPCU a una pieza de software.

Practicante	Tamaño funcional medido “COSMIC”	Tamaño funcional aproximado “Equal Size Bands”	MRE	Tamaño funcional aproximado “EPCU (a 16.4)”	MRE	Diferencia
1	250	81.7	67%	186.32	25%	42%
2	250	93.3	63%	132.76	47%	16%
3	250	84.6	66%	62.19	75%	9%
4	250	93.3	63%	114.34	54%	9%
5	250	105.2	58%	168.13	33%	25%
6	250	81.7	67%	111.26	55%	12%
7	250	93.5	63%	130.43	48%	15%
8	250	114	54%	199.82	20%	34%
MMRE			63%		45%	
SDMRE			5%		18%	

Tabla 4.8 - Análisis de MRE de ESB y EPCU con la medición. Estudio 2014 [45].

En este estudio el método EPCU presentó mejores resultados con un MMRE de 45% en comparación con ESB (MMRE = 63%). Sin embargo, ESB presentó un mejor resultado en la SDMRE (5%) que EPCU (18%).

La Gráfica 4.2 muestra la dispersión de los resultados de cada practicante al aproximar el tamaño funcional utilizando ESB y EPCU en comparación con el tamaño funcional medido.



Gráfica 4.2 - Dispersión de aproximaciones de ESB y EPCU. Estudio 2014 [45].

El tamaño funcional aproximado con ESB y EPCU por todos los practicantes está por debajo del tamaño funcional medido (real).

Los mejores resultados obtenidos por el método EPCU podrían estar asociados al uso de casos de uso en lugar de procesos funcionales, aunque los casos de uso se encuentran en un nivel de granularidad superior al de los procesos funcionales.

4.3.3. Mejorando el método de aproximación EPCU Approximation Approach

Tomando como referencia los resultados obtenidos en el estudio del 2014, en este estudio se agregó la aproximación de tamaño funcional utilizando el método EPCU Approximation Approach a nivel de granularidad de casos de uso (con un máximo aproximado por caso de uso de 44 CFP), que se obtuvo al analizar los datos del estudio realizado por Vogelezang [40], e identificar que había valores más grandes que podían ser considerados. Se analizaron los resultados al igual que en el estudio anterior.

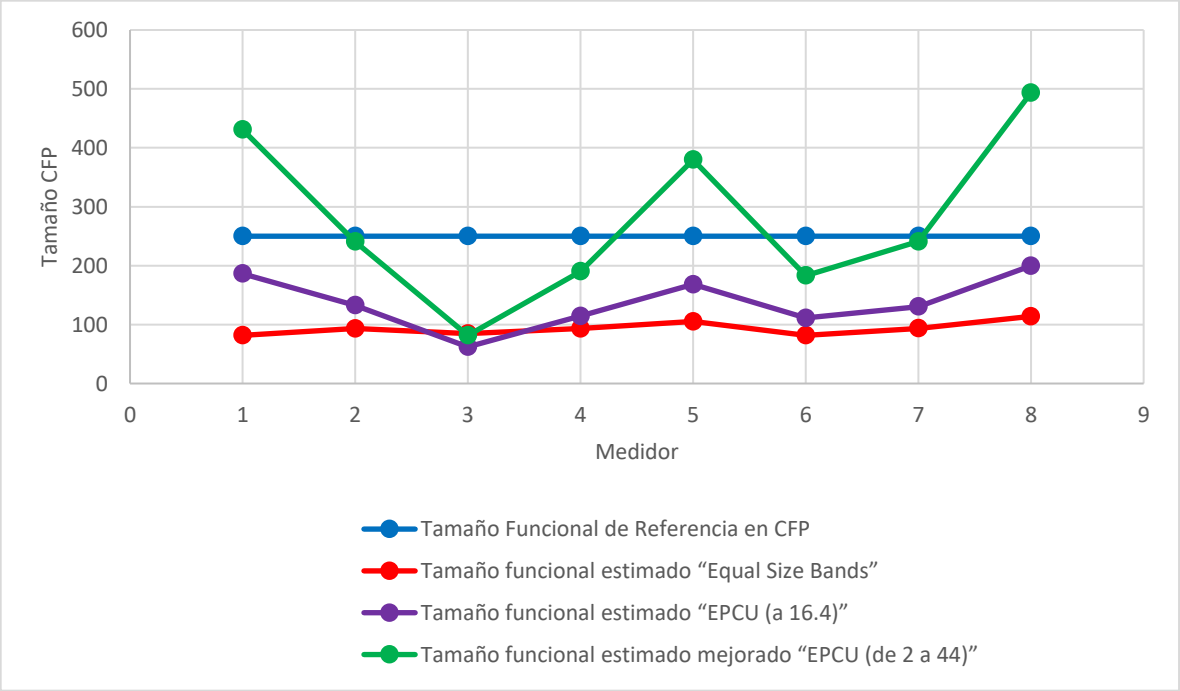
La Tabla 4.9 muestra los resultados obtenidos en la Tabla 4.8 agregando la aproximación de tamaño funcional utilizando EPCU a nivel de casos de uso.

Medidor	Tamaño funcional medido "COSMIC"	Tamaño funcional aproximado "Equal Size Bands"	MRE	Tamaño funcional aproximado "EPCU (a 16.4)"	MRE	Tamaño funcional aproximado "EPCU (a 44)"	MRE
1	250	81.7	67%	186.32	25%	430.76	72%
2	250	93.3	63%	132.76	47%	240.74	4%
3	250	84.6	66%	62.19	75%	81.65	67%
4	250	93.3	63%	114.34	54%	190.33	24%
5	250	105.2	58%	168.13	33%	379.86	52%
6	250	81.7	67%	111.26	55%	183.14	27%
7	250	93.5	63%	130.43	48%	240.91	4%
8	250	114	54%	199.82	20%	493.47	97%
MMRE			63%		45%		43%
SDMRE			5%		18%		34%

Tabla 4.9 - Análisis de MRE de ESB y EPCU a 16.4 y 44 con la medición. Estudio 2015 [46].

En este experimento, el método EPCU mejorado (a 44 CFP) presentó mejores resultados con un MMRE del 43% en comparación con ESB con un MMRE = 63% y EPCU (a 16.4 CFP) con un MMRE = 45%.

La Gráfica 4.3 agrega a la Gráfica 4.2 los resultados de cada practicante al aproximar el tamaño funcional utilizando el método EPCU mejorado (a 44 CFP).



Gráfica 4.3 - Dispersión de aproximaciones ESB y EPCU a 16.4 y 44. Estudio 2015 [46].

Para los métodos ESB y EPCU (a 16.4 CFP), es posible observar una subestimación del tamaño funcional. Por otro lado, utilizando el método EPCU mejorado (a 44 CFP), los resultados están por encima y por debajo del valor real.

En resumen, en este caso de estudio del 2015, un corte a 44 CFP presenta resultados más realistas, ya que considera procesos funcionales con un mayor tamaño funcional, algo que no está sucediendo con el corte en 16.4 CFP.

4.3.4. Conclusiones de los estudios del 2012, 2014 y 2015

Es importante considerar que la calibración local es un problema común para la mayoría de los métodos considerados en la Tabla 4.6, y una calibración local implica un costo ya que no muchas organizaciones cuentan con datos históricos. Tomando esto en cuenta, el método de aproximación EPCU no requiere calibración local y es muy útil cuando no hay datos históricos disponibles; Además, es menos costoso que la calibración del método ESB que requiere datos históricos.

Con estos métodos de aproximación, una organización puede recopilar datos históricos sobre los valores de los criterios de calidad (como MMRE y SDMRE) sin necesidad de realizar mediciones detalladas, después cuando se completan y verifican los requerimientos de software, las aproximaciones se pueden validar con mediciones detalladas. Los valores de los criterios de calidad se deben identificar mediante experimentación extensiva, y en su caso, se debe generar un informe de resultados para aquellos que utilicen estos métodos, con el fin de aproximar el tamaño funcional.

Estos experimentos tuvieron como objetivo mejorar el método EPCU para aproximar el tamaño funcional sin datos históricos, tomando en cuenta el conjunto de datos utilizados para definir el método ESB.

4.4. Ejemplos

4.4.1. Conjunto de Datos 1 para calibración de métodos de aproximación

Resultados de medición (CFP) de casos de uso (CU) con sus procesos funcionales (PF).

CU	PF	CFP v4.0.1
1	1	7
	2	2
	3	3
	4	7
2	5	14
	6	5
	7	4
3	8	10
	9	17
4	10	6
5	11	7
	12	6
6	13	10
	14	10
	15	5
7	16	9
8	17	5
	18	9
	19	7
	20	8
9	21	4
	22	8
	23	8
10	24	8
	25	9
11	26	6
	27	6
	28	4
12	29	2
	30	5
	31	5
13	32	6
	33	6
	34	7
14	35	10
15	36	11
	37	2
	38	12
	39	6
16	40	3
	41	23
	42	5
17	43	5
	44	6
	45	6
	46	4
	47	3
18	48	4
	49	7
	50	5
19	51	9
	52	3
	53	13
	54	5
	55	12
	56	5
	57	6
	58	4
	59	7
	60	6
20	61	2
	62	5
	63	5
	64	5
	65	6
21	66	6
	67	12
	68	7
22	69	3
	70	5
	71	6
	72	7
	73	3
23	74	4
	75	8
	76	7
24	77	4
	78	6
25	79	15
	80	12
	81	16
	82	5
26	83	14
	84	10
27	85	7
	86	15
	87	15
	88	3
	89	8
	90	8
	91	20
28	92	25
	93	8
29	94	15
	95	2
	96	7
	97	5
30	98	14
	99	3
	100	15

4.4.2. Calibración con métodos de aproximación AFP, AUC, FSC, ESB del Conjunto de Datos 1

En este ejemplo se realizará la calibración local de los métodos de aproximación AFP, AUC, FSC, ESB. Cabe mencionar que los resultados obtenidos son únicamente como referencia y solo son válidos para propósitos de ejemplos. Para poder ser utilizados en entornos reales se deben calibrar los métodos de aproximación con datos proyectos reales.

- Average Functional Process (AFP)

  AFP = tamaño funcional / número de procesos funcionales

No. de PF	CFP (v4.0.1)	AFP (CFP v4.0.1 x FP)
100	760	**7.6**

- Average Use Case (AUC)

  AUC = (número de procesos funcionales / número de casos de uso) x AFP

No. de CU	No. de PF	Promedio PF x CU	AFP (CFP v4.0.1 x FP)	AUC (CFP v4.0.1 x UC)
30	100	3.33	7.6	**25.33**

  o bien AUC = tamaño funcional / número de casos de uso

No. de CU	CFP (v4.0.1)	AUC (CFP v4.0.1 x UC)
30	760	**25.33**

- Fixed Size Classification (FSC)

Banda	CFP (v4.0.1)
Pequeño	5
Mediano	10
Grande	15
Muy Grande	20

- Equal Size Bands (ESB)

Banda	Total de CFP (v4.0.1)	Número de Procesos Funcionales	Promedio CFP (v4.0.1)
Pequeño	188	44	4.3
Mediano	188	27	7
Grande	194	18	10.8
Muy Grande	190	11	17.3

4.4.3. Aproximación del Caso de uso: Iniciar sesión y del Caso de uso: Administración de avisos de venta de vehículos

En este ejemplo se realizará la aproximación de tamaño funcional utilizando los métodos de aproximación AFP, AUC, FSC, ESB calibrados en la sección 4.4.2 y utilizando el método EPCU a 16.4 y EPCU a 44, para el "caso de uso: Iniciar sesión" y el "caso de uso: Administración de avisos de venta de vehículos" detallados en las secciones 3.4.1 y 3.4.3 respectivamente.

- Aproximación por casos de uso utilizando EPCU a 44

Caso de Uso	**EPCU a 44**				**CFP v4.0.1**
	Tamaño del CU		Nivel de presencia de objetos de interés		
	Clasificación	Valor	Clasificación	Valor	
Iniciar Sesión	Pequeño	1.5	Pocos	1	**2.85**
Administración de avisos de venta de vehículos	Grande	3.9	Promedio	3	**26.36**

Nota: una vez evaluadas las variables cualitativas y cuantitativas por cada caso de uso se utilizó la herramienta de aproximación de EPCU a 44 (http://www.mepe.com.mx/utilizar-mecanismos/).

- Aproximación por casos de uso utilizando AUC

Caso de Uso	**AUC (CFP v4.0.1)**
Iniciar Sesión	**25.33**
Administración de avisos de venta de vehículos	**25.33**

- Aproximación por procesos funcionales utilizando FSC

Proceso Funcional	**Clasificación**	**Tamaño (CFP v4.0.1)**
Iniciar Sesión	Pequeño	**5**
Buscar aviso de venta	Pequeño	**5**
Registrar aviso de venta	Mediano	**10**
Actualizar aviso de venta	Mediano	**10**
Cancelar aviso de venta	Pequeño	**5**

- Aproximación por procesos funcionales utilizando AFP

Proceso Funcional	**AFP (CFP v4.0.1)**
Iniciar Sesión	**7.6**

Proceso Funcional	AFP (CFP v4.0.1)
Buscar aviso de venta	**7.6**
Registrar aviso de venta	**7.6**
Actualizar aviso de venta	**7.6**
Cancelar aviso de venta	**7.6**

- Aproximación por procesos funcionales utilizando ESB

Proceso Funcional	Clasificación	Tamaño (CFP v4.0.1)
Iniciar Sesión	Pequeño	**4.3**
Buscar aviso de venta	Pequeño	**4.3**
Registrar aviso de venta	Mediano	**7**
Actualizar aviso de venta	Mediano	**7**
Cancelar aviso de venta	Pequeño	**4.3**

- Aproximación por procesos funcionales utilizando EPCU a 16.4

Proceso Funcional	EPCU a 16.4				CFP v4.0.1
	Tamaño del CU		Nivel de presencia de objetos de interés		
	Clasificación	Valor	Clasificación	Valor	
Iniciar Sesión	Pequeño	1.5	Pocos	1	**2.62**
Buscar aviso de venta	Pequeño	1.8	Promedio	2.2	**6.76**
Registrar aviso de venta	Mediano	2.7	Promedio	2.2	**8.95**
Actualizar aviso de venta	Mediano	2.3	Pocos	1.7	**6.89**
Cancelar aviso de venta	Pequeño	1.5	Pocos	1.4	**3.49**

Nota: una vez evaluadas las variables cualitativas y cuantitativas por cada caso de uso se utilizó la herramienta de EPCU a 16.4 (http://www.mepe.com.mx/utilizar-mecanismos/).

Una vez realizadas las aproximaciones y tomando en cuenta el valor real medido (sección 3.4.2 y 3.4.4) utilizando el método COSMIC, se puede realizar un análisis para evaluar que método de aproximación fue el que presentó mejores resultados respecto al valor real. Dicho análisis se presenta en el ejemplo de la sección 5.4.1.

4.5. Ejercicios

4.5.1. Conjunto de Datos 2 para calibración de métodos de aproximación

1. Realizar la calibración local de los métodos de aproximación AFP, AUC, FSC y ESB utilizando el "Conjunto de Datos 2".

2. Realiza la aproximación de tamaño funcional del "caso de uso: Administrar Empleados (sección 3.6.2) utilizando los métodos AFP, AUC, FSC, ESB calibrados (sección 4.4.2) y los métodos EPCU a 16.4 y EPCU a 44.

Conjunto de Datos 2: Resultados de medición (CFP) de casos de uso (CU) con sus respectivos procesos funcionales (PF).

CU	PF	CFP v4.0.1
1	1	4
	2	6
	3	6
	4	3
	5	2
	6	7
2	7	6
3	8	6
4	9	8
	10	6
	11	6
5	12	6
6	13	9
7	14	12
8	15	8
9	16	10
10	17	6
	18	6
	19	4
11	20	3
	21	3
	22	4
	23	3
12	24	4
	25	3
	26	3
13	27	7
	28	7
	29	10
	30	3
14	31	4
	32	3
	33	3
	34	6

CU	PF	CFP v4.0.1
15	35	3
	36	6
16	37	3
	38	6
17	39	7
	40	3
	41	7
	42	12
	43	7
	44	7
	45	9
	46	7
	47	7
	48	7
	49	6
18	50	7
	51	7
	52	10
	53	7
	54	7
	55	11
	56	7
	57	3
	58	7
	59	7
	60	6
19	61	7
	62	6
	63	7
	64	8
	65	8
	66	7
	67	3
	68	7

CU	PF	CFP v4.0.1
20	69	7
	70	7
	71	6
21	72	7
	73	7
	74	6
	75	7
22	76	3
23	77	9
24	78	12
25	79	8
	80	7
	81	7
26	82	6
	83	7
	84	7
27	85	9
	86	12
28	87	8
	88	10
	89	6
	90	6
29	91	4
30	92	7
	93	6
	94	7
31	95	9
	96	12
	97	8
32	98	10
33	99	6
	100	6
	101	4
34	102	7

5. Estimación de Proyectos de Software

Este quinto capítulo trata sobre la estimación de proyectos de software. Una vez que hemos estudiado para qué nos sirven las Métricas de Software (capítulo 2) y como aplicar un método de medición (capítulo 3) podemos realizar estimaciones de proyectos de software confiables.

La primera parte del capítulo da una introducción a la estimación y la realidad de las estimaciones de software en la industria en los últimos años. La segunda parte del capítulo abarca el proceso de estimación, cuáles son las entradas y salidas de un proceso de estimación completo. La tercera parte da una introducción de los modelos de productividad, como el modelo de regresión lineal, y los criterios de calidad que debe tener un modelo de productividad adecuado.

El objetivo de este quinto capítulo es dar una introducción a la estimación de proyectos de software, por qué es indispensable contar con buenos modelos de estimación y cómo podemos construir modelos de estimación confiables.

Este capítulo termina con una serie de ejemplos y ejercicios propuestos para poner en práctica lo aprendido e investigar más sobre el tema.

La información de la primera y segunda parte que se presenta en este capítulo fue recopilada principalmente del libro “Software Project Estimation: The Fundamentals for Providing High Quality Information to Decision Makers” [47], la tercera parte del capítulo fue recopilada principalmente de los libros “Applied Regression Analysis: A Research Tool” [48] y “Linear Regression Analysis” [49].

5.1. Introducción a la Estimación

5.1.1. ¿Qué es la Estimación?

La definición de *estimación* según La Guía de los Fundamentos para la Dirección de Proyectos (A Guide to the Project Management Body of Knowledge - PMBOK) [50] sería el siguiente enunciado:

> *"Una evaluación cuantitativa de la cantidad o resultado probable. Normalmente se aplica a los costos, recursos, esfuerzo y duraciones del proyecto y suele estar precedida por un modificador".*

Cuando se pide una estimación, a menudo se está pidiendo un compromiso o un plan para cumplir con un objetivo. Las distinciones entre las estimaciones, las metas y los compromisos son fundamentales para la comprensión de lo que es una estimación, lo que no es una estimación, y cómo hacer mejores estimaciones.

> *Definición*. Una **estimación** [51] es la predicción más optimista con una probabilidad distinta de cero de ser cierta. Una estimación es una predicción que tiene la misma probabilidad de estar por encima o por debajo del valor real.

Estimación y planeación son temas relacionados, pero la estimación no es la planeación y la planeación no es la estimación. La estimación debe ser entendida como un proceso imparcial y analítico, la planeación debe ser entendida como un proceso parcial, de búsqueda de metas.

¿Qué es una *buena estimación*? Expertos de estimación han propuesto diversas definiciones de una buena estimación. McConnell [51] ha declarado que con una exactitud de ±10% es posible, pero sólo en proyectos bien controlados, ya que proyectos poco controlados, tienen demasiada variabilidad para lograr ese nivel de precisión.

La mayor parte de los métodos de estimación tienen como objetivo predecir el costo y esfuerzo del proyecto, de acuerdo a Tuya [52] estos métodos pueden ser clasificados en cuatro categorías:

- Juicio de experto: basado en la experiencia de una persona.
- Analogía: un enfoque más formal que el juicio de experto, donde los expertos comparan el proyecto a estimar con uno o más proyectos anteriores intentando encontrar similitudes y diferencias.
- Descomposición: basado en un análisis de las características del proyecto, haciendo estimaciones individuales sobre los componentes del proyecto.
- Modelos algorítmicos: basados en técnicas que identifican los factores clave que contribuyen al esfuerzo y generan un modelo matemático que relaciona dichos factores con el esfuerzo. Los modelos se basan normalmente en información obtenida de experiencias pasadas.

Los modelos algorítmicos representan el enfoque más formal y son los que proporcionan resultados más confiables. Estos modelos utilizan procedimientos como la regresión para

producir un modelo construido por una o varias expresiones matemáticas que relacionan el esfuerzo con una variable primaria, generalmente el tamaño, y en algunos casos, varios factores de ajuste secundarios.

Actualmente, los modelos de estimación basados en modelos empíricos incorporan técnicas heurísticas consideradas no algorítmicas en el sentido de "no deterministas" [52], es decir, no solo se considera la solución, sino la probabilidad de que esa solución sea cierta. El conocimiento requerido para implementarlos, así como el costo es usualmente alto y no hay herramientas sencillas para los usuarios.

El momento en el que se define el método de estimación también es importante, ya que algunos de los métodos de estimación necesitan de datos históricos para poder ser implementados. Por ello, Abran [53] define otra clasificación con estimación *a priori* y a *posteriori*:

- A priori: se tiene un ambiente de alta incertidumbre y un nivel de abstracción de necesidad muy alto, usualmente se tiene una ventana de tiempo o de recursos limitados y la mayoría de las variables que se conocen son cualitativas. Los métodos utilizados generalmente son el juicio de experto.
    - Ventajas: no se necesitan datos históricos, manejo de incertidumbre a través de precisión de significado y la decisión sobre juicio de experto puede ser muy valiosa debido a la información incompleta con la que se cuenta.
    - Desventajas: la experiencia le pertenece al experto no a la organización y la experiencia no se puede replicar sistemáticamente.
- A posteriori: no hay incertidumbre y se conoce toda la información. Los métodos utilizados se basan en el análisis numérico y estadístico, como COSMIC, IFPUG, COCOMO, etc.
    - Ventajas: fácil de interpretar debido a que siguen un algoritmo.
    - Desventajas: requiere datos históricos, información oportuna, confiable y precisa, lo que incrementa la infraestructura de adquisición de información y en consecuencia el costo.

5.1.2. Prácticas de estimaciones favorables y poco favorables

A pesar de que en la industria se han desarrollado un amplio número de prácticas basadas en la estimación de proyectos, se llevan a cabo *prácticas de estimación poco favorables* [54], es decir, con resultados poco confiables. La Figura 5.1 muestra una visión en un alto nivel de un proceso de estimación poco favorable.

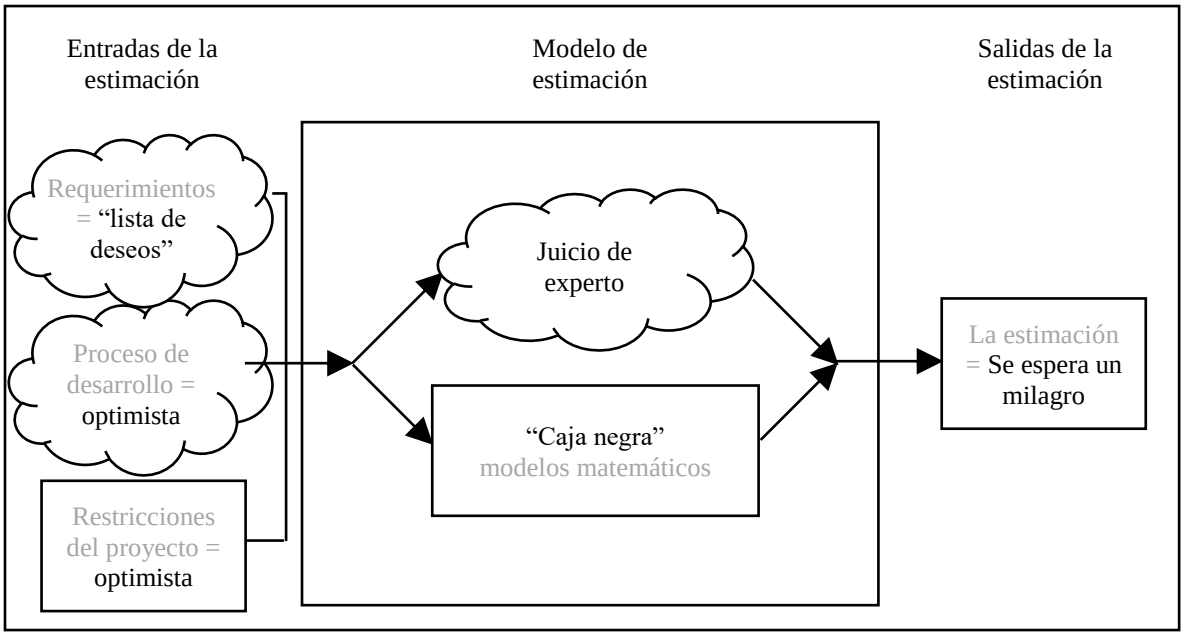


Figura 5.1 - Visión en un alto nivel de un proceso de estimación poco favorable [47].

Las entradas de un proceso de estimación poco favorable son:

- Una muy breve descripción y a un alto nivel de la descripción de los requerimientos, que inevitablemente van cambiando.
- Un proceso de desarrollo de software (ágil, iterativo, etc.), esperando tener los resultados más favorables (optimista) debido a la falta de descripción en los requerimientos.
- Restricciones del proyecto (presupuesto, tiempo, etc.), tomando en cuenta tantos factores de costos como sea posible (optimista).

La representación del modelo de estimación poco favorable incluye:

- La experiencia propia (Juicio de experto).
- Modelos matemáticos descritos en "caja negra" (juicio de experto o modelos matemáticos no documentados)

La salida del proceso de estimación poco favorable esperada consiste en:

- Una sola estimación, que se compone del presupuesto del proyecto.
- Una actitud demasiado optimista.

En este tipo de prácticas, tanto los clientes como los directores de proyectos esperan que su personal (y proveedores) se comprometen a la entrega de la funcionalidad del software dentro del tiempo y del presupuesto, dicho de otra manera, se espera un milagro.

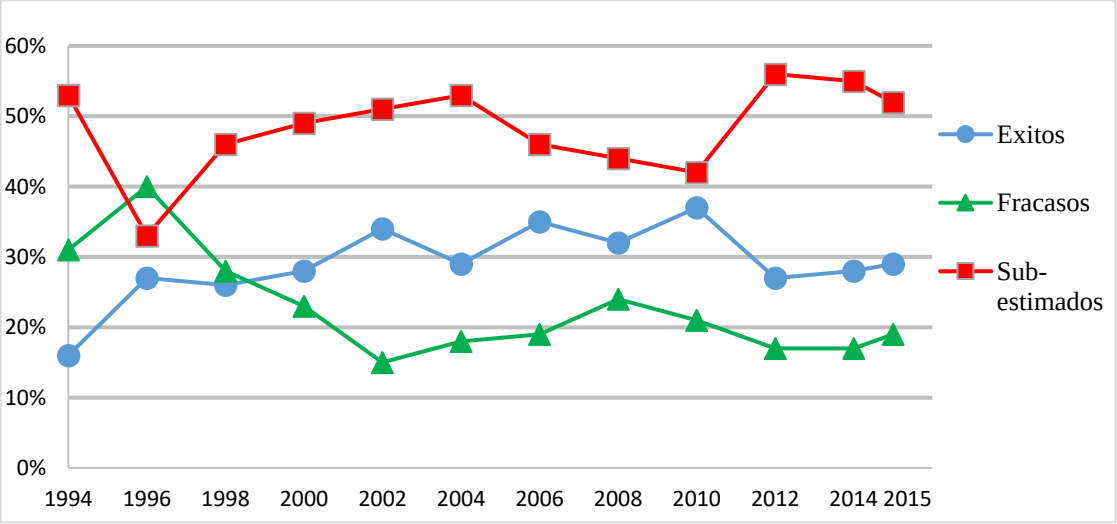
En algunas de las *mejores prácticas de estimación* en la industria, las empresas de software consideran la estimación como un proceso que les da una ventaja competitiva sobre las demás empresas, para lograr esta ventaja, consideran los siguientes factores [47]:

- La inversión en la recolección de los requerimientos del proyecto y en la comprensión de sus cualidades.
- El uso de estándares internacionales para la medición del software.
- Una medición cuantitativa continua durante todo el ciclo de vida del proyecto.
- Un análisis cuantitativo de los resultados de sus estimaciones anteriores.
- Un análisis a profundidad del seguimiento de la estimación (actual vs estimado).

5.1.3. La realidad en las estimaciones de Software

La estimación de proyectos de software es una actividad importante en pequeñas y grandes organizaciones alrededor del mundo. Una gran cantidad de investigaciones se han realizado en la estimación de proyectos de software durante más de 20 años y una gran cantidad de modelos han sido propuestos a la industria. La cuestión es: ¿qué tan eficiente es la estimación en la industria?, la respuesta es: no muy eficiente (Gráfica 5.1).

La Gráfica 5.1 muestra las tendencias de éxito de proyectos de software del año 1994 al 2015. Los resultados se obtuvieron utilizando datos de Standish Group Chaos Report.



Gráfica 5.1 - Tendencias de éxito de proyectos de acuerdo a Standish Group (adaptada de [54]).

5.2. Proceso de Estimación

5.2.1. Proceso de estimación en un alto nivel

La Figura 5.2 muestra una visión en un alto nivel de un proceso de estimación de software.

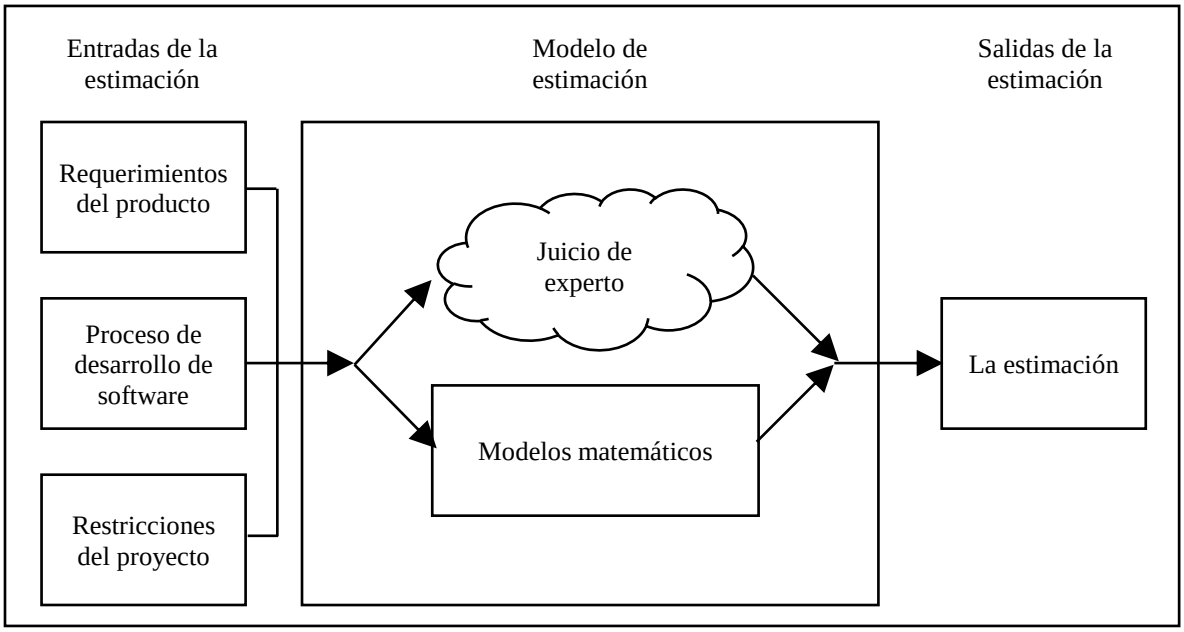


Figura 5.2 - Visión en un alto nivel del proceso de estimación de software [47].

Las entradas de un proceso de estimación de software son:

- Requerimientos del producto (funcionales y no funcionales)
- Proceso de desarrollo de software (ágil, iterativo, etc.)
- Restricciones del proyecto (presupuesto, tiempo, etc.)

La representación del modelo de estimación incluye:

- Juicio de experto (frecuentemente no documentado)
- Modelos matemáticos (regresión, razonamiento basado en casos, redes neuronales, etc.)

La salida del proceso de estimación esperada consiste en:

- Una estimación de la cantidad de esfuerzo (tiempo o costo) necesario para entregar el software.

5.2.2. Conjuntos de datos para la estimación

Estimar proyectos de software es a menudo llevado a un contexto de estimación que es caracterizado por [47]:

- La información incompleta.
- La imprecisión de los requerimientos en una estimación temprana en el ciclo de vida de un proyecto.

- La incertidumbre por factores que pueden impactar en el proyecto.
- El tamaño que puede ser mejor aproximado que medido.
- Un número de riesgos.

La técnica de estimación que se utiliza con mayor frecuencia en la industria, que no suele documentar los parámetros que toma en cuenta, es el llamado “Juicio de Experto” [55], que a diferencia de los modelos algorítmicos, es menos transparente ya que no es posible crear datos históricos de los modelos en los que se basa.

Un proyecto puede parecer tener éxito si cuenta con el presupuesto “oficial”. Sin embargo, sin la capacidad de verificar que toda la funcionalidad prometida se ha entregado, es un error afirmar que las estimaciones fueron correctas.

Cuando se entregan sólo una parte de la funcionalidad requerida, entonces no se puede obtener toda la ganancia esperada. Por lo tanto, analizar el rendimiento de una estimación basada en el Juicio de Experto sin un análisis correspondiente de la funcionalidad entregada es de un valor muy limitado.

De acuerdo con un estudio de la Government Accountability Office [56], si se establecieran bases de referencia más realistas de los requerimientos, costos, calendarios y riesgos durante las fases de planificación de los proyectos de software, se podrían evitar casi la mitad de los programas de TI cancelados o sobre estimados.

Cuando una organización no ha medido su propia productividad basada en proyectos pasados, desconoce los siguientes aspectos [47]:

- Cuál es el rendimiento de la organización.
- Qué tanto difiere el rendimiento de un gerente de algún otro gerente.
- Qué tanto difieren las estimaciones basadas en juicio de experto de un gerente con algún otro gerente.

En cambio, cuando una organización ha recolectado sus propios datos, desarrollado sus propias capacidades para analizar esos datos y documentado la calidad de sus modelos de estimación, entonces ha desarrollado credibilidad para las estimaciones, además de conocer los aspectos que desconocen las organizaciones que no han medido su propia productividad.

5.2.3. El proceso de estimación completo

El proceso de estimación contiene las siguientes cinco fases [47]:

1. Reunir las entradas del proceso de estimación:
    - Medición o aproximación del tamaño de los requerimientos.
    - Supuestos para la mayoría de los otros factores de costo.
2. Usar un modelo de productividad.
3. Utilizar un proceso de ajuste para tomar en cuenta variables e información no incluida en el modelo de productividad, incluyendo:
    - Identificación de factores de incertidumbre.
    - Evaluación de riesgos.

4. Tomar una decisión sobre el presupuesto.
5. Reestimar cuando sea requerido por el seguimiento y control de proyectos.

Existe una fase adicional a nivel de la organización y no a nivel del proyecto, la cual analiza la eficiencia del proceso de estimación una vez que un proyecto ha sido terminado.

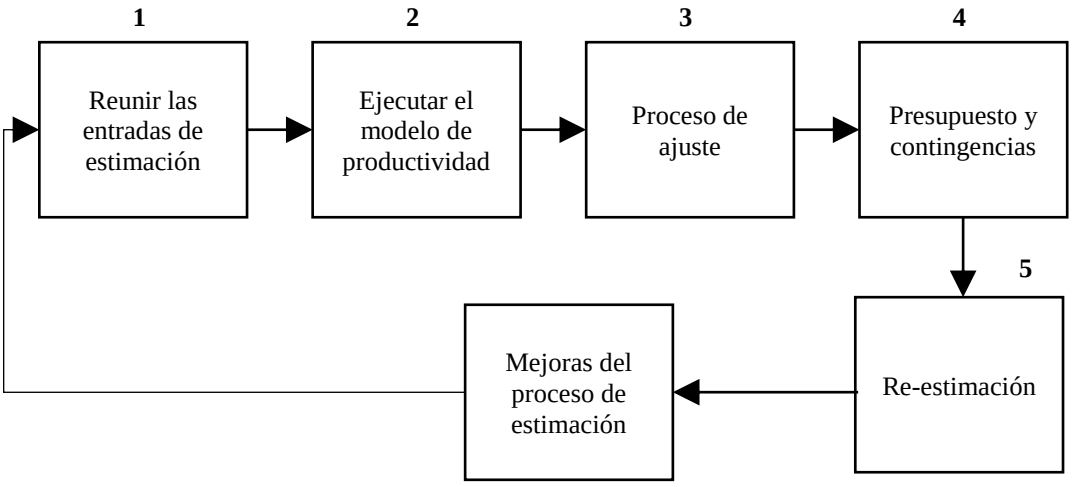


Figura 5.3 – El proceso de estimación [47].

El rol y responsabilidad del estimador de software en el proceso de estimación es:

- Construir el modelo de productividad. Esto incluye reunir los datos históricos de los proyectos pasados y documentar la calidad del modelo de productividad.
- Cuando la organización no cuenta con datos históricos de los proyectos pasados, es responsabilidad del estimador encontrar una solución alternativa.
- Llevar a cabo las tres primeras fases del proceso de estimación (Figura 5.3).

Es responsabilidad del administrador la toma de riesgos y tomar una decisión al seleccionar el presupuesto óptimo para un proyecto específico. Otras responsabilidades del administrador son:

- Implementar el proceso de estimación, incluyendo:
  - La asignación de recursos para la recolección y análisis de datos.
  - La asignación de recursos para la integración del modelo de productividad.
- La asignación de recursos calificados y capacitados para el proceso de estimación cuando se requiere una estimación para un nuevo proyecto.

El objetivo de un proceso de estimación es proporcionar:

- Información sobre rangos estimados de posibles valores.
- Retroalimentación acerca de que tan buena es la información.
- Limitaciones de la información utilizada como entrada al proceso de estimación.
- Limitaciones de la información proporcionada como salida del proceso de estimación.
- Análisis y mitigación de riesgos mediante la documentación.

Figura 5.4 muestra el detalle del proceso de estimación tomando en cuenta el rol del estimador y el rol del administrador.

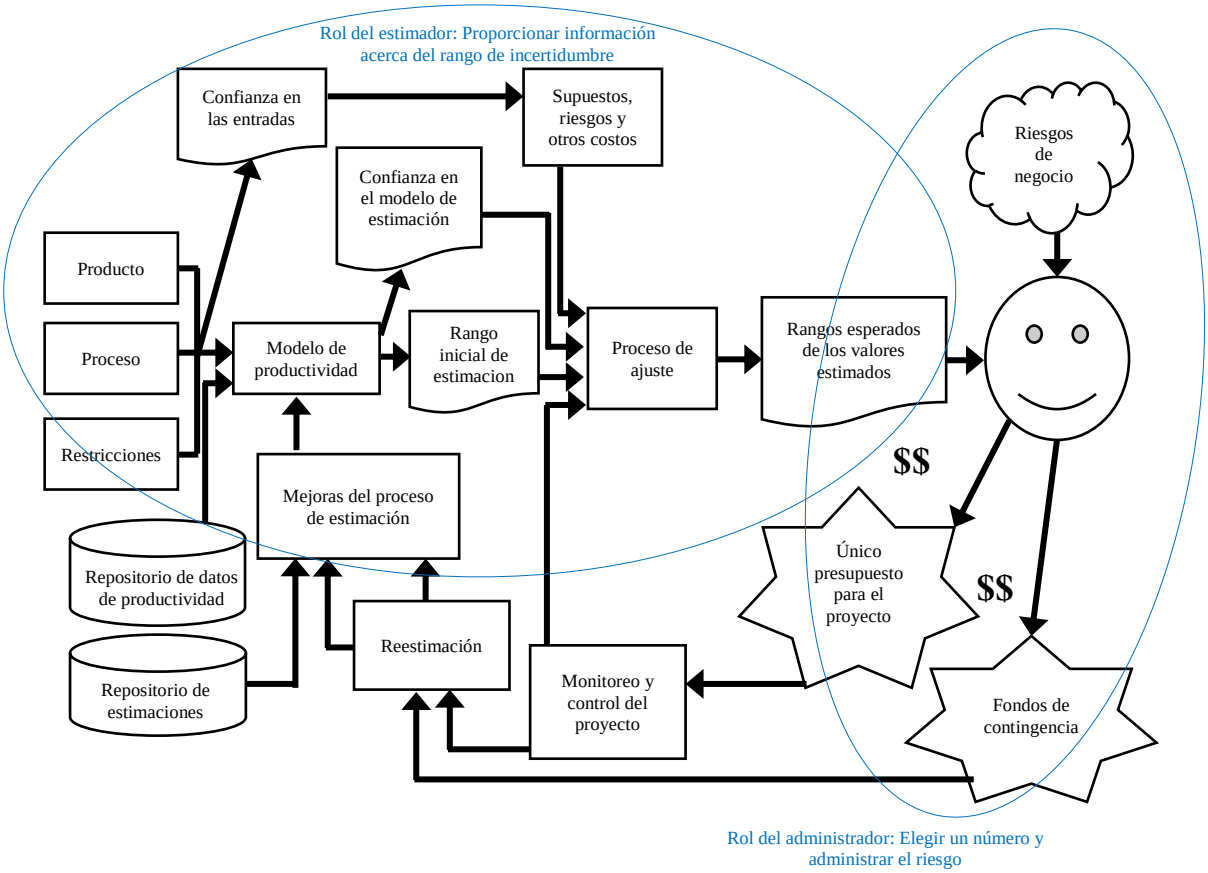


Figura 5.4 – Detalle del proceso de estimación (adaptada de [47]).

5.3. Modelos de Productividad

5.3.1. Introducción a los Modelos de Productividad

Los modelos matemáticos típicamente son construidos utilizando datos de proyectos terminados. Por lo tanto, la mayor parte de los llamados "modelos de estimación" en la literatura, son en realidad modelos de productividad [47].

Un modelo de productividad de software puede utilizar fórmulas derivadas empíricamente para estimar el esfuerzo como una función de líneas de código o puntos de función. Un modelo de productividad común utiliza un análisis de regresión sobre los datos históricos de proyectos de software. Un ejemplo es el modelo COCOMO [5] básico que calcula el esfuerzo (y el costo) del desarrollo de software en función del tamaño del programa, expresado en líneas código.

Algunos de los beneficios de los modelos de productividad basados en proyectos pasados son las siguientes [47]:

- La eficiencia de estos modelos puede ser analizada y descrita.
- Cualquiera puede utilizar estos modelos para estimar futuros proyectos.
- Siempre que se ingresa la misma información al modelo, se tendrá como resultado el mismo número.

Por lo tanto, los modelos de productividad son modelos de proyectos pasados construidos a partir de información conocida con [47]:

- Variables cuantitativas basadas en mediciones precisas.
- Variables cuantitativas recolectadas durante el ciclo de vida del proyecto.
- Otras variables descriptivas.

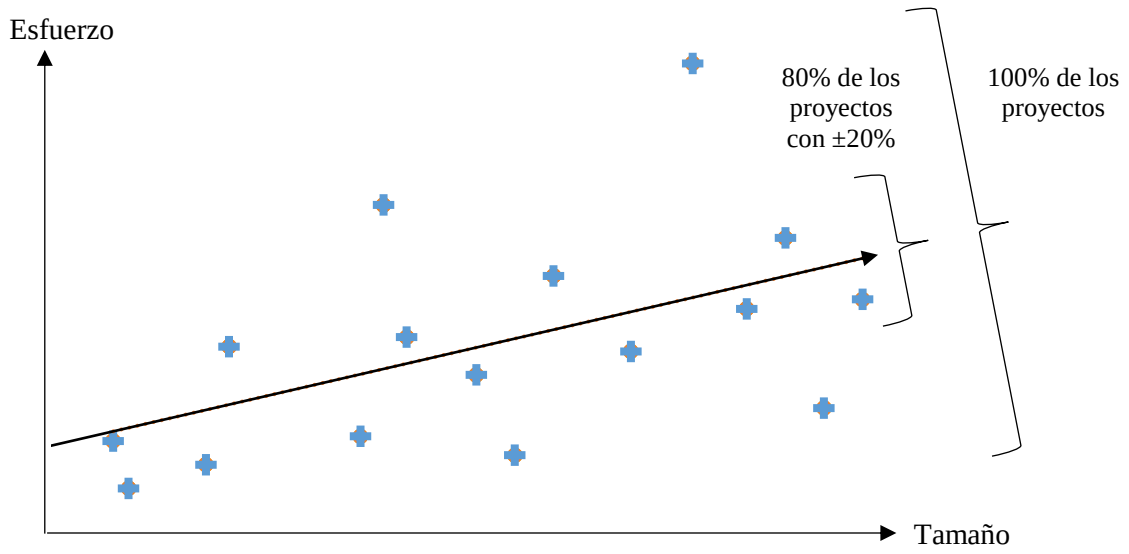
5.3.2. Incertidumbre en un modelo de productividad

Un ejemplo de una representación gráfica de un modelo de productividad (con base en proyectos completados) es representado en la Gráfica 5.2 donde el eje 'x' representa el tamaño de los proyectos y el eje 'y' representa el esfuerzo requerido de los proyectos.

El modelo de productividad representa la relación entre las dos variables (tamaño y esfuerzo), es decir, entre la variable independiente (el tamaño del software) y la variable dependiente (el esfuerzo del proyecto completado).

En Gráfica 5.2 la se puede observar que la mayor parte de los datos no entran exactamente en la ecuación matemática, pero si a cierta distancia de ella. Esto significa que el modelo de productividad no modela exactamente la relación tamaño-esfuerzo, algunos datos están cerca de la línea, mientras que otros datos están muy separados, a pesar de no haber incertidumbre en las entradas al proceso de estimación.

Esto es, 80% de los proyectos están a 20% de distancia de la línea, y 20% de los proyectos fuera de esta distancia.



Gráfica 5.2 – Incertidumbre en un modelo de productividad (adaptada de [47]).

5.3.3. Modelo de regresión lineal

Uno de los principales objetivos de la ciencia estadística es el "resumir" datos, dada una base de datos de proyectos completados se plantea generar información precisa que nos proporcione una descripción de su contenido, con ello podemos tener una representación del comportamiento general de los datos. Existen diversas medidas iniciales que dan representaciones básicas pero importantes sobre los datos, por medio de ellas podemos anticipar el comportamiento de los datos sin necesidad de detenernos en la revisión individual de cada uno de ellos.

La idea anterior nos lleva a la creación de modelos que toman en cuenta la relación entre variables, pero que además nos dan el patrón para cuantificar qué tanto cambia una variable cuando la otra cambia. Esto es, un modelo lineal que nos permita conocer la influencia directa de una variable sobre otra, cuando previamente se ha establecido una relación.

Los modelos lineales son fundamentales para la práctica de la estadística, son parte del conocimiento básico esperado de cualquier estadística aplicada. Los modelos lineales son la base de una amplia gama de metodologías estadísticas [57].

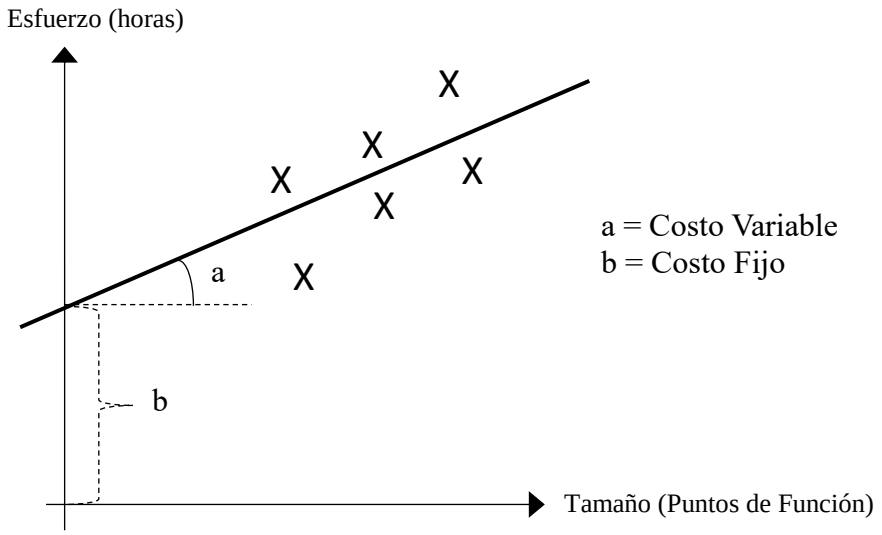
El modelo de regresión lineal [49] es uno de los casos más comunes de modelos lineales, el cual establece una relación lineal entre dos variables, la variable '*x*' y la variable '*y*'.

En este caso '*y*' es la variable dependiente o variable respuesta, pues establecemos que un cambio en *'x'* tendrá un efecto en '*y*', es decir que depende de '*x*', ésta es la variable regresora o independiente:

$$y = \alpha + \beta x$$

Dado que el modelo viene dado por una recta, la variable ‘x’ representa el tamaño funcional mientras que la variable ‘y’ representa el esfuerzo que se calcula a partir del tamaño funcional, los coeficientes tendrán justamente las interpretaciones matemáticas correspondientes, *‘α’* representa la ordenada al origen, la cual nos da el valor para ‘*y’* cuando ‘*x’* vale cero, podríamos tomarlo entonces como un valor inicial o costo fijo, ‘*β’* representa la pendiente de la recta e indica la cantidad que crecerá (o decrecerá) la variable respuesta en términos de un incremento unitario de la variable independiente.

En la Gráfica 5.3 se muestra el objetivo de un modelo de regresión lineal con fines de estimación. Podemos observar que el modelo de regresión lineal ajustado al conjunto de puntos involucra dos parámetros fundamentales el primero llamado “b” que se interpreta como un costo fijo inherente al fenómeno modelado y el cual está incluido siempre, es decir es una condición inicial. El segundo llamado “a” que representa el costo variable de acuerdo al tamaño de la variable que se está modelando.



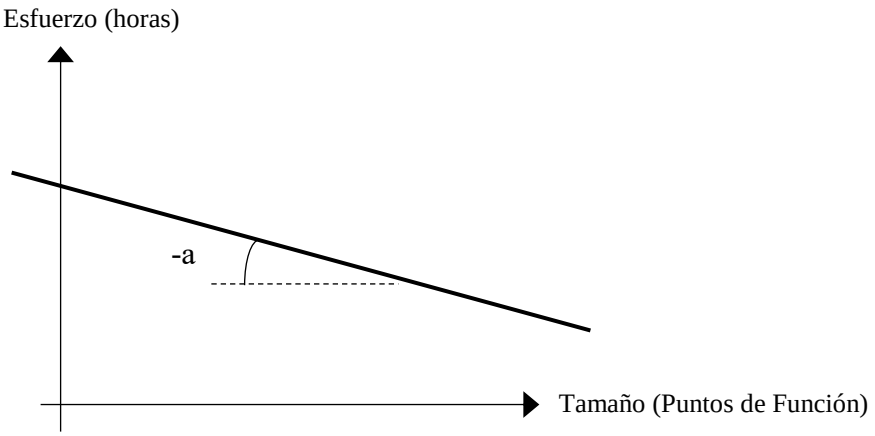
Gráfica 5.3 - Modelo de productividad con costos variables y fijos [47].

En el modelo de regresión lineal, la ecuación toma la siguiente forma cuantitativa:

Esfuerzo = a x Tamaño Funcional + b

En algunas ocasiones, podría haber modelos en los que el costo total disminuye con el aumento de las entradas. En este caso, la pendiente del modelo sería negativa (Gráfica 5.4). Siempre que se observen tales modelos:

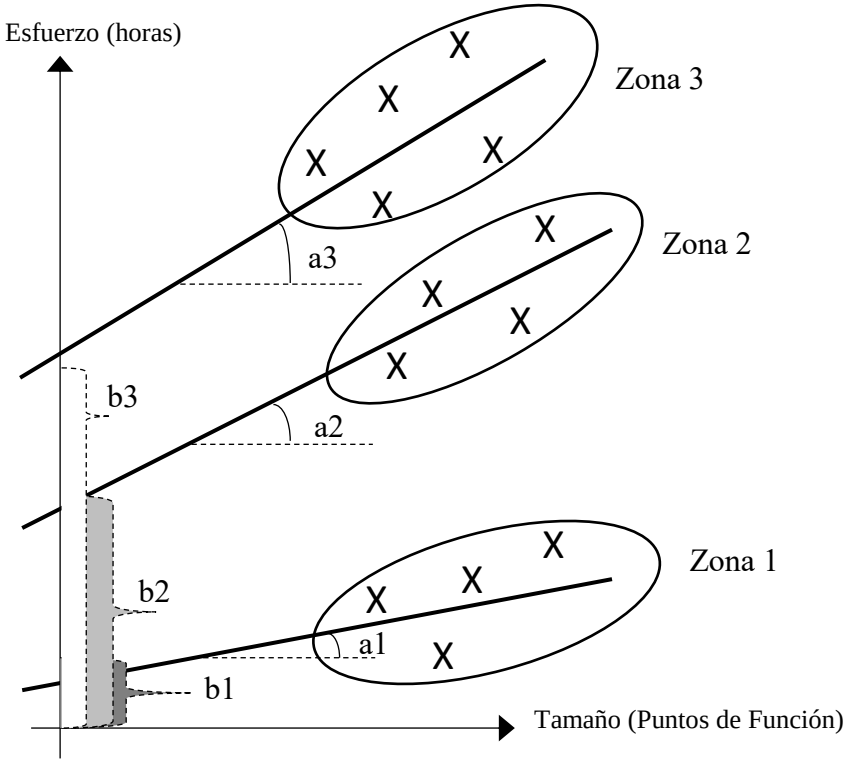
- Los estimadores deben verificar la calidad de la entrada de datos a sus modelos.
- Los estimadores deben tener cuidado de no utilizar el modelo para rangos de valores negativos, para estos rangos, los valores estimados carecen de sentido.



Gráfica 5.4 - Modelo de productividad con pendiente negativa [47].

Cuando un modelo de productividad no puede modelar de la mejor manera todas las circunstancias posibles de datos de entrada, puede ser necesario identificar más de un modelo de productividad para el mismo conjunto de datos. Los subconjuntos de datos para generar los diferentes modelos de productividad pueden tener algunas características en común como el contexto de desarrollo (arquitectura, lenguaje de programación, sistema operativo, etc.).

La Gráfica 5.5 muestra un ejemplo de un conjunto de datos en el que se identificaron tres diferentes zonas (subconjuntos de datos), cada una con su respectiva recta de regresión lineal, su costo variable y costo fijo.



Gráfica 5.5 - Tres subconjuntos (zonas) con su costo variable y costo fijo [47].

5.3.4. Cómo pasar de un modelo de productividad a un modelo de estimación

Una vez generados los modelos de productividad, es posible representar la relación entre las variables independientes (el tamaño del software) y la variable dependiente (esfuerzo final del proyecto) para proyectos pasados [47].

Para determinar los modelos de estimación, es necesario evaluar la calidad de los modelos de productividad relacionados con un conjunto distinto de proyectos que no fueron incluidos para generar los modelos de productividad.

Para esta cuestión, se pueden utilizar la mitad de los proyectos históricos para construir el modelo de productividad y la otra mitad para ser evaluados con el fin de obtener una estimación y poder evaluar la calidad del modelo.

Se debe tomar en cuenta que una vez que se construye un modelo de productividad, como un modelo de regresión lineal, es necesario validar que tan bien ajustado está el modelo, es decir, se debe validar que tan fuerte es la relación entre la variable independiente y la variable dependiente. Una de las primeras medidas a este criterio es el llamado coeficiente de determinación, conocido también como R^2:

Coeficiente de determinación (**R^2**) [48]: Es la proporción de la suma total de los cuadrados de la variable dependiente por las variables independientes en el modelo.

$$R^2 = SS_R / SS_T$$

donde SS_T es el total de la suma de los cuadrados y SS_R es la regresión de la suma de los cuadrados. Es decir:

$$R^2 = \frac{(n\sum xy - (\sum x)(\sum y))^2}{(n(\sum x^2) - (\sum x)^2)\ (n(\sum y^2) - (\sum y)^2)}$$

Una R^2 cerca de 1 indica que la existe una fuerte relación entre la variable independiente y la variable dependiente. Un valor de R^2 cercano a 0 estaría hablando de un modelo poco adecuado pues la variabilidad del error sería muy grande.

5.3.5. Criterios de calidad para modelos de estimación

Para un modelo de estimación generado a partir de un modelo de productividad (como un modelo de regresión lineal), existe una serie de los criterios de evaluación de calidad conocidos de tales modelos estadísticos, tales como:

1. Magnitud del error relativo (Magnitude of relative error - **MRE**) [47]: Indica la divergencia entre los valores estimados por el modelo y los valores reales, expresados en porcentaje. MRE corresponde al valor absoluto de la diferencia entre el esfuerzo conocido del proyecto (Actual) y el valor calculado por el modelo de estimación (Estimado) dividido por el esfuerzo conocido del proyecto (Actual):

   $$MRE = | (Actual - Estimado) / Actual |$$

2. Raíz de la media de los cuadrados (Root of the mean square - **RMS**) [47]: Define la desviación estándar del MRE (SDMRE):

$$RMS = \frac{1}{n} \sum (\text{Actual}_i - \text{Estimado}_i)^2$$

3. Nivel de predicción del modelo (**PRED**) [47]: El nivel de predicción de un modelo de estimación indica la proporción de proyectos que son menores o iguales a un porcentaje establecido respecto al MRE:

$$PRED(l) = k / n$$

donde k es el número de proyectos en el conjunto de datos de tamaño n para los cuales su MRE $\leq l$.

Una vez calculados los criterios de calidad de un modelo de estimación, se deben verificar de acuerdo a lo siguiente:

- Un MRE cercano a 0 indica que el esfuerzo estimado es muy cercano al esfuerzo real del proyecto, mientras que un MRE distante de 0 indica que la diferencia entre el esfuerzo estimado y el esfuerzo real es demasiado alta.
- Un RMS cercano a 0 indica que la desviación estándar de los MRE de los proyectos es muy baja y por lo tanto se tiene un modelo de estimación adecuado. Un RMS distante de 0 indica que la desviación estándar de los MRE de los proyectos es alta y por lo tanto el modelo de estimación es inadecuado.
- Un nivel de predicción (PRED) cercano al 100% indica que los proyectos evaluados tienen un MRE menor (adecuado) al porcentaje establecido (l), mientras que un nivel de predicción cercano a 0 indica que los proyectos evaluados tienen un MRE mayor (inadecuado) al porcentaje establecido (l).

Por lo tanto, los resultados esperados idealmente de un modelo de estimación adecuado son:

- Un MRE bajo (cerca de 0).
- Un RMS bajo (cerca de 0).
- Un nivel de predicción (PRED) alto (cerca de 100%).

5.4. Ejemplos

5.4.1. Análisis de resultados de aproximación y medición de tamaño funcional con COSMIC

- Análisis de resultados por casos de uso comparando la medición con los métodos de aproximación AUC y EPCU a 44 (sección 3.5.2, 3.5.4 y 4.4.3).

Caso de Uso	COSMIC CFP v4.0.1	AUC CFP v4.0.1	MRE	EPCU a 44 CFP v4.0.1	MRE
Iniciar Sesión	3	25.33	744%	2.85	5%
Administración de avisos de venta de vehículos	33	25.33	23%	26.36	20%
MMRE			384%		13%
SDMRE			510%		11%

La SDMRE con el modelo EPCU a 44 es del 11% y el MMRE del 13%, presentando mejores resultados que AUC con SDMRE del 510% y el MMRE del 384%.

- Análisis de resultados por procesos funcionales comparando la medición con los métodos de aproximación AFP, FSC, ESB y EPCU a 16.4 (sección 3.5.2, 3.5.4 y 4.4.3).

Proceso Funcional	COSMIC CFP v4.0.1	AFP CFP v4.0.1	MRE	EPCU a 16.4 CFP v4.0.1	MRE	FSC CFP v4.0.1	MRE	ESB CFP v4.0.1	MRE
Iniciar Sesión	3	7.6	153%	2.62	13%	5	67%	4.3	43%
Buscar aviso de venta	10	7.6	24%	6.76	32%	5	50%	4.3	57%
Registrar aviso de venta	11	7.6	31%	8.95	19%	10	9%	7	36%
Actualizar aviso de venta	7	7.6	9%	6.89	2%	10	43%	7	0%
Cancelar aviso de venta	5	7.6	52%	3.49	30%	5	0%	4.3	14%
MMRE			54%		19%		34%		30%
SDMRE			58%		13%		28%		23%

La SDMRE con el modelo EPCU a 16.4 es del 19% y el MMRE del 13%, presentando mejores resultados que los demás métodos de aproximación, mientras que el método AFP presenta los resultados más desviados con una SDMRE del 54% y el MMRE del 58%.

5.4.2. Conjunto de datos 3 para modelos de estimación

Conjunto de Datos 3: Información de 40 proyectos con la medición de tamaño funcional (CFP v4.0.1) con COSMIC y el esfuerzo (HH) requerido para desarrollar todo el proyecto.

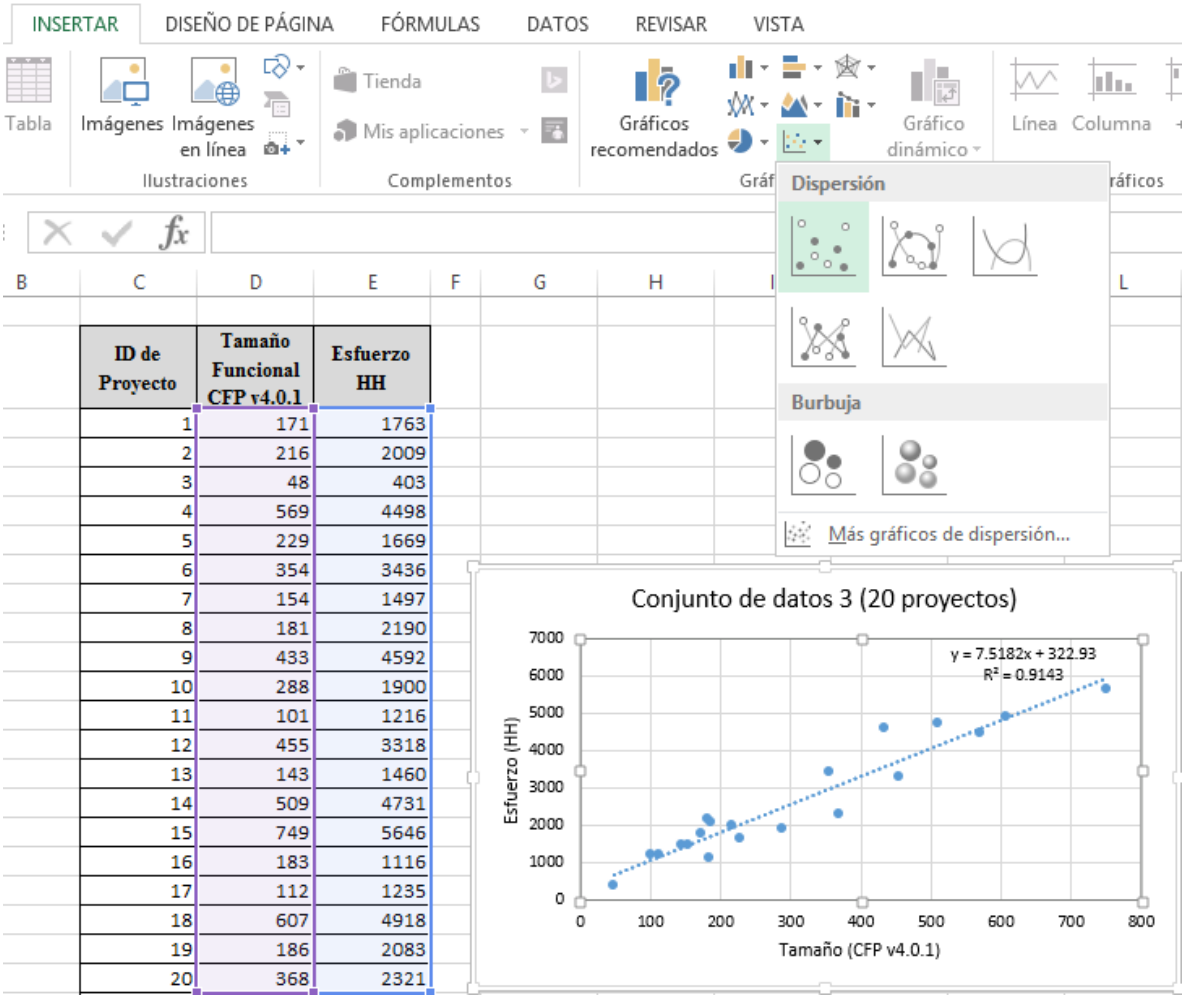
ID de Proyecto	Tamaño Funcional CFP v4.0.1	Esfuerzo HH
1	171	1,763
2	216	2,009
3	48	403
4	569	4,498
5	229	1,669
6	354	3,436
7	154	1,497
8	181	2,190
9	433	4,592
10	288	1,900
11	101	1,216
12	455	3,318
13	143	1,460
14	509	4,731
15	749	5,646
16	183	1,116
17	112	1,235
18	607	4,918
19	186	2,083
20	368	2,321
21	283	3,202
22	213	1,473
23	67	798
24	83	522
25	49	488
26	124	954
27	467	4,718
28	41	464
29	420	3,821
30	188	1,428
31	464	4,224
32	545	4,686
33	195	1,973
34	384	3,187
35	596	4,467
36	234	1,498
37	158	1,514
38	319	2,140
39	81	682
40	373	2,611

5.4.3. Identificación de modelo de productividad con regresión lineal del conjunto de datos 3

Para la generación del modelo de productividad con regresión lineal se utilizará el 50% de la muestra (los primeros 20 proyectos), el otro 50% se utilizará para evaluar la calidad del modelo (sección 5.4.4)

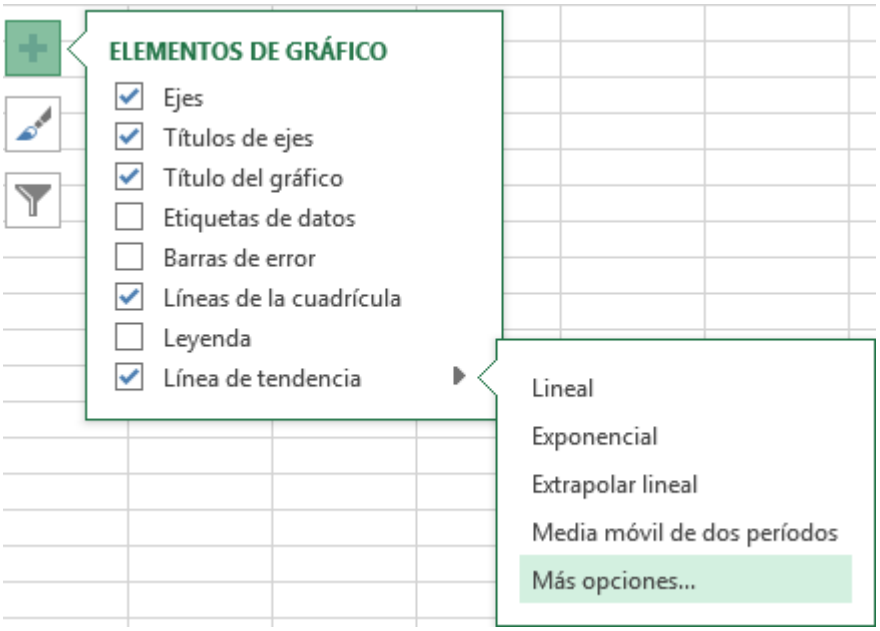
Excel es una manera sencilla de generar la ecuación de la recta de regresión lineal:

1. Agregar en una tabla los valores del tamaño funcional (x) y valores del esfuerzo (y).
2. Seleccionar todos los valores agregados en el paso 1 y seleccionar:
   - Insertar → Gráficos → Dispersión

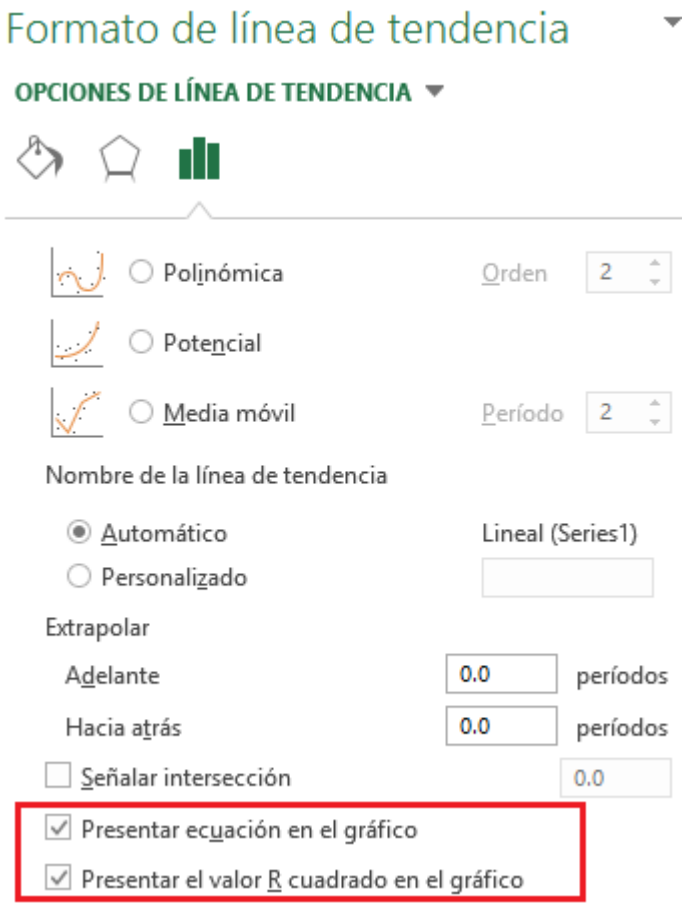


ID de Proyecto	Tamaño Funcional CFP v4.0.1	Esfuerzo HH
1	171	1763
2	216	2009
3	48	403
4	569	4498
5	229	1669
6	354	3436
7	154	1497
8	181	2190
9	433	4592
10	288	1900
11	101	1216
12	455	3318
13	143	1460
14	509	4731
15	749	5646
16	183	1116
17	112	1235
18	607	4918
19	186	2083
20	368	2321

3. Una vez generada la gráfica, corroborar que en la sección de “Elementos de gráfico” estén habilitados los siguientes campos:



4. Seleccionar “Más opciones…” en el campo de línea de tendencia y habilitar:
   - Presentar ecuación en el gráfico
   - Presentar el valor R cuadrado en el gráfico



Una vez generada la gráfica, hemos obtenido la ecuación de la recta y la R^2:

- Ecuación de la recta: y = 7.5182x + 322.93
- Costo fijo = 322.93
- Costo variable = 7.5182
- R^2 = 0.9143

5.4.4. Evaluación de criterios de calidad del modelo de productividad con regresión lineal del conjunto de datos 3

Una vez generado el modelo de productividad utilizando el 50% de la muestra (los primeros 20 proyectos), el otro 50% se utilizará para evaluar la calidad del modelo utilizando los criterios de calidad MMRE, SDMRE y PRED(25%).

ID de Proyecto	Tamaño Funcional CFP v4.0.1	Esfuerzo Real HH	Esfuerzo Estimado HH	MRE
21	283	3,202	2,453	23%
22	213	1,473	1,928	31%
23	67	798	823	3%
24	83	522	946	81%
25	49	488	694	42%
26	124	954	1,254	31%
27	467	4,718	3,835	19%
28	41	464	632	36%
29	420	3,821	3,480	9%
30	188	1,428	1,736	22%
31	464	4,224	3,813	10%
32	545	4,686	4,419	6%
33	195	1,973	1,792	9%
34	384	3,187	3,210	1%
35	596	4,467	4,801	7%
36	234	1,498	2,083	39%
37	158	1,514	1,509	0.004%
38	319	2,140	2,724	27%
39	81	682	933	37%
40	373	2,611	3,127	20%
		MMRE		**22.65%**
		SDMRE		**19.24%**
		PRED(25%)		**60%**

Para obtener el *esfuerzo estimado* de cada proyecto utilizando el modelo de estimación, solo hay que sustituir en la ecuación de la recta la ‘x’ por el tamaño funcional y tener en cuenta los criterios de calidad del modelo de estimación.

- Ecuación de la recta: y = 7.5182x + 322.93

5.5. Ejercicios

5.5.1. Cuestionario de autoevaluación

5.1. ¿Cuál es la técnica de estimación que se utiliza con mayor frecuencia en la industria?

5.2. Menciona las entradas de un proceso de estimación de software:

5.3. ¿Cuáles son las fases del proceso de estimación?

5.4. Menciona 3 beneficios de los modelos de productividad basados en proyectos pasados

5.5. En un modelo de regresión lineal, ¿qué representa la pendiente de la recta?

5.6. Menciona tres desventajas del juicio de experto:

5.7. ¿Qué se entiende por modelos de estimación no algorítmicos?

5.8. Menciona tres criterios de calidad para los modelos de estimación:

5.9. ¿En qué consiste el razonamiento basado en casos?

5.10. ¿Cuál es la salida esperada de un proceso de estimación?

5.11. Responde verdadero o falso las siguientes afirmaciones:

- Una estimación es una predicción que tiene la misma probabilidad de estar por encima o por debajo del valor real.
- Los modelos de estimación basados en analogía hacen estimaciones individuales sobre los componentes del proyecto.
- En los métodos de estimación a priori no hay incertidumbre y se conoce toda la información.
- Cuando una organización ha recolectado sus propios datos, generado y documentado la calidad de sus modelos de estimación, entonces ha desarrollado credibilidad para las estimaciones.
- Es responsabilidad del medidor de software tomar decisiones sobre el presupuesto del proyecto.
- La mayor parte de los llamados “modelos de estimación” en la literatura, son en realidad modelos de productividad.
- El modelo de regresión lineal establece una relación lineal entre dos variables, que pueden ser tamaño (x) y esfuerzo (y).
- En un modelo de regresión lineal, la ordenada al origen se interpreta como un costo fijo inherente al fenómeno modelado y el cual está incluido siempre.
- Una R^2 cerca de 1 estaría hablando de un modelo de regresión poco adecuado pues la variabilidad del error sería muy grande.
- Un MRE cercano a 0 indica que el esfuerzo estimado por el modelo es muy cercano al esfuerzo real del proyecto.

5.5.2. Conjunto de datos 4 para modelos de estimación

Genera un modelo de estimación con regresión lineal con el 50% de los proyectos del conjunto de datos 4 y evalúa la calidad del modelo utilizando los criterios de calidad MMRE, SDMRE y PRED(25%).

Conjunto de Datos 4: Información de 48 proyectos con la medición de tamaño funcional (CFP v4.0.1) con COSMIC y el esfuerzo (HH) requerido para desarrollar todo el proyecto.

ID de Proyecto	Tamaño Funcional CFP v4.0.1	Esfuerzo HH
1	201	2,373
2	21	244
3	526	3,892
4	62	685
5	338	3,346
6	498	3,533
7	266	2,390
8	337	3,837
9	341	3,374
10	636	4,197
11	355	2,204
12	129	1,158
13	300	3,626
14	113	1,006
15	141	1,493
16	145	1,301
17	122	782
18	191	2,157
19	118	1,046
20	87	827
21	88	1,020
22	434	4,773
23	327	3,568
24	114	1,210

ID de Proyecto	Tamaño Funcional CFP v4.0.1	Esfuerzo HH
25	545	3,814
26	96	723
27	180	1,401
28	397	4,761
29	432	3,756
30	82	723
31	81	774
32	230	1,930
33	163	1,403
34	129	1,355
35	506	4,048
36	241	2,000
37	25	299
38	324	2,820
39	546	4,205
40	122	1,047
41	310	3,700
42	326	3,520
43	240	2,787
44	302	3,534
45	399	2,830
46	364	3,057
47	390	3,039
48	561	4,433

6. Evaluación de Desempeño de Proyectos

Este sexto capítulo trata sobre el seguimiento de proyectos de software. Una vez que se ha estimado el esfuerzo, tiempo y/o costo del proyecto (capítulo 5) podemos llevar a cabo el seguimiento (monitoreo) del proyecto mediante metodologías de gestión de proyectos de software.

La primera parte del capítulo trata sobre las principales restricciones que afectan el rendimiento del proyecto. El resto del capítulo describe las metodologías de gestión de proyectos para poder darle seguimiento y control a las principales restricciones en los proyectos software.

El objetivo de este sexto capítulo es dar una introducción a las metodologías de gestión de proyectos de software que ayudan al seguimiento y avance de los proyectos para lograr llevar un control del proyecto a lo largo del ciclo de vida del mismo.

Este capítulo termina con una serie de ejemplos y ejercicios propuestos para poner en práctica lo aprendido e investigar más sobre el tema.

La información que se presenta en este capítulo fue recopilada principalmente de la “Guía PMBOK” [50] y de algunos libros y artículos que describen las metodologías que se abordan en este capítulo.

6.1. Introducción a la Evaluación de Desempeño de Proyectos

6.1.1. Triángulo de hierro

Tres de las principales restricciones en los proyectos son el *alcance* (tamaño), el *costo* (dinero) y el *tiempo* (calendario). Estas restricciones se conocen como el "triángulo de gestión de proyectos" también llamado "triple restricción" o "triángulo de hierro" en la gestión de proyectos, aunque no representan todos los factores que afectan el rendimiento del proyecto, son restricciones importantes para poder llevar a cabo cualquier proyecto [58].

Las partes del triángulo de hierro son:

1. **Alcance** [50]: rasgos y funciones que caracterizan a un producto, servicio o resultado.
2. **Tiempo** [50]: también conocido como calendario, es un modelo que presenta actividades vinculadas con fechas planificadas, duraciones, objetivos y recursos.
3. **Costo** [50]: dinero real incurrido por el trabajo llevado a cabo en una actividad durante un período de tiempo específico.

Hay que aclarar que el Project Management Institute (PMI) hace una distinción entre el alcance del producto y alcance del proyecto, el PMBOK [59] define el Alcance del Producto como "las características y funciones que caracterizan un producto, servicio o resultados", mientras que el Alcance del Proyecto se define como "el trabajo que se debe realizar para entregar un producto, servicio o resultado con las características y funciones especificadas". Teniendo en cuenta esta clasificación, es posible identificar dos tipos de alcance en el desarrollo de software, un alcance del producto, que está relacionado con el producto de software y el alcance del proyecto que está relacionado con el éxito del proyecto incluyendo el alcance del producto.

Algunos autores han investigado la incorporación de la calidad, tales como Yaping Wang que introdujo el concepto de costo de calidad, que se refiere a "todos los costos incurridos para asegurar y mejorar la calidad del producto y todas las pérdidas que no cumplan los estándares de calidad" como se hace referencia en [58].

El triángulo de hierro (Figura 6.1) muestra los elementos fundamentales que deberá controlar un administrador de proyectos, ya que son ellos los que determinarán al final si el proyecto ha sido exitoso o por el contrario el proyecto fracasó, cualquier modificación en uno de los parámetros implica la modificación de alguno(s) de los otros.

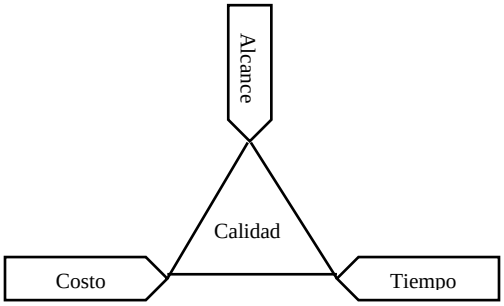


Figura 6.1 - Triángulo de hierro

6.2. Gestión del Valor Ganado o Devengado (Earned Value Management - EVM)

6.2.1. Introducción a la Gestión del Valor Ganado

La Gestión del Valor Ganado o Devengado (Earned Value Management - EVM) [50] es una metodología de gestión de proyectos que combina medidas de alcance, calendario y recursos para evaluar el desempeño y el avance de un proyecto.

EVM surgió como una especialidad de análisis financiero en los programas del Gobierno de los Estados Unidos en la década de 1960, pero desde entonces se ha convertido en una rama significativa de la gestión de proyectos [60].

A principios 1990, EVM surgió como una metodología de gestión de proyectos para ser entendida y utilizada por los gerentes y ejecutivos, no sólo por los especialistas de EVM. Hoy en día, EVM se ha convertido en una parte esencial de seguimiento de proyectos [60].

EVM ha demostrado ser una de las herramientas de medición de desempeño y retroalimentación más eficaz para la gestión de proyectos, ayuda a responder clara y objetivamente [59]:

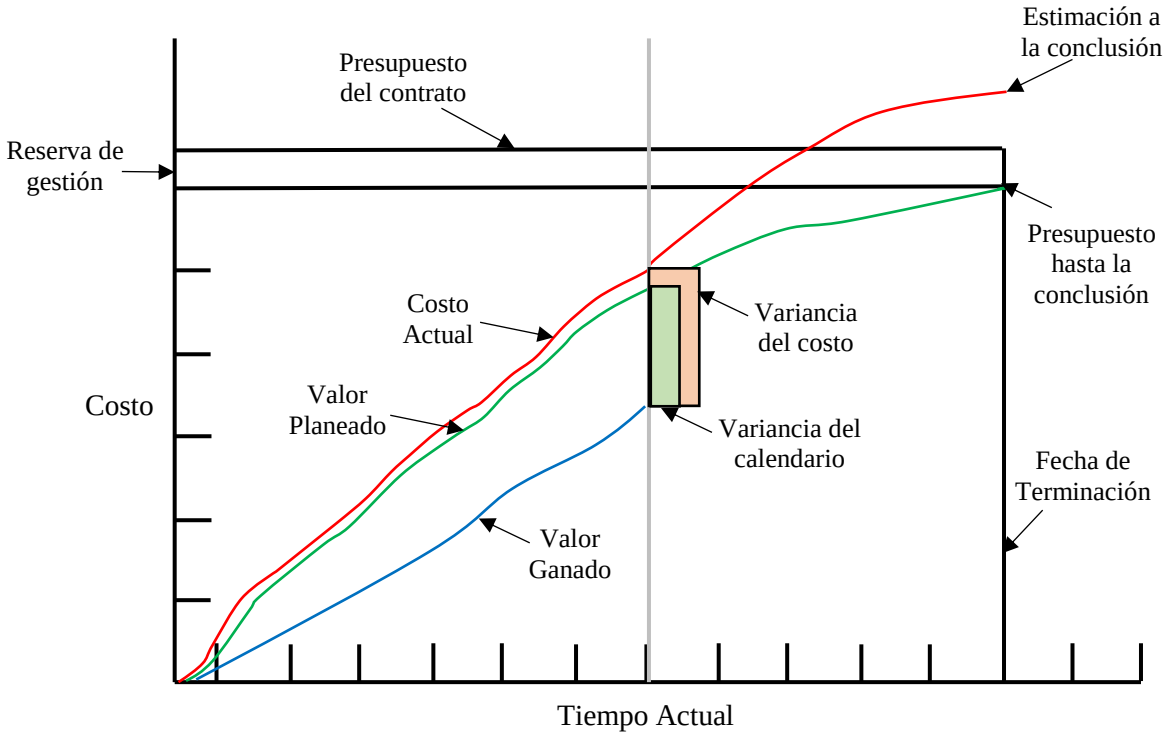
- ¿dónde hemos estado?
- ¿dónde nos encontramos ahora?
- ¿a dónde vamos?

Más detalladamente [50]:

- Es un método muy utilizado para la medida del desempeño de los proyectos.
- Integra la línea base del alcance con la línea base de costos, junto con la línea base del calendario, para generar la línea base para la medición del desempeño, que facilita la evaluación y la medida del desempeño y del avance del proyecto por parte del equipo del proyecto.
- Es una técnica de dirección de proyectos que requiere la constitución de una línea base integrada con respecto a la cual se pueda medir el desempeño a lo largo del proyecto.
- Los principios del EVM se pueden aplicar a todos los proyectos, en cualquier sector.

6.2.2. Fórmulas de la Gestión del Valor Ganado

El concepto de EVM se fundamenta en las curvas de crecimiento, como se puede observar en la Gráfica 6.1, y que se complementa con las fórmulas establecidas en la Tabla 6.1.



Gráfica 6.1 – Gestión del Valor Ganado o Devengado (adaptada de [59])

Termología y Definición del Valor Ganado	Fórmula
Valor Planeado (Planned Value - PV) = Costo Presupuestado del Trabajo Planificado (Budgeted Cost of Work Scheduled - BCWS) El presupuesto autorizado asignado al trabajo planificado que debe realizarse respecto de una actividad del calendario o componente de la estructura de desglose del trabajo.	PV = BCWS
Costo Actual (Actual Cost - AC) = Costo Real del Trabajo Realizado (Actual Cost of Work Performed - ACWP) Costos totales incurridos y registrados para llevar a cabo un trabajo realizado en un período determinado para una actividad del calendario o componente de la estructura de desglose del trabajo.	AC = ACWP
Presupuesto hasta la Conclusión (Budget At Completion - BAC) La suma de todos los valores del presupuesto establecidos para el trabajo que se realizará en un proyecto, componente de la estructura de desglose del trabajo o actividad del calendario.	BAC = Presupuesto hasta la conclusión

Valor Ganado (Earned Value - EV) El valor del trabajo completado al día actual expresado en términos del presupuesto aprobado asignado a dicho trabajo para una actividad del calendario o un componente de la estructura de desglose del trabajo.	EV = BAC (% completado)
Variación del Costo (Cost Variance - CV) Una medida de desempeño en función de los costos de un proyecto. Es la diferencia entre el valor ganado (EV) y el costo real (AC). Si es positiva está debajo del presupuesto, si es negativa se ha gastado más del presupuesto.	CV = EV –AC
Variación del Calendario (Schedule Variance - SV) Una medida de desempeño del calendario en un proyecto. Es una diferencia entre el valor ganado (EV) y el valor planificado (PV). Si es positiva va delante del calendario planeado respecto del tiempo, si es negativa está retrasado respecto del calendario planeado respecto del tiempo.	SV = EV – PV
Índice de Desempeño del Costo (Cost Performance Index - CPI) Una medida de eficiencia en función de los costos de un proyecto. Es la proporción entre el valor ganado (EV) y costos reales (AC). Por cada peso ($) gastado, se está obteniendo x% del valor del peso, si es menor a 1, se está gastando más del presupuesto, es decir se está obteniendo menor valor por cada peso gastado, si es mayor 1, se está gastando menos del presupuesto, es decir se está obteniendo mayor valor por cada peso gastado.	CPI = EV / AC
Índice de Desempeño del Calendario (Schedule Performance Index - SPI) Una medida de eficiencia del calendario en un proyecto. Es la razón entre el valor ganado (EV) y valor planificado (PV). El proyecto progresa a un x% del calendario planeado, si es menor a 1 el proyecto está retrasado respecto del calendario, si es mayor a 1, el proyecto está adelantado respecto del calendario.	SPI = EV / PV
Estimación a la Conclusión (Estimate AT Completion - EAC) El costo total previsto de una actividad del calendario, de un componente de la estructura de desglose del trabajo o del proyecto, cuando se complete el alcance definido del trabajo. El EAC puede ser calculado sobre la base del desempeño hasta la fecha. El resultado representa el costo total esperado actualmente como x pesos.	EAC = BAC / CPI
Estimación hasta la Conclusión (Estimate To Complete - ETC) El costo previsto necesario para terminar todo el trabajo restante para una actividad del calendario, un componente de la estructura de desglose del trabajo o el proyecto.	ETC = EAC –AC

El resultado representa que va costar x pesos completar el Proyecto.	
Variación hasta la Conclusión (Variance At Completion - VAC) La variación de costo al terminar el Proyecto, el resultado representa que el Proyecto va a tener x pesos ($) más o menos del presupuesto original.	VAC = BAC -EAC

Tabla 6.1 - Fórmulas de la Gestión del Valor Ganado o Devengado [59].

6.3. Calendario Ganado o Devengado (Earned Schedule - ES)

6.3.1. Introducción al Calendario Ganado

El Calendario Ganado o Devengado (Earned Schedule - ES) [61] es una extensión de EVM que proporciona a los administradores de proyectos la capacidad de analizar el desempeño del calendario.

Más detalladamente [62]:

- ES no necesita datos adicionales para el cálculo de las medidas, sólo necesita los datos de EVM.
- Los indicadores de desempeño de ES se basan en el tiempo, haciéndolos más fáciles de comprender.
- Los indicadores ES proporcionan un estado y capacidad predictiva para el calendario, análogo para el costo usando EVM.
- Debido a que las métricas ES utilizan medidas basadas en el tiempo, extienden EVM y el análisis de planificación integrado.
- ES es el puente entre EVM y el análisis del calendario.
- ES se puede utilizar para el análisis del calendario detallado y tiene el potencial para mejorar tanto el costo como la predicción del calendario.

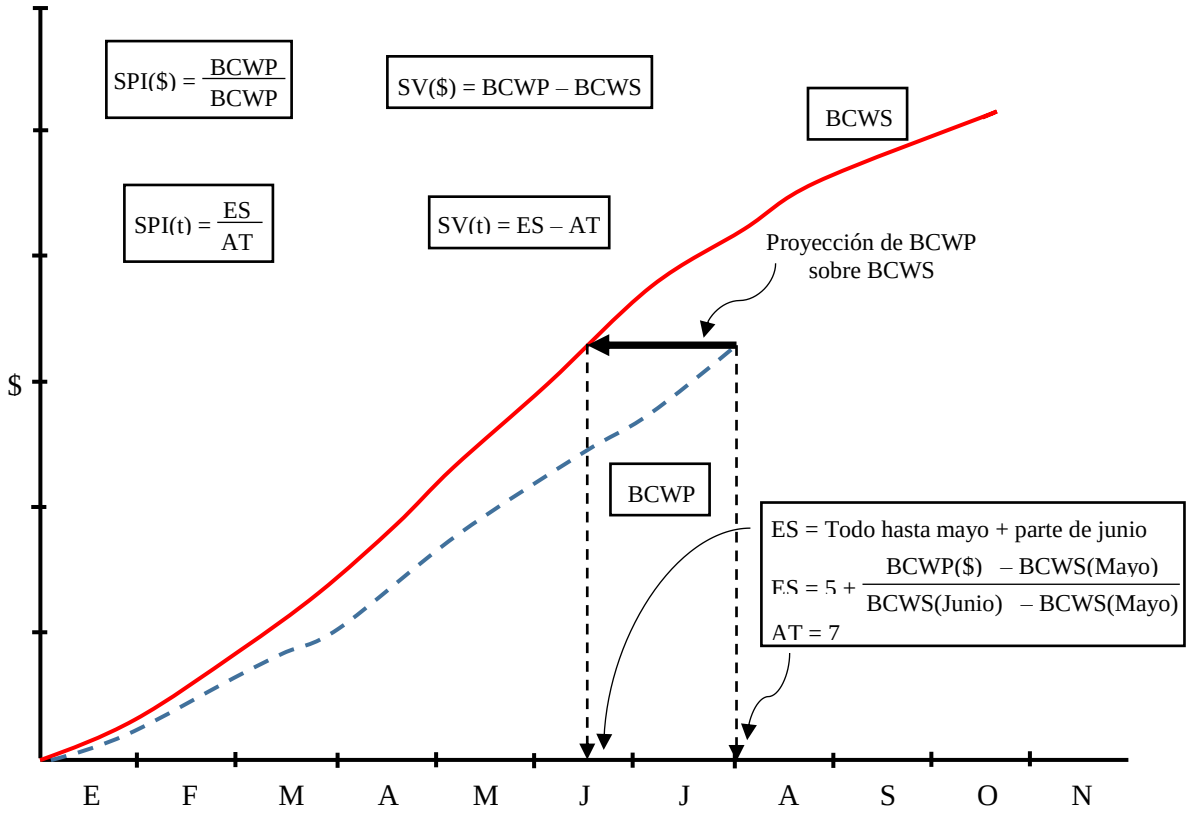
ES resuelve algunas de las deficiencias de EVM:

- Los resultados de EVM son en costo, incluso los indicadores de tiempo se dan en términos de costo.
- Al final del proyecto la variación de calendario (SV) siempre termina en cero y el Índice de Desempeño del Calendario (SPI) termina en uno.

El PMI Practice Standard para EVM publicado en noviembre de 2011 incluye Earned Schedule como el método principal en el apéndice "Schedule Analysis Using EVM Data”.

6.3.2. Fórmulas del Calendario Ganado

El concepto de ES se fundamenta en las curvas de crecimiento, como se puede observar en la Gráfica 6.2, y que se complementa con las fórmulas establecidas en la Tabla 6.2.



Gráfica 6.2 – Calendario Ganado o Devengado (adaptada de [63]).

Tipo	Termología y Definición del Calendario Ganado	Fórmula
Métricas	Calendario Ganado (Earned Schedule - ES) La cantidad de periodos de la línea base para los cuales ya se ha realizado el gasto planeado.	ES = C + I Número de periodos completados (C) más la proporción incompleta (I)
	Tiempo Actual (Actual Time - AT) La cantidad de periodos transcurridos a la fecha de revisión.	AT = Número de periodos ejecutados
	Duración Planeada (Planned Duration - PD) La cantidad de periodos en los que la línea base plantea la terminación del proyecto.	PD = Duración planeada del proyecto

Indicadores	Variación de Calendario (Schedule Variance - SV) La diferencia entre el Calendario Ganado (ES) y el Tiempo Actual (AT), es decir cuánto se ha avanzado más con respecto a la línea base.	SV(t) = ES – AT
	Índice de Desempeño del Calendario (Schedule Performance Index - SPI) Indicador que representa la razón a la que se ha cumplido la línea base con respecto al tiempo real.	SPI(t) = ES / AT
	Índice de Desempeño del Calendario hasta la Conclusión (To Complete Schedule Performance Index - TSPI) Indica la proporción de Cronograma por avanzar con respecto al tiempo restante hasta la fecha fin planeada del proyecto.	TSPI = (PD - ES) / (PD - AT) TSPI = (PD - ES) / (ED - AT)
Predictores	Estimado Independiente hasta la Conclusión (Independent Estimate at Completion - IEAC) Representa una estimación del tiempo que durará el proyecto si se continúa con el mismo comportamiento.	IEAC(t) = PD / SPI(t)
	Variación hasta la Conclusión (Variance At Completion - VAC) Representa una estimación del número de periodos faltantes para terminar el proyecto, suponiendo que se mantiene el mismo comportamiento.	VAC(t) = PD – IEAC(t)

Tabla 6.2 - Fórmulas del Calendario Ganado o Devengado [62].

6.4. Gestión del Alcance Ganado o Devengado (Earned Scope Management – ESM)

6.4.1. Introducción a la Gestión del Alcance Ganado

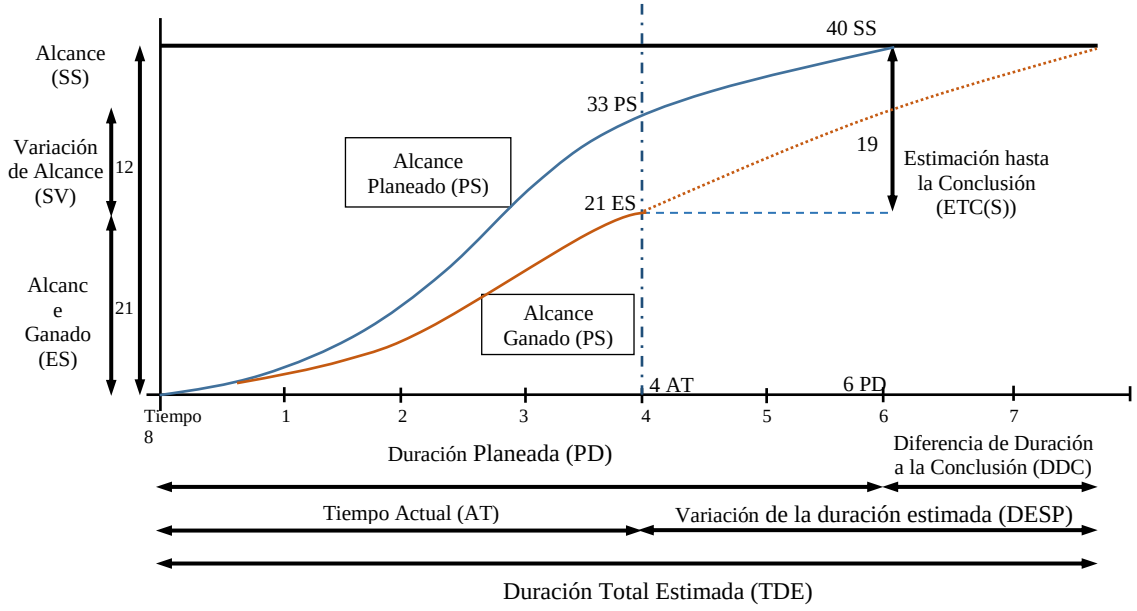
Mientras que las técnicas de EVM y ES gestionan las restricciones de costo y tiempo, no abordan explícitamente la restricción de alcance. Si bien la gestión del alcance se reconoce como un factor clave de éxito en los proyectos de software, existe una falta de técnicas formales para gestionar el alcance de los proyectos [58].

El control del alcance del proyecto asegura que todos los cambios solicitados se procesen a través del control integrado de cambios. La expansión incontrolada del alcance del proyecto sin ajustes de tiempo, costo y recursos se denomina corrupción del alcance (scope creep). Los cambios son inevitables; por lo tanto, es obligatorio para todo proyecto contar con algún tipo de proceso de control de cambios [50].

La Gestión del Alcance Ganado o Devengado (Earned Scope Management - ESM) [58] es una técnica que no sustituye a las otras dos (EVM y ES), por el contrario, las complementa permitiendo controlar el alcance de los proyectos y logrando tener de una manera integrada una evaluación del desempeño de las tres principales restricciones de los proyectos (alcance, tiempo y costo).

6.4.2. Fórmulas de la Gestión del Alcance Ganado

El concepto de ES se fundamenta en las curvas de crecimiento, como se puede observar en la Gráfica 6.3, y que se complementa con las fórmulas establecidas en la Tabla 6.3 [58].



Gráfica 6.3 – Gestión del Alcance Ganado o Devengado

Tipo	Termología y Definición del Alcance Ganado	Fórmula
Datos de Entrada	Tiempo Actual (Actual Time - AT)	AT = Número de periodos ejecutados
	Duración Planeada (Planned Duration - PD)	PD = Duración planeada del proyecto
	Alcance del proyecto (SS) El alcance del Proyecto definido de preferencia en términos de unidades estandarizadas o convencionales. Lo que se ha planeado realizar.	SS = Lo que se ha planeado realizar
	Alcance Ganado (Earned Scope - ES) Alcance ganado en un periodo determinado, se refiere a las unidades estandarizadas o convencionales terminadas en el periodo revisado.	Si el %C es un dato de entrada entonces: ES = SS (% completado)
	Alcance Planeado (Planned Scope - PS)	PS = Lo que se ha planeado hacer
	% completado (%C)	Si el ES es un dato de entrada. %C = ES / SS
	Recursos Humanos usados en el Proyecto (Project Human Resources - PHR) Los recursos humanos utilizados en el período revisado que participaron en alcance previsto.	PHR = Recursos humanos usados en el proyecto
Estado del Alcance	Variación de Alcance (Scope Variance - SV) Una medida de desempeño del alcance en un proyecto. Es la diferencia entre el alcance ganado (ES) y el alcance planificado (PS). Si el resultado es positivo, el alcance del proyecto ha avanzado más que el planeado, si es negativo está retrasado en alcance.	SV = ES – PS
	Índice de Desempeño del Alcance (Scope Performance Index - SPI) Una medida de eficiencia del alcance en un proyecto. Es la división entre el alcance ganado (ES) y el alcance planificado (PV). El proyecto progresa a un x% del alcance planeado, si es < 1 el proyecto está retrasado respecto del alcance, si es > 1, el proyecto está adelantado respecto del alcance.	SPI = ES / PS

	Productividad por Recurso (Productivity by Resource - PR) La productividad promedio por persona involucrada en el alcance logrado en el proyecto.	PR = ES / PHR
	Productividad hasta la Conclusión (Productivity to Complete - PTC) Es la productividad requerida por los recursos definidos para completar el alcance del proyecto como fue planeado.	PTC = (PSn-1 – PSn) / PHR
	Productividad Promedio por Recurso Humano (Average Productivity by Human Resource - PROAVG) La productividad promedio en el periodo revisado.	PROAVG = AVG(PRn)
	Variación de la Productividad (Productivity Variation - PV) Una medida de desempeño de la productividad en un proyecto. Es la diferencia entre la productividad por recurso y la productividad hasta la conclusión. Si el resultado es positivo se está teniendo una productividad mayor de la requerida, si es negativa se está teniendo una productividad menor que la requerida.	PV = PR - PTC
	Estimación hasta la Conclusión (Estimate To Complete – ETC(s)) El alcance previsto necesario para terminar todo el trabajo restante del proyecto. El resultado representa que falta completar x unidades convencionales de alcance para completar el proyecto.	ETC(s) = SS - ES
Predicción del Alcance	Duración Total Estimada (Total Duration Estimate - TDE) La duración estimada para completar el proyecto con la misma productividad por recurso para los próximos periodos.	TDE = AT + ETC(s) / (PR * PHR)
	Variación de la Duración Estimada. (Estimate Duration Variation - DESP) Una medida de desempeño del tiempo en un proyecto. Es la diferencia entre la Duración Total Estimada y el Periodo Actual hasta la conclusión. Representa la variación en tiempo respecto de la duración estimada, considerando la productividad por recurso actual.	DESP = TDE - AT
	Diferencia de Duración a la Conclusión (Duration Difference at Completion – DDC) La diferencia entre la Duración Planeada y la Duración Total Estimada en el período evaluado.	DDC = PD - TDE

	Productividad Requerida por Recursos definida para Completar el Proyecto como se Planeó (Productivity required by resources defined to complete the scope as was planned - PRTC). Representa la productividad que se necesita considerando los recursos definidos, para completar el proyecto como fue planeado. Si es >1, se requiere incrementar la productividad, si es < a 1, quiere decir que se debe de disminuir la productividad.	PRTC = (ETC / PHR) / (PD - AT)
	Variación de Recursos para completar el Alcance planeado con la misma Productividad (Resources Variation to Complete Planed Scope by period - RVTC) Representa la diferencia de los Recursos Humanos necesarios para completar el alcance restante en los periodos restantes considerando la productividad por recurso en el periodo evaluado.	RVTC = (-SV) / PR
	Recursos Humanos Necesarios para Completar el Proyecto (Human Resources Need to Complete the Project - RNTC) Representan los Recursos Humanos Estimados para los periodos restantes, para completar el alcance como fue planeado.	RNTC = RVTC + PHR

Tabla 6.3 - Fórmulas de la Gestión del Alcance Ganado o Devengado [58].

6.5. Ejemplos

6.5.1. Ejemplo 1 de Earned Value Management

Tenemos un proyecto para ejecutar en 4 semanas y el presupuesto es de $100,000. Nos informan que al finalizar la tercera semana se ha completado sólo el 50% del trabajo, de acuerdo al calendario se debía haber realizado el 75%, también que los gastos actuales ascienden a $90,000. ¿Cuál será el costo total del proyecto?

En este ejemplo, el presupuesto hasta la conclusión es de $100,000, se ha realizado el 50% de progreso del proyecto y se han gastado $90,000. De acuerdo al calendario se debía haber realizado el 75%

- **BAC** = $100,000
- **AC =** $90,000
- **PV** = %planeado x presupuesto del proyecto = 75% x $100,000 = $75,000

El valor ganado es igual al porcentaje actual completado por el presupuesto del proyecto.

- **EV =** %completado x BAC = 50% x $100,000 = $50,000

La varianza del costo es la diferencia entre los costos actuales del trabajo realizados y el presupuesto del proyecto. La varianza del cronograma es la diferencia del progreso logrado con respecto al cronograma del proyecto.

- **CV =** EV - AC = $50,000 - $90,000 = -$40,000
- **SV =** EV - PV = $50,000 - $75,000 = $25,000

Como la variancia es negativa el proyecto está retrasado en el calendario y estará por encima del presupuesto al final, por lo que se necesita extender el calendario del proyecto y/u obtener fondos adicionales para completar el proyecto.

El índice de desempeño en costo y tiempo son muy útiles para comunicar una valoración objetiva del estado del proyecto.

- **CPI =** EV / AC **=** $50,000 / $90,000 = 0.56
- **SPI =** EV / PV = $50,000 / $75,000 = 0.67

Para calcular la estimación a la terminación y responder a la pregunta de cuál será el costo total del proyecto, se divide el presupuesto original por el índice de desempeño del costo.

- **EAC** = BAC / CPI = $100,000 / 0.56 = **$180,000 = costo total del proyecto**

Por lo tanto el costo necesario para terminar el proyecto se calcula restando la estimación a la conclusión menos el costo actual.

- **ETC** = EAC – AC = $180,000 - $90,000 = $90,000
- **VAC** = BAC – EAC = $100,000 - $180,000 = -$80,000

6.5.2. Ejemplo 1 de Earned Schedule

Calcular el tiempo necesario para terminar el proyecto a partir de la siguiente información obtenida del calendario de un proyecto:

- Duración planeada = 10 meses
- Meses anteriores completados = 7
- Presupuesto autorizado hasta julio = $18,500
- Presupuesto autorizado hasta agosto = $21,350
- Presupuesto gastado hasta agosto = $19,000

Como han transcurrido 7 meses, entonces nos encontramos en el octavo mes.

- **AT =** 8
- **ES =** C + I = 7 + ($19,000 - $18,500) / ($21,350 - $18,500) = 7 + 0.288 = 7.288

Se ha completado el trabajo de 7.288 meses en 8 meses, es decir, se tiene un atraso de 0.712 meses.

- **SV(t) = ES – AT =** 7.288 – 8 = -0.712
- **SPI(t) = ES / AT =** 7.288 / 8 = 0.91 = 91%

Por lo tanto el tiempo estimado independiente hasta la conclusión se calcula como el valor planeado entre el índice de desempeño del calendario.

- **IEAC(t) = PD / SPI(t) =** 10 / 0.91 = **10.98 = duración total del proyecto**

6.5.3. Ejemplo 1 de Earned Scope Management

Se tienen los siguientes datos (datos de entrada) del avance del de desarrollo de un proyecto de software:

Terminología	1er Periodo (mes 1)	2do Periodo (mes 2)	3rd Periodo (mes 3)	Unidad
PD	10.5	10.5	10.5	meses
AT	1	2	3	periodo
SS	254	254	254	CFP
PS	24	48	86	CFP
ES	20	40	87	CFP
SPHR	11	11	11	personas
%completado de alcance	8	16	34	%

Utilizando los datos del avance del desarrollo de la tabla anterior, se calcula el estado del alcance.

Terminología	1er Periodo (mes 1)	2do Periodo (mes 2)	3rd Periodo (mes 3)	Unidad
SV(s)	-4	-8	1	CFP
SPI	83%	83%	101%	%
PR	1.82	1.82	4.27	CFP/persona
PTC	2.18	2.18	3.45	CFP/persona
PROAVG	1.82	1.82	2.64	CFP/persona
PV	-0.36	-0.36	.82	CFP/persona
ETC(s)	234	214	167	CFP

El índice de desempeño de alcance (SPI) es menor al 100% en los dos primeros periodos, lo que significa que se han desarrollado menos CFP que el alcance planeado (de igual manera la variación del alcance).

La productividad por recurso se mantiene constante durante los dos primeros periodos, sin embargo, en el tercer periodo dicha productividad aumenta considerablemente, logrando que en el tercer periodo se haya desarrollado más funcionalidad que el alcance planeado.

La estimación hasta la conclusión (ETC(s)) para el primer periodo es de 234 CFP y decrece a través de los tres primeros periodos, lo que representa que se está completando el alcance del proyecto

De igual manera, utilizando los datos del avance del desarrollo, se calcula la predicción del alcance.

Terminología	1er Periodo (mes 1)	2do Periodo (mes 2)	3rd Periodo (mes 3)	Unidad
TDE	12.70	12.70	6.55	meses
DESP	11.70	10.70	3.55	meses
DDC	-2.20	-2.20	3.95	meses
PRTC	2.24	2.29	2.02	CFP/persona
RVTC	2.20	4.40	-0.23	persona
RNTC	13.20	15.40	10.77	persona

Debido a que una de las principales restricciones en los proyectos es el tiempo, se debe calcular la productividad requerida para terminar a tiempo el proyecto. Por ello, se calcula la productividad por recurso (PRTC) observando que la productividad requerida en el tercer periodo es menor a la requerida en los dos primeros periodos debido a que la productividad en el tercer periodo fue mayor.

La variación de recursos para completar el alcance planeado (RVTC) representa la diferencia de los recursos necesarios para completar el alcance en los periodos restantes,

mostrando, al igual que el RNTC, como en los primeros dos periodos se necesitaban más recursos para completar el alcance del periodo, sin embargo, en el tercer periodo esta variación disminuyó llegando a ser menor a cero.

La duración total estimada (TDE) se mantuvo constante los dos primeros periodos, mostrando una variación (DESP) uniforme, sin embargo, en el tercer periodo hubo un incremento en la productividad reduciendo casi a la mitad la duración total estimada indicando que el proyecto se terminará en 6.55 meses en vez de los 10.5 planeados.

6.6. Ejercicios

6.6.1. Ejercicio 1 de Earned Value Management

Has sido contratado(a) para construir 10 vagones de tren, todos idénticos. Cada vagón es presupuestado a $3,000.00. Tú has calendarizado 12 meses para completar el proyecto. Al final de 9 meses has construido 6 vagones completos y has gastado $20,000.00.

Completa la siguiente tabla y responde la siguiente pregunta:

- ¿Tienes suficiente dinero en el presupuesto original para terminar el proyecto?

Terminología	Cálculos
BAC	
% Completado	
PV	
AC	
EV	
CV	
SV	
CPI	
SPI	
EAC	
ETC	
VAC	

6.6.2. Ejercicio 1 de Earned Schedule

Calcular el tiempo necesario para terminar el proyecto a partir de la siguiente información obtenida del calendario de un proyecto:

- Duración planeada = 24 meses
- Meses anteriores completados = 5
- Presupuesto autorizado hasta el quinto mes = $11,000
- Presupuesto autorizado hasta el sexto mes = $12,550
- Presupuesto gastado hasta el sexto mes = $18,750

Completa la siguiente tabla y responde la siguiente pregunta:

- ¿Tienes suficiente tiempo en el calendario original para terminar el proyecto?

Terminología	Cálculos
PD	
AT	
ES	
SV(t)	
SPI(t)	

TSPI	
IEAC	
VAC	

6.6.3. Ejercicio 1 de Earned Scope Management

Se tienen los siguientes datos (datos de entrada) del avance del de desarrollo de un proyecto de software:

Terminología	1er Periodo (mes 1)	2do Periodo (mes 2)	3rd Periodo (mes 3)	Unidad
PD	12	12	12	meses
AT	1	2	3	periodo
SS	727	727	727	CFP
PS	60	120	180	CFP
ES	60	130	185	CFP
SPHR	7	7	7	personas
%completado de alcance	8	18	25	%

Utilizando los datos del avance del desarrollo de la tabla anterior, completa la tabla del estado del alcance y de la predicción del alcance, y responde la siguiente pregunta:

- ¿Se terminará el proyecto en el tiempo planeado según el calendario?

Estado del alcance

Terminología	1er Periodo (mes 1)	2do Periodo (mes 2)	3rd Periodo (mes 3)	Unidad
SV(s)				CFP
SPI				%
PR				CFP/persona
PTC				CFP/persona
PROAVG				CFP/persona
PV				CFP/persona
ETC(s)				CFP

Predicción del alcance

Terminología	1er Periodo (mes 1)	2do Periodo (mes 2)	3rd Periodo (mes 3)	Unidad
TDE				meses
DESP				meses
DDC				meses
PRTC				CFP/persona
RVTC				persona
RNTC				persona

Conclusiones

La Ingeniería de Software aún está aprendiendo a medir, a estimar y a mejorar la calidad de sus productos y procesos. Establecer el método correcto suele ser muy complicado y lleva tiempo, años o incluso siglos. La determinación exacta de la longitud en la navegación inició con viajes transoceánicos a finales del siglo XV, y fue hasta finales del siglo XVIII cuando se obtuvo una medición adecuada de la longitud (aproximadamente 300 años).

Además, la Ingeniería de Software cuenta con un número limitado de medidas aceptadas por los practicantes y reconocidas como maduras a través de un estándar y con un número pequeño de estudios experimentales rigurosos, a diferencia de las disciplinas maduras que cuentan con un número significativo de estas medidas y estudios.

¿Qué sucede si no utilizamos los estándares de medición que tiene la Ingeniería de Software? La respuesta sería que cada uno mediría cosas diferentes en unidades de medida diferentes, y peor aún, si no medimos no podemos controlar ni dirigir un proyecto.

Actualmente la única característica del software que podemos medir con un estándar es la funcionalidad. A pesar de que existan estándares que sobre la calidad del software, dichos estándares solo indican que atributos podemos medir del software, pero no definen la manera en la que se deben medir éstos. Es por ello por lo que debemos revisar los Métodos de Medición de Tamaño Funcional que reconoce como estándares el ISO/IEC 14143. En este documento se revisó de manera general el método IFPUG ya que es el método de medición más utilizado internacionalmente. Sin embargo, el método de medición que se revisó a detalle fue COSMIC ya que es el único método de segunda generación que se basa en sólidos principios de ingeniería de software y no está basado en estudios estadísticos como los demás métodos, se basa más bien en una representación de las funciones o tareas de cualquier software.

Es generalmente conocido que los requerimientos definen el tamaño de un proyecto (alcance), el cual determina el esfuerzo requerido para desarrollarlo, que a su vez determina la duración y costo del proyecto. Si al aplicar un método de medición que recibe como entrada los requerimientos y da como resultado el tamaño funcional del proyecto, la cuestión sería como transformar dicho tamaño a esfuerzo. Para ello necesitamos utilizar un modelo de estimación.

En los últimos años, se han desarrollado diversos modelos de estimación enfocados a la estimación de esfuerzo. A pesar de ello, la técnica de estimación que se utiliza con mayor frecuencia en la industria es el Juicio de Experto, el problema de ésta es que no se puede replicar ya que la experiencia pertenece a una persona (o varias) y no a la empresa. El reporte del Caos nos dice que en 2015 menos del 30% de los proyectos de software fueron exitosos y más del 50% fueron proyectos subestimados, esto quiere decir que no estamos estimando correctamente.

Lo recomendable es que una empresa genere y utilice sus propios modelos de estimación basados en datos de proyectos pasados, ya que de esta manera se garantiza que el modelo refleja la productividad real de la empresa. Solo hay que validar los criterios de calidad del modelo, ya que si los criterios no son adecuados los resultados de aplicar el modelo no serán los más acertados. Además, es importante que una vez que un nuevo proyecto finalice, éste sea integrado al modelo de estimación y este a su vez recalibrado para que el modelo de estimación siempre modele la productividad actual de la empresa.

Una vez que se ha estimado un proyecto de software, se debe llevar a cabo un monitoreo a lo largo de todo el ciclo de vida del proyecto con el fin de llevar el control de éste. Esencialmente se debe dar seguimiento a las tres de las principales restricciones en los proyectos que son el alcance (tamaño), el costo (dinero) y el tiempo (calendario), ya que éstas determinarán al final si el proyecto ha sido exitoso.

Este documento se basa principalmente en la medición de tamaño funcional y en la estimación de proyectos de software. Sin embargo, no son las únicas métricas que acompañan a la Ingeniería de Software, es por ello que en el capítulo 2 se da una introducción a las Métricas de Software, abarcando los fundamentos, la definición de los conceptos básicos de éstas y los principios para diseñar y aplicar un método de medición formal.

Al impartir los temas que se desarrollan en este documento en los últimos semestres de la asignatura Métricas de Software, se ha observado que los estudiantes muestran un gran interés en esta asignatura identificando un área de oportunidad tanto en la investigación como en el ámbito laboral, ya que las Métricas de Software carecen de métodos de medición formales y la industria carece de buenas prácticas de estimación de proyectos de software.

Bibliografía

[1] SWEBOK - Software Engineering Body of Knowledge, P. Bourque y R. E. Fairley,, IEEE, versión 3.0, 2013, p. 335.

[2] S. Sánchez, M. Á. Sicilia y D. Rodríguez, Ingeniería del Software. Un enfoque desde la guía SWEBOK, Primera ed., Alfaomega Grupo Editor, S.A. de C.V., México, 2012, p. 568.

[3] IEEE Std 610-1990, IEEE Standard Glossary of Software Engineering Terminology.

[4] Coordinación de los Servicios de Cómputo de la Facultad de Ciencias, «Facultad de Ciencias. UNAM, México.,» 2013. [En línea]. Available: http://www.fciencias.unam.mx/licenciatura/resumen/104. [Último acceso: 2017].

[5] R. S. Pressman, Ingeniería del software: un enfoque práctico, Quinta ed., D. Ince, Ed., Madrid: McGraw-Hill, 2003, p. 601.

[6] W. D. Halsey, Macmillan Dictionary, London: Macmillan, 1973.

[7] ISO/IEC/IEEE 24765:2010, Systems and software engineering -- Vocabulary.

[8] IEEE Std 1002-1987, IEEE Standard Taxonomy for Software Engineering Standards.

[9] IEEE Std 610-1990, IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries.

[10] IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications.

[11] COSMIC Measurement Practice Commitee, The COSMIC Functional Size Measurement Method, Version 4.0.1, Measurement Manual, Julio 2015, p. 105.

[12] ISO/IEC 42010:2007, Systems and software engineering -- Recommended practice for architectural description of software-intensive systems.

[13] IEEE Std 1219-1998, IEEE Standard for Software Maintenance.

[14] T. M. Pigoski, Practical software maintenance: best practices for managing your software investment, illustrated ed., the University of Michigan: Wiley Computer Pub., 1997, p. 384.

[15] A. Abran, Software Metrics and Software Metrology, Illustrated ed., IEEE, Ed., Hoboken, New Jersey: John Wiley & Sons, INC., Publication, 2010, p. 328.

[16] Centro Español de Metrología, Vocabulario Internacional de Metrología, Conceptos fundamentales y generales, y términos asociados (VIM), 3a ed., 2012, p. 88.

[17] ISO/IEC 15939:2007, Systems and software engineering -- Measurement process.

[18] The International Function Point Users Group, Function Point Counting Practices Manual, Release 4.1.1, Abril 2000, p. 370.

[19] ISO/IEC 14143-6:2012, Information technology -- Software measurement -- Functional size measurement -- Part 6: Guide for use of ISO/IEC 14143 series and related International Standards.

[20] ISO/IEC 14143-1:2007, Information technology -- Software measurement -- Functional size measurement -- Part 1: Definition of concepts.

[21] David Consulting Group, «el Laboratorio de las TI,» 18 Febrero 2013. [En línea]. Available: http://www.laboratorioti.com/2013/02/18/informe-2012-metodos-utilizados-en-la-industria-para-medir-el-software/. [Último acceso: 2017].

[22] I. Jacobson, I. Spence y K. Bittner, USE-CASE 2.0, The Guide to Succeeding with Use Cases, Ivar Jacobson International, December 2011, p. 55.

[23] G. Karner, Resource Estimation for Objetory Projects, Objetive Systems SF AB, 1993.

[24] M. Chemuturi, Software Estimation Best Practices, Tools & Techniques: A Complete Guide for Software Project Estimators, Illustrated ed., J. Ross Publishing, 2009, p. 320.

[25] T. J. McCabe, A Complexity Measure, vol. Se2 No 4, IEEE Transactions on Software Engineering, December 1976.

[26] CMMI Institute, CMMI-SVC, vol. 1.3, Noviembre 2013, p. 556.

[27] IEEE Std. 1061-1998, IEEE Standard for a Software Quality Metrics Methodology.

[28] ISO/IEC 25000:2014, ystems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) -- Guide to SQuaRE.

[29] «Portal ISO 25000,» ISO 25000 calidad del producto de software, [En línea]. Available: http://iso25000.com/index.php/normas-iso-25000. [Último acceso: 2017].

[30] S. P. Mesa, Construcción de una herramienta para evaluar la calidad de un producto, Medellín: Departamento de Ingeniería de Sistemas Universidad EAFIT, 2007, p. 281.

[31] International Function Point Users Group, Allan J. Albrecht Father of Function Points, vol. 5, January 2011, p. 28.

[32] UKSMA Metrics Practices Committee, Mk II Function Point Analysis, Counting Practices Manual Version 1.3.1, September 1998, p. 100.

[33] Copyright © 2017 NESMA, [En línea]. Available: http://nesma.org/nesma/history. [Último acceso: 2017].

[34] ISO/IEC 24570:2005, Software engineering -- NESMA functional size measurement method version 2.1 -- Definitions and counting guidelines for the application of Function Point Analysis.

[35] NESMA, FPA according to NESMA and IFPUG, July 2015.

[36] NESMA, Early Function Point Analysis, 2015.

[37] FiSMA FSM Working Group, FiSMA Functional Size Measurement Method version 1.1, June 2004.

[38] FiSMA FSM Working Group, FiSMA 1.1, A Functional Size Measurement Method with continuous scale: Basic principles and practical examples, 2008, p. 54.

[39] ISO/IEC 29881:2008, Information technology -- Systems and software engineering -- FiSMA 1.1 functional size measurement method.

[40] COSMIC Measurement Practice Commitee, COSMIC Guideline for Early or Rapid Functional Size Measurement using approximation approaches, 2015, p. 46.

[41] M. D. Jean y A. Alain, Approximation Techniques for Measuring Function Points, Proc, in 13th Inter. Workshop on Software Measurement (IWSM 2003), 2003.

[42] S. L, Early and Quick COSMIC FFP Overview, in COSMIC Function Points: Theory and Advanced Practices, A. A. Reiner Dumke, Ed. Boca Raton, FL, USA: CRC Press, 2011.

[43] D. Carole y A. Alain, Proposed Measurement Etalon: C-Registration System, Software Engineering Research Laboratory, École de Technologie Supérieure (Canada), January 4, 2007.

[44] F. V. Souto y A. Abran, Case Study: COSMIC Approximate Sizing Approach without Using Historical Data, IWSM MENSURA, International Workshop on Software Measurement, Joint Conference of the 22nd: http://www.iwsm-mensura.org/2012, 2012.

[45] F. V. Souto y A. Abran, COSMIC Approximate Sizing Using a Fuzzy Logic Approach: An Experiment with Industry Data, IWSM MENSURA, International Workshop on Software Measurement: http://www.iwsm-mensura.org/2014, 2014.

[46] F. V. Souto y A. Abran, Improving the COSMIC Approximate Sizing Using the Fuzzy Logic EPCU Model, IWSM MENSURA, International Workshop on Software Measurement: http://www.iwsm-mensura.org/2015-cracow, 2015.

[47] A. Abran, Software Project Estimation: The Fundamentals for Providing High Quality Information to Decision Makers, Illustrated ed., Hoboken, New Jersey: John Wiley & Sons, Inc., 2015, p. 261.

[48] J. O. Rawlings, S. G. Pantula y D. A. Dickely, Applied Regression Analysis: A Research Tool, Second ed., Department of Statistics, North Carolina State University: Springer Science & Business Media, 2001, p. 660.

[49] G. A. F. Seber y A. J. Lee, Linear Regression Analysis, 2 ed., vol. 936 de Wiley Series in Probability and Statistics, John Wiley & Sons, 2012, p. 582.

[50] Project Management Institute, Inc., A Guide to the Project Management Body of Knowledge (PMBOK), 5 ed., Newtown Square, Pennsylvania USA: PMI Publishing Division, 2000 ed., p. 596.

[51] S. McConnell, Software Estimation: Demystifying the Black Art, Illustrated ed., the University of California: Microsoft Press, 19 Nov 2009, p. 308.

[52] J. Tuya, I. R. Román y J. J. D. Cosín, Técnicas cuantitativas para la gestión en la ingeniería del software, Netbiblo, 2007, p. 373.

[53] A. Abran, Software Benchmarking, Estimation and Quality Models Based on Functional Size with COSMIC – ISO 19761, Draft April 2008. MGL-841: La mesure: Concept clef en ingénierie du logiciel: Programme de Doctorat en genie.

[54] J. Lynch, CHAOS Report 2015, http://blog.standishgroup.com/post/50: The Standish Group, 2015.

[55] M. S. Magne Jørgensen, A Systematic Review of Software Development Cost Estimation Studies, IEEE Transactions on software engineering, 2007.

[56] Price Systems LLC, A Cracked Foundation: A Cracked Foundation: A Critical Look at the Role of Baselining in Government IT Project Management, Mount Laurel, New Jersey: GAO Testimony Before US Senate, September 7, 2006.

[57] J. J. Faraway, Extending the Linear Model with R: Generalized Linear, Mixed Effects and Nonparametric Regression Models, Boca Raton: CRC Press, 2006, p. 345.

[58] F. V. Souto, Earned Scope Management: A Case of Study of Scope Performance using COSMIC (ISO 19761) with a Real Project, IWSM-Mensura, International Workshop on Software Measurement: http://www.iwsm-mensura.org/2016, 2016.

[59] PMI, Practice Standard for Earned Value Management, Newtown Square, Pennsylvania: Project Management Institute, Inc., 2005, p. 54.

[60] P. Solanki, Earned Value Management: Integrated View of Cost and Schedule Performance, Global India Publications, 2009, p. 312.

[61] College of Performance Management, The Quarterly Magazine of the College of Performance Management, The Measurable News, Marzo 2013, p. 52.

[62] ES, «The official site for Earned Schedule information,» Walt Lipke, 2006. [En línea]. Available: http://www.earnedschedule.com/.

[63] W. Lipke, Earned Schedule, ES, Ed., PMI - Tulsa, Junio 7, 2006, p. 48.

Anexo A – Glosario de acrónimos

AC	Actual Cost	Costo Actual
ACWP	Actual Cost of Work Performed	Costo Real del Trabajo Realizado
AFP	Adjusted Function Points	Puntos de Función Ajustados
AFP	Average Functional Process	Proceso funcional promedio
AT	Actual Time	Tiempo Actual
AUC	Average Use Case	Caso de Uso Promedio
BAC	Budget At Completion	Presupuesto hasta la Conclusión
CFP	COSMIC Function Point	Punto de Función COSMIC
CMMI	Capability Maturity Model Integration	Integración de Modelos de Madurez de Capacidades
COSMIC	Common Software Measurement International Consortium	Consorcio Internacional para la Medición Común de Software
CPI	Cost Performance Index	Índice de Desempeño del Costo
CV	Cost Variance	Variación del Costo
DDC	Duration Difference at Completion	Diferencia de Duración a la Conclusión
DESP	Estimate Duration Variation	Variación de la Duración Estimada
DET	Data Element Type	Tipo de Elemento de Datos
DI	Degree of Influence	Grado de Influencia
DWH	Data Warehouse	Almacén de Datos
E	Entry	Entrada
EAC	Estimate AT Completion	Estimación a la Conclusión
EASY	Early & Speedy	Temprano y Veloz
EI	External Inputs	Entradas Externas

EIF	External Interface Files	Archivos de Interfaz Externa
EO	External Output	Salidas Externas
EPCU	Estimation of Projects in Contexts of Uncertainty	Estimación de Proyectos en Contextos de Incertidumbre
EQ	External Inquiries	Consultas Externas
ERP	Enterprise Resource Planning	Planificación de Recursos Empresariales
ES	Earned Schedule	Calendario Ganado o Devengado
ES	Earned Scope	Alcance Ganado
ESB	Equal Size Bands	Bandas de Igual Tamaño
ESM	Earned Scope Management	Gestión del Alcance Ganado o Devengado
ETC	Estimate To Complete	Estimación hasta la Conclusión
EV	Earned Value	Valor Ganado
EVM	Earned Value Management	Gestión del Valor Ganado o Devengado
FiSMA	Finnish Software Measurement Association	Asociación Finlandesa de Medición de Software
FP	Function Point	Punto de Función
FPA	Function Points Analysis	Análisis de Puntos de Función
FSC	Fixed Size Classification	Clasificación de Tamaño Fijo
FSM	Functional Size Measurement	Medición del Tamaño Funcional
FSMM	Functional Size Measurement Method	Método de Medición del Tamaño Funcional
GSCs	General System Characteristics	Características Generales del Sistema
IEAC	Independent Estimate at Completion	Estimado Independiente hasta la Conclusión

IEE	Institute of Electrical and Electronics Engineers	Instituto de Ingeniería Eléctrica y Electrónica
IFPUG	International Function Point User Group	Grupo Internacional de Usuarios de Puntos de Función
ILF	Internal Logic Files	Archivos Lógicos Internos
LOC	Lines of Code	Líneas de Código
MIS	Management Information System	Sistema de Información Gerencial
MMRE	Mean Magnitude of Relative Error	Media de la Magnitud del Error Relativo
MoProSoft	-	Modelo de Procesos para la Industria del Software
MRE	Magnitude of Relative Error	Magnitud del Error Relativo
NESMA	Netherlands Software Metrics Users Association	Asociación Holandesa de Usuarios de Métricas de Software
OIML	International Organization of Legal Metrology	Organización Internacional de Metrología Legal
PD	Planned Duration	Duración Planeada
PHR	Project Human Resources	Recursos Humanos usados en el Proyecto
PMBOK	Guide to the Project Management Body of Knowledge	Guía de los Fundamentos para la Dirección de Proyectos
PMI	Project Management Institute	-
PR	Productivity by Resource	Productividad por Recurso
PRED	Prediction Level	Nivel de Predicción
PROAVG	Average Productivity by Human Resource	Productividad Promedio por Recurso Humano
PRTC	Productivity required by resources defined to complete the scope as was planned	Productividad Requerida por Recursos definida para Completar el Proyecto como se Planeó
PS	Planned Scope	Alcance Planeado

PTC	Productivity to Complete	Productividad hasta la Conclusión
PV	Planned Value	Valor Planeado
PV	Productivity Variation	Variación de la Productividad
R	Read	Lectura
RET	Record Element Type	Tipo de Elemento de Registro
RMS	Root of the Mean Square	Raíz de la Media de los Cuadrados
RNTC	Human Resources Need to Complete the Project	Recursos Humanos Necesarios para Completar el Proyecto
RVTC	Resources Variation to Complete Planed Scope by period	Variación de Recursos para completar el Alcance planeado con la misma Productividad
SDMRE	Standard Deviation of Magnitude of Relative Error	Desviación Estándar de la Magnitud del Error Relativo
SMART	Software Measurement Approximation Rapid Technique	Técnica Rápida de Aproximación de Medición de Software
SPI	Schedule Performance Index	Índice de Desempeño del Calendario
SPI	Schedule Performance Index	Índice de Desempeño del Calendario
SPI	Scope Performance Index	Índice de Desempeño del Alcance
SQuaRE	Systems and software Quality Requirements and Evaluation	Requerimientos y Evaluación de Calidad de Productos de Software
SS	-	Alcance del proyecto
SV	Schedule Variance	Variación del Calendario
SV	Schedule Variance	Variación del Calendario
SV	Scope Variance	Variación del Alcance
SWEBOK	Software Engineering Body of Knowledge	Cuerpo de Conocimiento de la Ingeniería de Software
TDE	Total Duration Estimate	Duración Total Estimada
TDI	Total Degree of Influence	Grado Total de Influencia

TSPI	To Complete Schedule Performance Index	Índice de Desempeño del Calendario hasta la Conclusión
UCP	Use Case Points	Puntos de Casos de Uso
UFP	Unadjusted Function Points	Puntos de Función no ajustados
UKSMA	United Kingdom Software Metrics Association	Asociación de Métricas del Reino Unido
VAC	Variance At Completion	Variación hasta la Conclusión
VAF	Value Adjustment Factor	Valor de Factor de Ajuste
VIM	International Vocabulary of Metrology	Vocabulario Internacional de Metrología
W	Write	Escritura
X	Exit	Salida

Anexo B – Glosario de términos

Actividad. Conjunto cohesivo de tareas de un proceso

Alcance de la medición. Conjunto de los FUR que deben incluirse en un determinado ejercicio de medición de tamaño funcional.

Almacén persistente. Almacén que permite a un proceso funcional almacenar un grupo de datos más allá de la vida del proceso funcional y/o del cual un proceso funcional puede recuperar un grupo de datos almacenado por otro proceso funcional, o almacenado por una ocurrencia anterior del mismo proceso funcional, o almacenado por otro proceso.

Arquitectura de un sistema. Organización fundamental de dicho sistema plasmada en sus componentes, las relaciones entre éstos y con el entorno, y los principios que guían su diseño e implementación.

Atributo. Característica de una entidad.

Atributo de datos. Pieza más pequeña de información, dentro de un grupo de datos identificados, que tiene un significado desde la perspectiva de los requerimientos funcionales del software.

Calidad de software. Grado con el que un sistema, componente o proceso cumple con los requerimientos especificados y las necesidades o expectativas del cliente o usuario.

Capa. Partición funcional de una arquitectura de un sistema de software.

Caso de uso. Todas las formas de utilizar un sistema para lograr un objetivo particular para un usuario en particular.

Ciclo de vida del software. Evolución del proyecto de software desde el momento de su creación hasta el momento en que deja de usarse, y puede describirse en función de las actividades que se realizan dentro de él.

Clasificación. Clasificar cada requerimiento y asignar un tamaño al mismo (es decir, se aplica un factor de escala) que representa el tamaño funcional.

Codificar. Proceso de expresar un programa de computadora en un lenguaje de programación.

Comando de control. Comando que le permite al usuario funcional humano controlar el uso del software pero que no involucra ningún movimiento de datos sobre el objeto de interés del software que se está midiendo.

Componente. Una de las partes para formar un sistema. Un componente puede ser hardware o software y puede subdividirse en otros componentes.

Construcción del software. Actividad de combinar los elementos de configuración del software, usando la configuración de datos apropiada, en un programa ejecutable para su distribución a los clientes u otros receptores.

Diseño. Proceso para definir la arquitectura, los componentes, las interfaces y otras características de un sistema o un componente, y como el resultado de este proceso.

Entidad. Objeto que va a ser caracterizado mediante la medición de sus atributos.

Entrada. Movimiento de datos que mueve un grupo de datos desde un usuario funcional a través de la frontera hacia el proceso funcional donde se necesita.

Entrada desencadenante. Movimiento de entrada de datos de un proceso funcional que mueve un grupo de datos generados por un usuario funcional que el proceso funcional necesita para iniciar el procesamiento.

Escala de medición. Conjunto ordenado de valores de utilizado para clasificar magnitudes en orden creciente o decreciente según sus valores cuantitativos.

Escalamiento. Contar o medir cada requerimiento y multiplicar el conteo o medición por un número, el "factor de escala" para determinar su tamaño funcional. Un factor de escala está determinado por un proceso de calibración.

Escritura. Movimiento de datos que mueve un grupo de datos a un almacén persistente que se encuentra dentro de un proceso funcional.

Especificación. Descripción de manera completa, precisa y verificable los requerimientos, el diseño, el comportamiento u otras características de un sistema o componente y, a menudo, los procedimientos para determinar si se han cumplido estas descripciones.

Estimación. Predicción más optimista con una probabilidad distinta de cero de ser cierta. Una estimación es una predicción que tiene la misma probabilidad de estar por encima o por debajo del valor real.

Evento desencadenante. Evento que genera que un usuario funcional del componente de software desencadene uno o más procesos funcionales.

Frontera. Interfaz conceptual entre el software que se está midiendo y sus usuarios funcionales.

Grupo de datos. Conjunto único, no vacío, no ordenado y no redundante de atributos de datos donde cada atributo de datos incluido describe un aspecto complementario del mismo objeto de interés.

Ingeniería. Aplicación de un enfoque sistemático, disciplinado y cuantificable de estructuras, máquinas, productos, sistemas o procesos.

Ingeniería de Software. Aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento de software; es decir, la aplicación de la ingeniería al software.

Interfaz. Componente de hardware o software que conecta dos o más componentes con el fin de transmitir información de uno a otro.

Lectura. Movimiento de datos que mueve un grupo de datos desde un almacén persistente en el proceso funcional que los requiere.

Manipulación de datos. Todo lo que les sucede a los datos, distinto de un movimiento de datos en o de un proceso funcional, o entre un proceso funcional y el almacén persistente.

Mantenimiento del software. Modificación de un producto de software después de la entrega para corregir fallos, para mejorar su rendimiento u otros atributos, o para adaptar el producto a un entorno modificado.

Medición. Proceso que consiste en obtener experimentalmente uno o varios valores de atributos de entidades del mundo real.

Medida. Variable a la que se le asigna un valor como resultado de una medición.

Mensaje de error/confirmación. Salida emitida por un proceso funcional a un usuario humano funcional que, o bien confirma solamente que los datos han sido aceptados, o solamente que hay un error en los datos introducidos.

Método. Describe las características de un proceso o procedimiento ordenado utilizado en la ingeniería de un producto o la elaboración de un servicio.

Método de medición. Descripción genérica de una secuencia lógica de operaciones utilizada en una medición.

Metodología. Conjunto de prácticas, procedimientos, técnicas y reglas utilizados en la especificación del proceso a seguir.

Métrica. Medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado.

Metrología. Ciencia de las mediciones y sus aplicaciones.

Movimiento de datos. Componente funcional base que mueve un único grupo de datos.

Nivel de descomposición. Cualquier nivel resultante de dividir una pieza de software en componentes, dividir dichos componentes en subcomponentes, y después dividir estos subcomponentes en sub-sub componentes, etc.

Nivel de granularidad. Cualquier nivel de expansión de la descripción de una pieza de software de tal manera que, a cada aumento del nivel de expansión, la descripción de la funcionalidad de la pieza de software se encuentra en un nivel de detalle mayor y uniforme.

Nivel de granularidad de un proceso funcional. Nivel de granularidad de la descripción de una pieza de software en el que los usuarios funcionales: Son seres humanos individuales o dispositivos de ingeniería o elementos de software (y no grupos de estos) y detectan ocurrencias únicas de eventos a los que la aplicación software debe responder (y no cualquier nivel en el cual se han definido grupos de eventos).

Objeto de interés. Cualquier cosa del mundo del usuario funcional que se identifica en los FUR, sobre la que se requiere que el software procese y/o almacene datos. Puede ser cualquier cosa física, así como cualquier objeto conceptual o parte de un objeto conceptual.

Principio de medición. Fenómeno que sirve como la base de una medición.

Procedimiento. Camino a seguir para llevar a cabo una tarea determinada.

Procedimiento de medición. Descripción detallada de una medición conforme a uno o más principios de medida y a un método de medida dado, basado en un modelo de medida y que incluye los cálculos necesarios para obtener un resultado de medida.

Proceso. Secuencia de pasos llevados a cabo para un propósito específico.

Proceso funcional. Conjunto de movimientos de datos, que representa una parte elemental de los FUR para el software que se está midiendo, que es único dentro de estos FUR y que se puede ser definido de manera independiente de cualquier otro proceso funcional en estos FUR.

Productividad. Relación entre la cantidad de trabajo obtenido y el esfuerzo invertido.

Propósito de la medición. Declaración que define por qué una medición es necesaria, y para qué se utilizará el resultado.

Prueba de software. Proceso de analizar un componente de software para detectar las diferencias entre las condiciones existentes y los requerimientos (es decir, los errores) y evaluar las características del componente de software.

Requerimiento de software. Capacidad que debe alcanzar o poseer un sistema o componente de un sistema para satisfacer un contrato, estándar, especificación u otro documento formal.

Requerimiento no funcional. Requerimiento de software que describe no lo que hará el software, sino cómo lo hará el software.

Requerimientos funcionales. Requerimientos que especifican una función que un sistema o componente del sistema debe ser capaz de realizar.

Salida. Movimiento de datos que mueve un grupo de datos desde un proceso funcional a través de la frontera hacia el usuario funcional que los requiere.

Software. Conjunto de los programas de cómputo, procedimientos, reglas, documentación y datos asociados que forman parte de las operaciones de un sistema de computación.

Tamaño funcional. Tamaño de software derivado de cuantificar los Requerimientos Funcionales de Usuario.

Unidad de medida. Determinada cantidad definida y adoptada por convenio con la que se comparan otras cantidades del mismo tipo para expresar su magnitud relativa a esa cantidad.

Unidad de medida COSMIC. 1 CFP. El tamaño de un movimiento de datos.

Usuario funcional. Tipo de usuario que es un emisor y/o destinatario de los datos de los FUR de una pieza de software.

Anexo C – Guía de uso de material didáctico

El material didáctico desarrollado con base en los capítulos 1 a 6 de este documento consiste en una serie de diapositivas que abarcan los temas abordados en la asignatura Métricas de Software.

Las diapositivas están divididas en 14 presentaciones. La siguiente tabla muestra la relación de cada presentación con los capítulos de este documento.

Capítulo	Presentación
1. Introducción a la Ingeniería de Software	1. Introducción a las Métricas de Software
2. Introducción a las Métricas de Software	
3.2.5. ISO/IEC 20926:2009 - IFPUG FSMM	2. IFPUG
3. Estándares para la Medición	3. Introducción a la Medición del Software con COSMIC
	4. Fase de Estrategia de Medición
	5. Fase de Representación (Procesos Funcionales, Objetos de Interés y Grupos de Datos)
	6. Fase de Representación (Movimientos de Datos)
	7. Fase de Medición
	8. Ejemplos de Medición COSMIC
4. Aproximación de Tamaño Funcional	9. Aproximación de Tamaño Funcional
	10. Ejemplos de Aproximación de Tamaño Funcional
5. Estimación de Proyectos de Software	11. Estimación de Software
	12. Modelos de Estimación de Software
6. Evaluación de Desempeño de Proyectos	13. Evaluación de Desempeño de Proyectos
	14. Ejemplos Evaluación de Desempeño de Proyectos

Tabla Anexo C.1 - Relación de capítulos con presentaciones.

La asignatura tiene una duración de 16 semanas, por lo que se propone que la primera semana se utilice para que el profesor dé la presentación del curso, es decir, cuáles son los temas que se abordarán a lo largo de éste, que conocimientos obtendrá el estudiante, la forma de evaluación, etc. Las siguientes 14 semanas el profesor expondrá los temas que presentan las diapositivas. Al ser una materia enfocada a la Ingeniería de Software, es recomendable que en las últimas semanas del curso se asigne un proyecto enfocado a la estimación de software derivado de la medición de tamaño funcional, por lo que en la última semana del curso se propone que se revise dicho proyecto y se entreguen calificaciones finales.

Métricas de Software

Introducción a las Métricas de Software

Jesús Iván Saavedra Martínez

Octubre 2017

Facultad de Ciencias

Agenda

I. Madurez de la Ingeniería de Software
II. Situación Actual del Desarrollo de Software
III. SWEBOK
IV. Modelos de Medición de Software

¿Qué son las métricas?

- El término se utiliza como referencia a diferentes conceptos:
  - La cantidad a ser medida
  - El procedimiento de medición
  - El resultado de la medición o para modelos entre múltiples mediciones

Madurez de la Ingeniería de Software

- Mantener el enfoque intuitivo o la forma ad-hoc de hacer las estimaciones de software NO contribuye a la madurez de la ingeniería de software.

- La ingeniería de software es una ingeniería que está apenas aprendiendo a medir, a estimar y a mejorar la calidad de sus productos y procesos.

"medir lo que sea medible y hacer medible lo que no lo es"

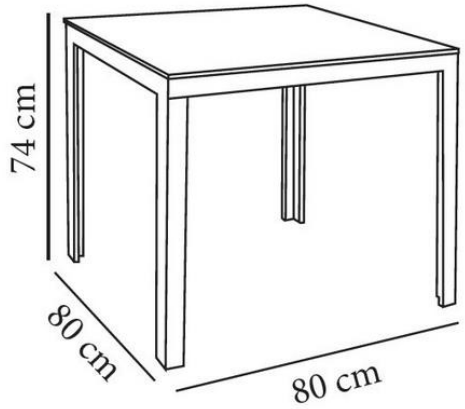
Galileo Galilei

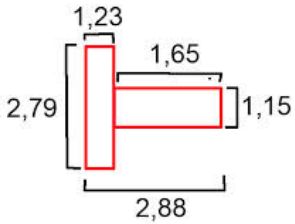
Madurez de la Ingeniería de Software

- IEEE (IEEE Standard Glossary of Software Engineering Terminology, IEEE std 610.12-1990, 1990)
  - "(1) La aplicación de un enfoque sistemático, disciplinado y **cuantificable** al desarrollo, operación y mantenimiento de software; es decir, la aplicación de la ingeniería al software.
  - (2) El estudio de enfoques como en (1)."

Madurez de la Ingeniería de Software

- Establecer el método correcto suele ser muy complicado y lleva tiempo años o incluso siglos.
  - Determinación exacta de la longitud en la navegación (inicio viajes transoceánicos finales S.XV, medición adecuada longitud finales S. XVIII)
  - La ingeniería de software es joven OTAN 1968 (46 años)
  - La mayoría de los atributos del software actualmente están descritos de una manera cualitativa en lugar de cuantitativa, y dependen demasiado del punto de vista de las personas.





Madurez de la Ingeniería de Software

- Podemos construir barcos, quizá pequeños comparados con los que se podrán realizar en un futuro.
- Todavía navegamos perdiendo el rumbo.
- Todavía existen muchas métricas incorrectas desde el punto de vista técnico.
- Muchas creencias erróneas sobre la bondad de ciertos métodos o herramientas.

Facultad de Ciencias

Disciplinas maduras VS inmaduras

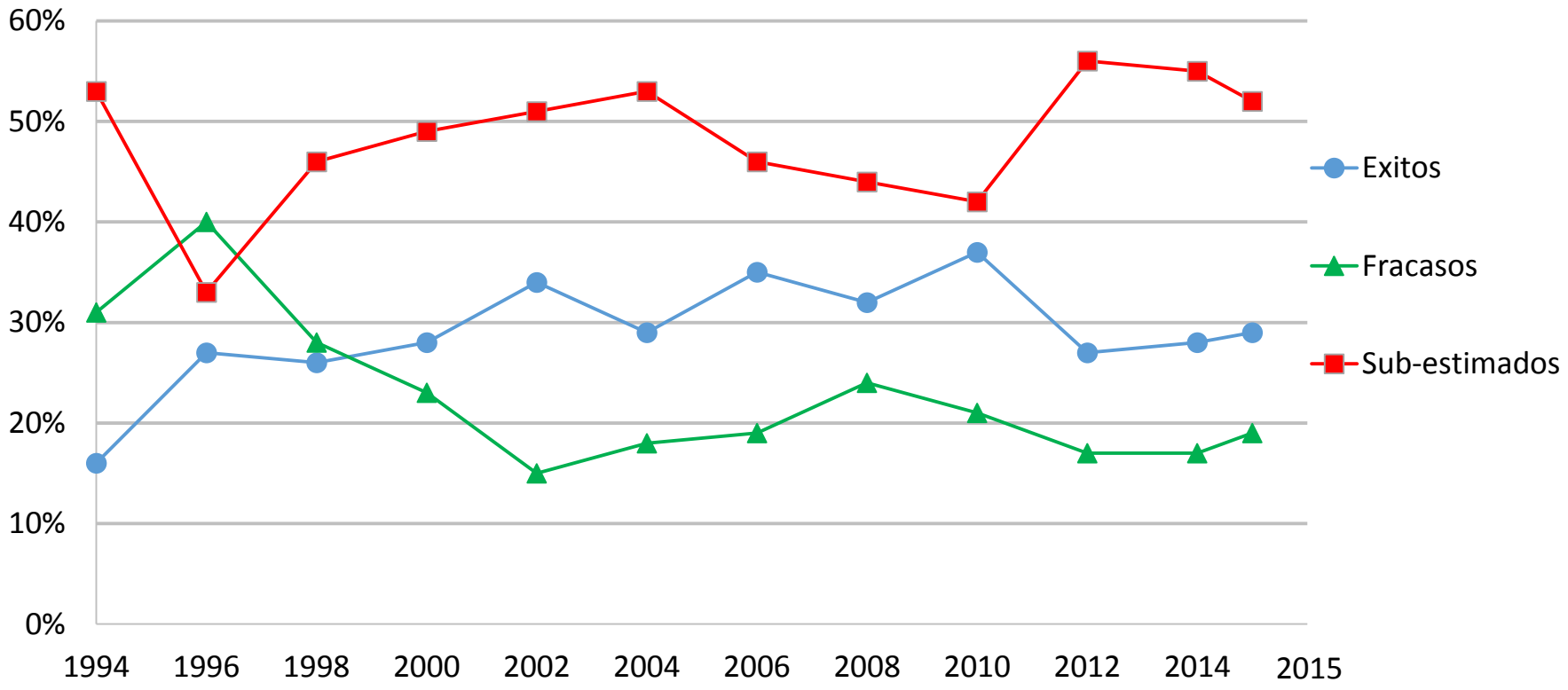
SOFTWARE	MADURAS
Número limitado de medidas de software aceptadas por los practicantes y reconocidas como maduras a través de un estándar	Un amplio consenso internacional sobre medidas.
Un pequeño número de estudios experimentales rigurosos.	Métodos de medición establecidos y referencias (etalón)
	Número significativo de instrumentos de medición para ser utilizados en distintos contextos y bajo ciertas, muchos de ellos calibrados y certificados contra referencias internacionales.

Facultad de Ciencias

Agenda

I. Madurez de la Ingeniería de Software
II. Situación Actual del Desarrollo de Software
III. SWEBOK
IV. Modelos de Medición de Software

Chaos Manifesto



Proceso de producción de Software (SW)

Productividad = **Producción** / Salida
Recursos / Insumos

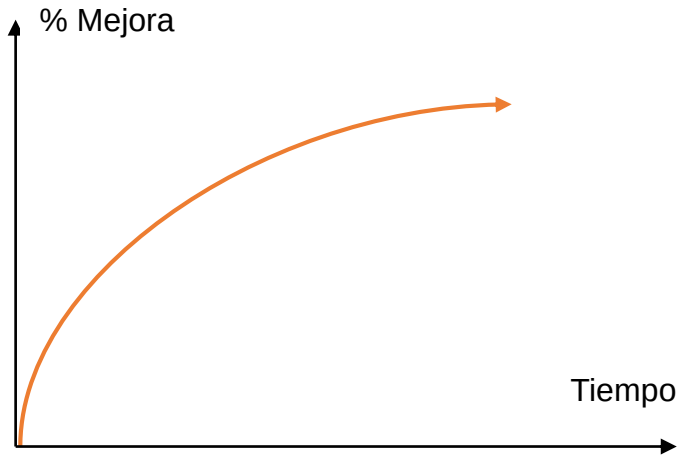
¿Cuánto se produce?

- ¿Qué podemos **automatizar en el proceso de producción** del software, cuando los que transforman los Insumos/factores (requerimientos) en productos (software) **son personas**?

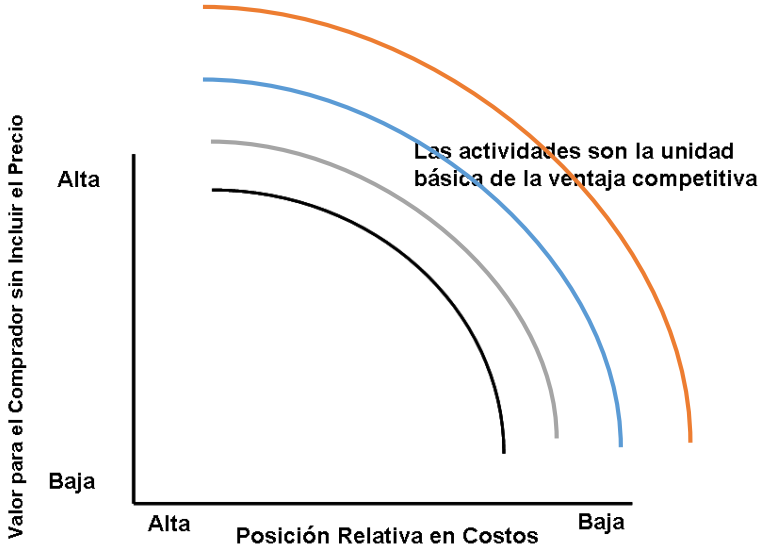
- **PRÁCTICAS** (Subprocesos, Procedimientos, Modelos, Actividades, etc.)

Efectividad Operacional

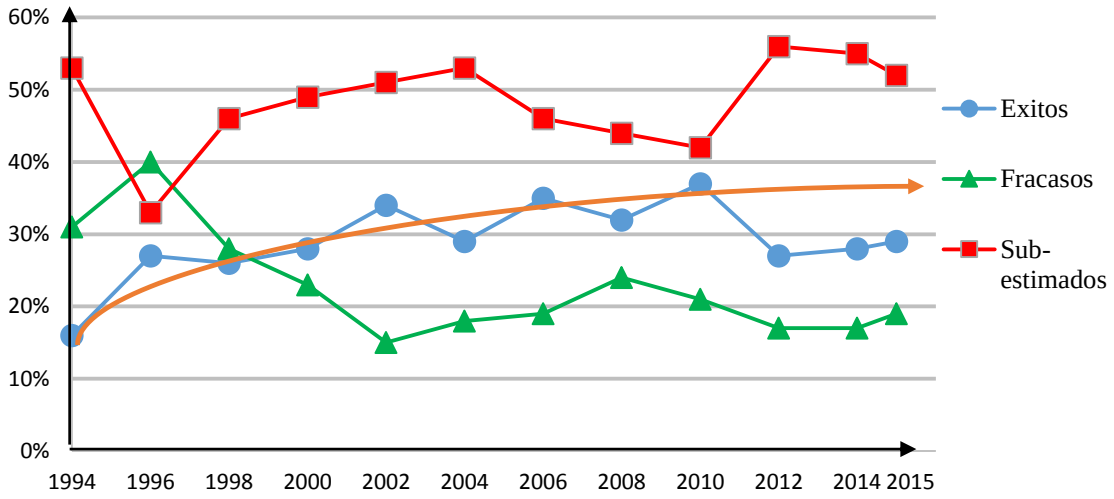
- **Efectividad Operacional**: conlleva cualquier número de **PRÁCTICAS** que le permiten a la empresa utilizar de mejor manera los insumos de producción



- **Barrera de Productividad**: Máximo valor que una compañía puede entregar sobre un producto o servicio a un costo dado.



Situación Actual del Desarrollo de Software



ISO, CMMI, MoPROSOFT,
PSP/TSP, PMP, Metodologías Ágiles
Etc…

Año	Proyectos Exitosos	Sub-estimados o cancelados
1994	16%	84%
1996	27%	73%
1998	26%	74%
2000	28%	72%
2002	34%	66%
2004	29%	71%
2006	35%	65%
2009	32%	68%
2010	37%	63%
2012	27%	73%
2014	28%	72%
2015	29%	71%

Facultad de Ciencias

Éxito en los proyectos de software

- "**17%** de los grandes proyectos de IT **van tan mal que pueden amenazar la existencia misma de la empresa**"
- "En promedio, los grandes proyectos de IT se ejecutan **sobre el 45% del presupuesto** y **sobre el 7% el tiempo**, mientras que se entrega un **valor menor al 56%** de lo predicho"

Source: McKinsey & Company in conjunction with the University of Oxford

Type of survey: Study on large scale IT Projects

Date: 2012

Facultad de Ciencias

Éxito en los proyectos de software

- "Un estudio muestra que un increíble **70%** de las organizaciones **han tenido al menos un proyecto fracasado en los últimos 12 meses**"
- "**50%** de los encuestados también indicaron que el proyecto **fracaso por no lograr lo que se propusieron lograr**"

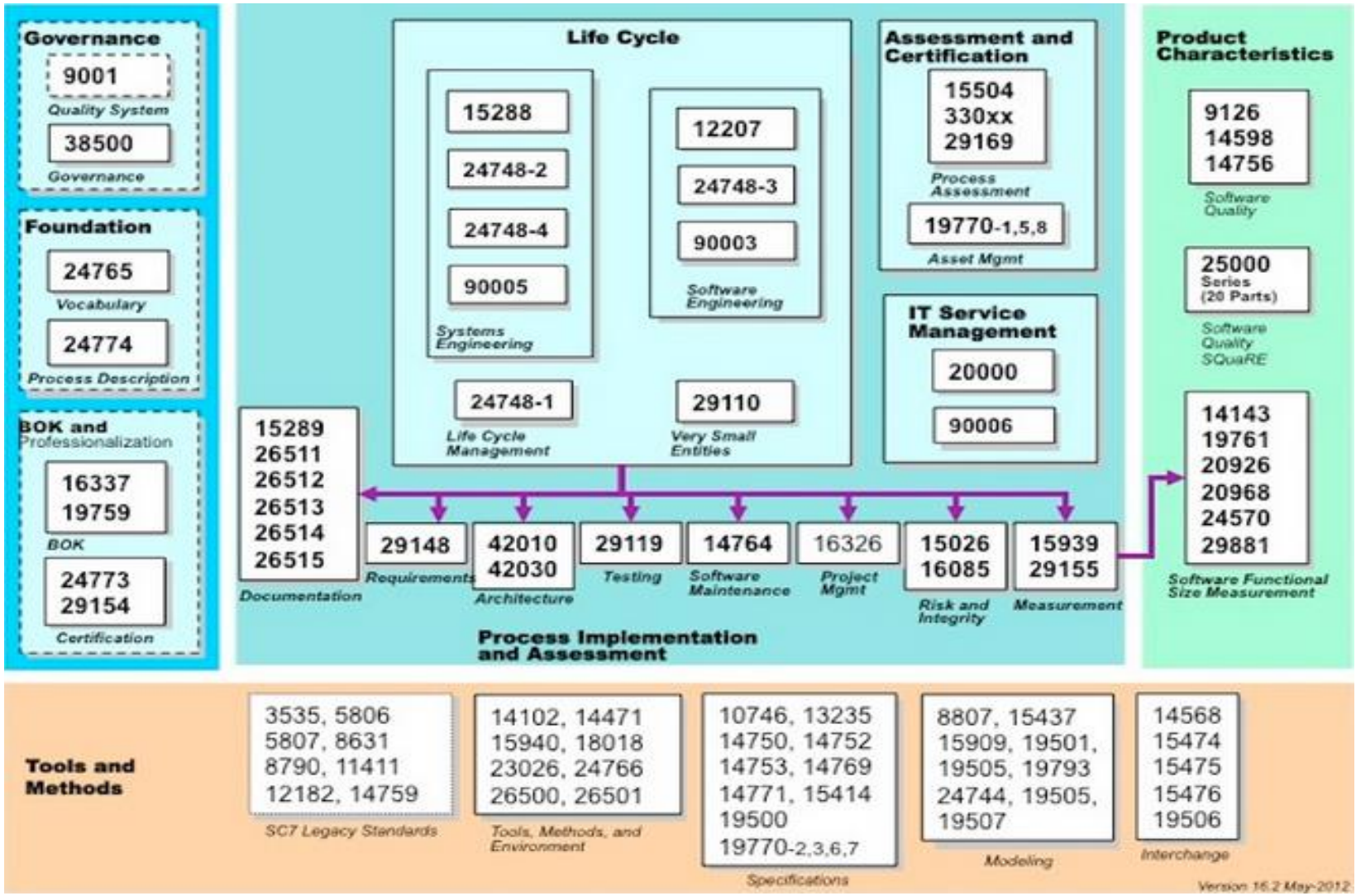
Source: KPMG (New Zealand)

Type of survey: Survey of 100 bussiness across a broad cross section of industries

Date: Dec 2010

Facultad de Ciencias

Estándares ISO relacionados con Software



Credibilidad

- Comunidad de investigación en medición de software v.s comunidades prácticas
  - Falta de credibilidad
  - Inmadurez de medidas propuestas

Facultad de Ciencias

Agenda

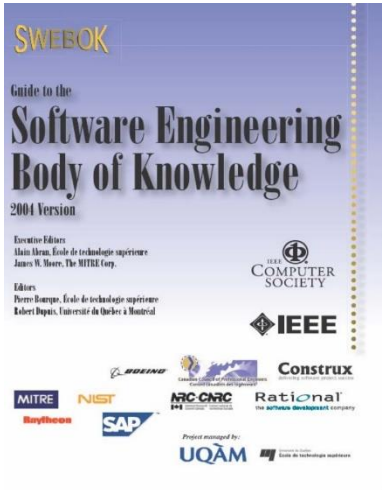
I. Madurez de la Ingeniería de Software
II. Situación Actual del Desarrollo de Software
III. SWEBOK
IV. Modelos de Medición de Software

SWEBOK

- La ingeniería de Software recientemente ha alcanzado el estado de disciplina legítima de ingeniería como una profesión reconocida.
  - Hasta 2003 no había un cuerpo de conocimiento generalmente aceptado en este campo
  - En 90´s la IEEE a través de varios grupos Task Force incluye el proyecto SWEBOK, que es un compendio del cuerpo del conocimiento que ha evolucionado en las últimas 4 o 5 décadas.
  - El SWEBOK es generalmente aceptado, esto es, aplica en la mayoría de los proyectos, la mayor parte del tiempo, y un amplio consenso valida su valor y efectividad. (2004)

Cuerpo de conocimiento (BOK)

- Cuerpo del conocimiento es un término usado para representar el conjunto suficiente de conocimientos en un área específica.



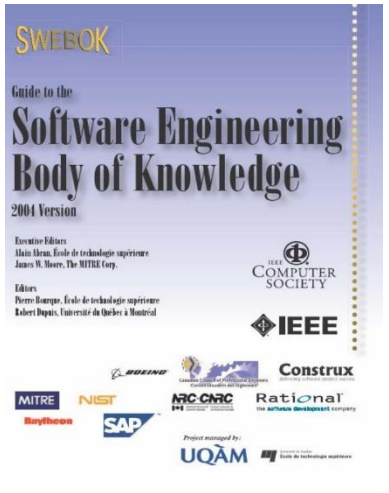
- Un cuerpo del conocimiento se compone generalmente de las áreas del conocimiento que representan una taxonomía de conceptos relevantes.

SWEBOK

SWEBOK Áreas de Conocimiento (KA)	ISO 12207 Tipos de Proceso
1. Requerimientos de Software	Procesos Fundamentales
2. Diseño de Software	
3. Construcción de Software	
4. Pruebas de Software	
5. Mantenimiento de Software	
6. Configuración de Software	Procesos de Soporte y Procesos de la Organización
7. Administración de la Ingeniería del Software	
8. Proceso de la Ingeniería de Software	
9. Herramientas y Métodos de Ingeniería de Software	
10. Calidad de Software	

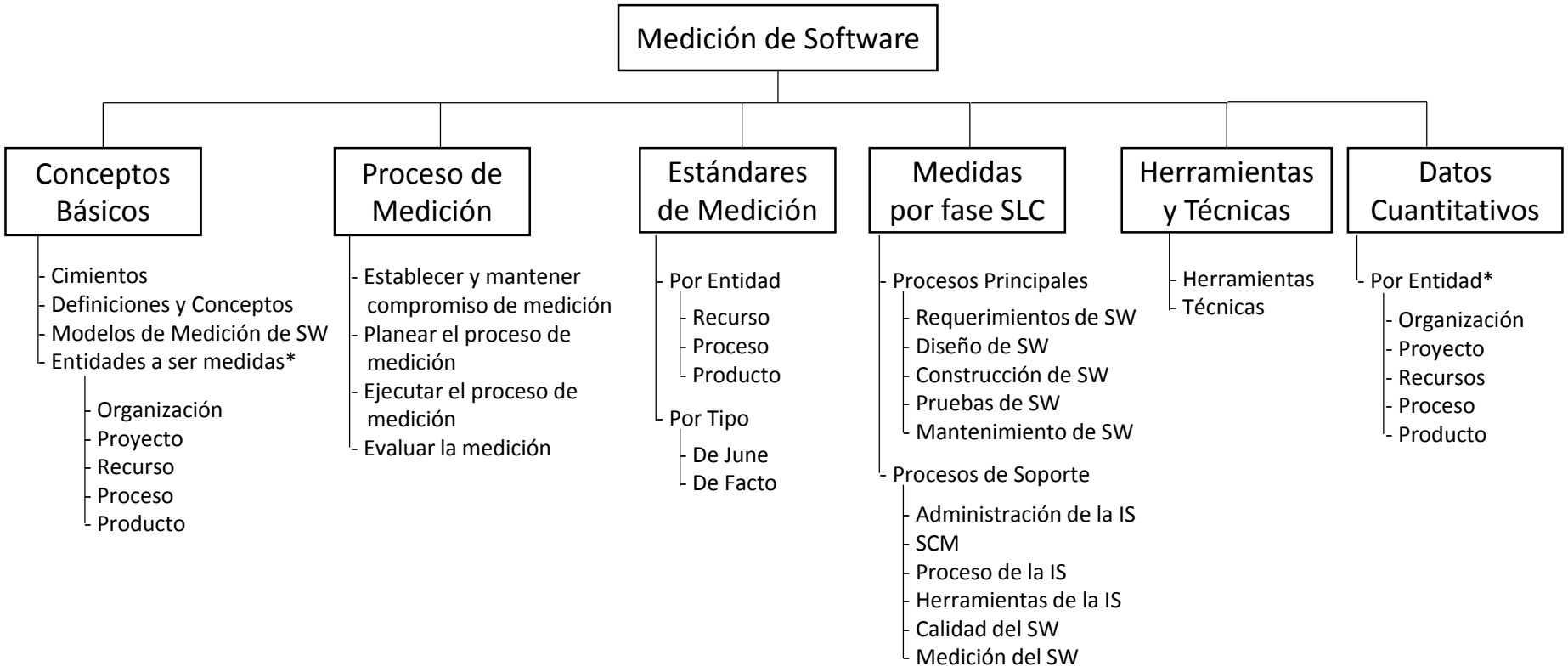
SWEBOK

- En el SWEBOK se identifican tres temas transversales que son:
  - Calidad,
  - Herramientas,
  - Medición.



- Los 2 primeros han sido reconocidos como áreas clave (KA)

SMKA

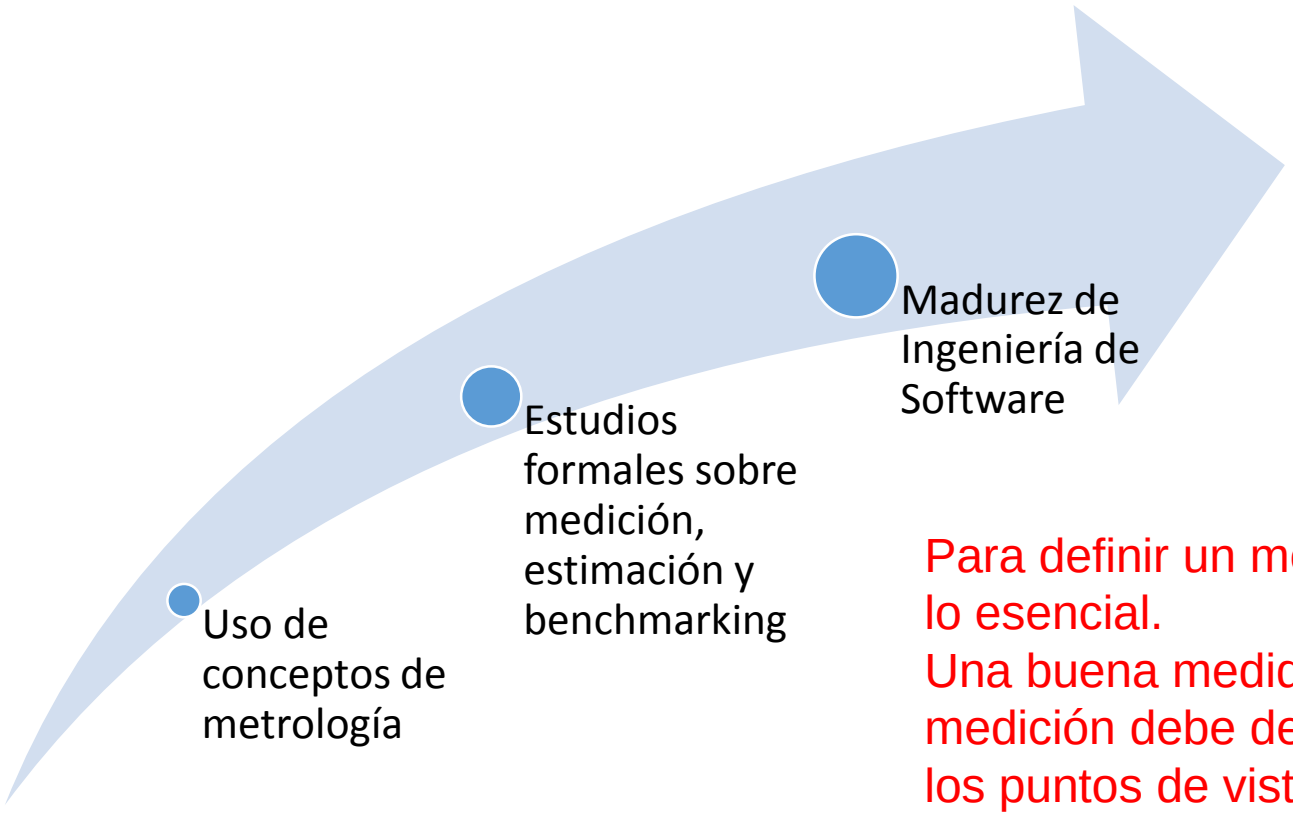

Medición de Software
Conceptos Básicos
Cimientos
Definiciones y Conceptos
Modelos de Medición de SW
Entidades a ser medidas*
Organización
Proyecto
Recurso
Proceso
Producto
Proceso de Medición
Establecer y mantener compromiso de medición
Planear el proceso de medición
Ejecutar el proceso de medición
Evaluar la medición
Estándares de Medición
Por Entidad
Recurso
Proceso
Producto
Por Tipo
De June
De Facto
Medidas por fase SLC
Procesos Principales
Requerimientos de SW
Diseño de SW
Construcción de SW
Pruebas de SW
Mantenimiento de SW
Procesos de Soporte
Administración de la IS
SCM
Proceso de la IS
Herramientas de la IS
Calidad del SW
Medición del SW
Herramientas y Técnicas
Herramientas
Técnicas
Datos Cuantitativos
Por Entidad*
Organización
Proyecto
Recursos
Proceso
Producto

Facultad de Ciencias

Agenda

I. Madurez de la Ingeniería de Software
II. Situación Actual del Desarrollo de Software
III. SWEBOK
IV. Modelos de Medición de Software

Madurez de la Ingeniería de Software



Para definir un método se debe de ver lo esencial.
Una buena medida de un método de medición debe de ser independiente de los puntos de vista.

Número en sentido matemático

- Se aprende desde la primaria
- Varios número y como se combinan
  - Comparación
  - Suma
  - División…
- No hay incertidumbre asociada en estos números

Número obtenido de ejercicio de medición

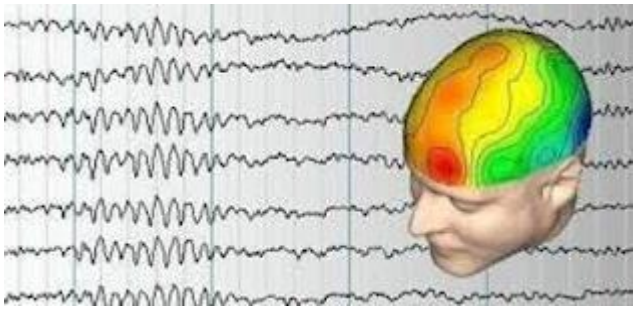
- Es más que un número
- Tiene asociado una unidad de medida y un procedimiento para obtener el número
- Este resultado de la medición está asociado a un atributo específico de una entidad específica.
- Tiene asociada incertidumbre por varios factores potenciales de error.

Modelos cuantitativos y modelos de decisiones

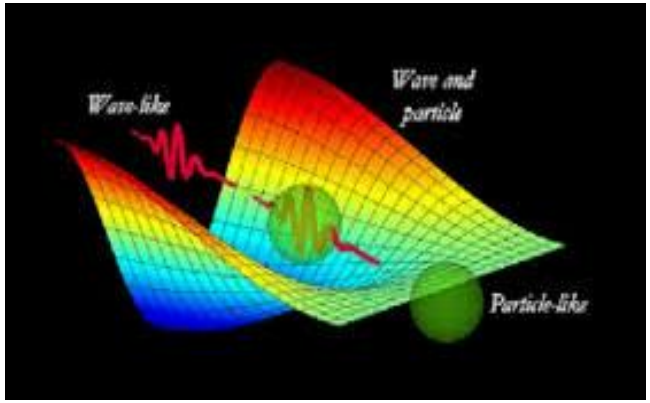
- Los modelos de decisión y los modelos de evaluación representan una combinación de figuras o reglas sobre cómo interpretar medidas de diferentes tipos.

Pensamiento General

- Desarrollo de software es un arte.
- Es un producto más intelectual que físico.
- Es influenciado por muchas variables. La mayoría cualitativas.
- Es más complejo.
- Etc.



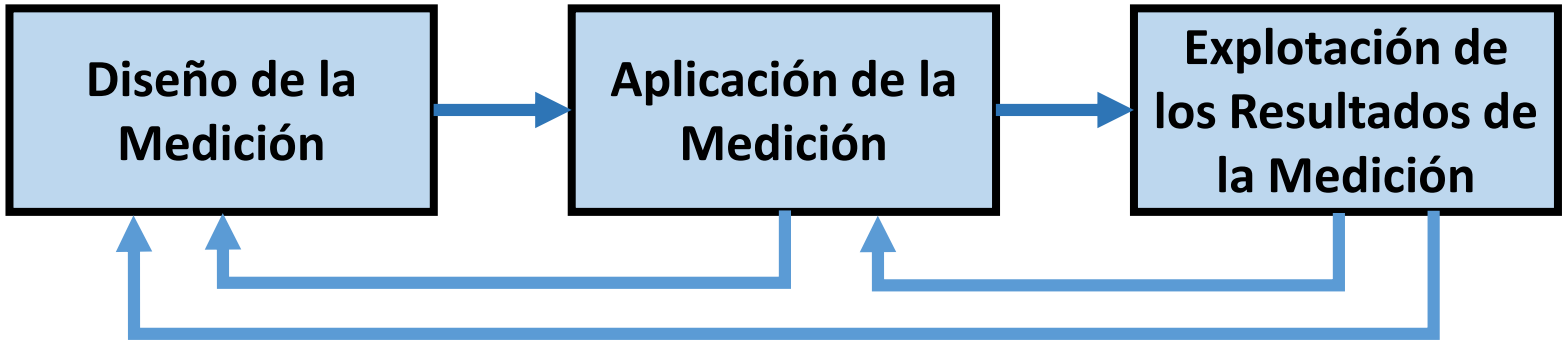
Características de un Modelo de Medición



- CONCEPTO: Bien definido, claro que represente la realidad
- PROCEDIMIENTO DE MEDICIÓN: Generalmente aceptado, bien fundamentado, reproducible.
- UNIDADES. Consistentes, claras, Etalón.

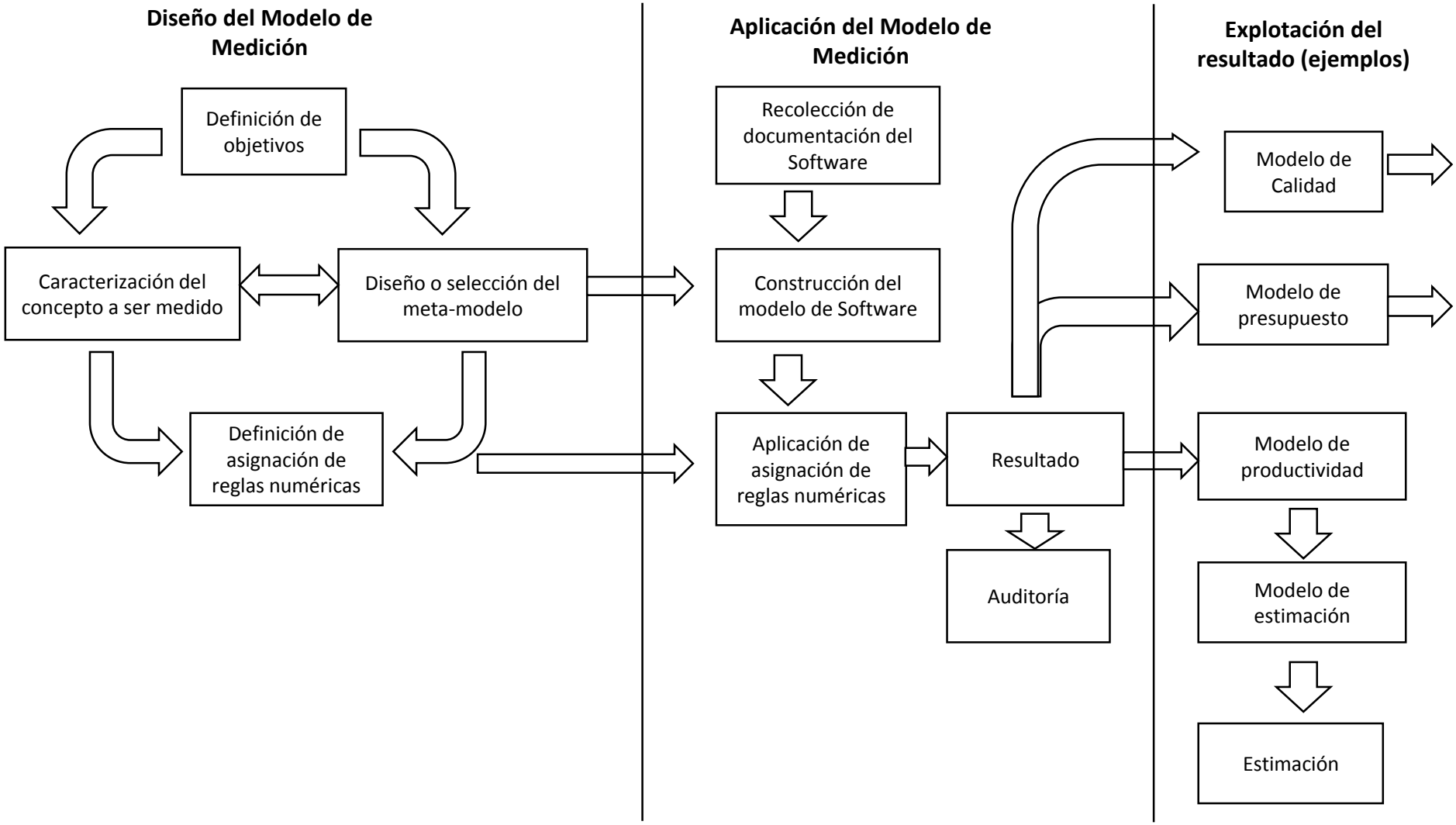
- CONCLUSIÓN: EL SOFTWARE SE PUEDE MEDIR, LO QUE FALTA ES TENER BUENOS MODELOS, QUE REPRESENTEN LA REALIDAD.

Modelo de contexto de medición



1. Diseño	2. Aplicación	3. Explotación
Verificación del diseño de un método de medición (con respecto al objetivo y el principio de medición)	Verificación del los procedimientos de medición, incluyendo los instrumentos o pasos de medición y los resultados obtenidos.	Verificación de la calidad de los modelos y la relación entre múltiples resultados de medición

Modelo de contexto de medición a detalle



Facultad de Ciencias

¿Cómo diseñar un método de medición?

1. Determinación de los objetivos de medición
2. Caracterización del concepto a ser medido especificando las entidades a ser medidas y las características (atributos) a considerar.
3. Desarrollo del meta modelo incluyendo las relaciones a través del concepto (entidad y atributo)
4. Definición de reglas de asignación numérica.
5. Aplicación Modelo desarrollado

Facultad de Ciencias

Contacto

- Jesús Iván Saavedra Martínez
- jism@ciencias.unam.mx

Métricas de Software

IFPUG

Jesús Iván Saavedra Martínez

Octubre 2017

Facultad de Ciencias

Agenda

I. Historia de IFPUG

II. Proceso de Medición

III. Caso de estudio: Compra-Venta de Autos

IV. Medición IFPUG

IFPUG

- IFPUG: ISO 20926

  (International Function Point Users Group)

Historia

- Fines de 70's - Allan Albrecht de IBM definió los conceptos que permitían medir el resultado de los proyectos de desarrollo de software

- Versión 2.0 (Abril 1988) – Con el crecimiento en el uso de los puntos de función, se crearon guías para interpretar las reglas originales en nuevos ambientes. Se crea en esta época el IFPUG International Function Point User Group

Historia

- Versión 3.0 (Abril 1990) – El IFPUG Counting Practices Committee se concentra en establecer el estándar de conteo de puntos de función

- Versión 4.0 (Enero 1994) - Refleja el uso de los puntos de función en fases tempranas del ciclo de vida para estimación, incluye ejemplos con GUI.

Historia

- Versión 4.1 (Enero 1999) – Clarifica reglas y conceptos importantes, cumple con estándar ISO/IEC 14143-1:2007 sin las 14 Características Generales del Sistema
- Versión 4.1.1 (Abril 2000) – Corrección de errores
- Versión 4.2 (Enero 2004) - Clarifica reglas ya existentes para obtener mayor consistencia, reestructura manual en 4 partes
- Versión 4.2.1 (Enero 2005) – Corrección de errores

Contexto referencia Estudio A. Albrecht

- La organización incluye a 450 personas que desarrollan aplicaciones.
- El desarrollo está bajo contrato con clientes de IBM.
- Los desarrolladores y los clientes están dispersos por todo Estados Unidos.
- En cualquier momento entre 150 y 170 contratos están abiertos.
- En promedio un contrato implica dos o tres personas.
- Cada contrato se realiza en el contexto de un marco de desarrollo específico.
- La mayoría de los contratos se limitan a ciertas fases de la metodología.

Facultad de Ciencias

Proceso de medición

- Basado en su experiencia, la fase de diseño representa el 20% de las horas de trabajo, la implementación del 80%.
- Es necesario medir todo el proceso, incluyendo el diseño y los costos incurridos durante el diseño.
- Los proyectos se completaron desde mediados de 1974 hasta principios de 1979.
- El tamaño de los proyectos varía de 500 a 105,000 horas de trabajo.
- De unos 1500 contratos para el período, sólo 22 cumplieron los criterios de selección

Facultad de Ciencias

Criterios de aceptación de proyectos en Estudio A. Albrecht

1. El proyecto debe haber pasado por todas las fases de la metodología (desde la definición de los requisitos hasta la implementación) y debe haber sido entregado al cliente.
2. Todo el proyecto debe haber sido gestionado adecuadamente con definiciones consistentes de las tareas y los procesos de gestión.
3. Todas las horas de trabajo de IBM y del cliente deben ser conocidas y deben haber sido cuidadosamente medidas
4. Los factores funcionales deben ser conocidos

Agenda

I. Historia de IFPUG

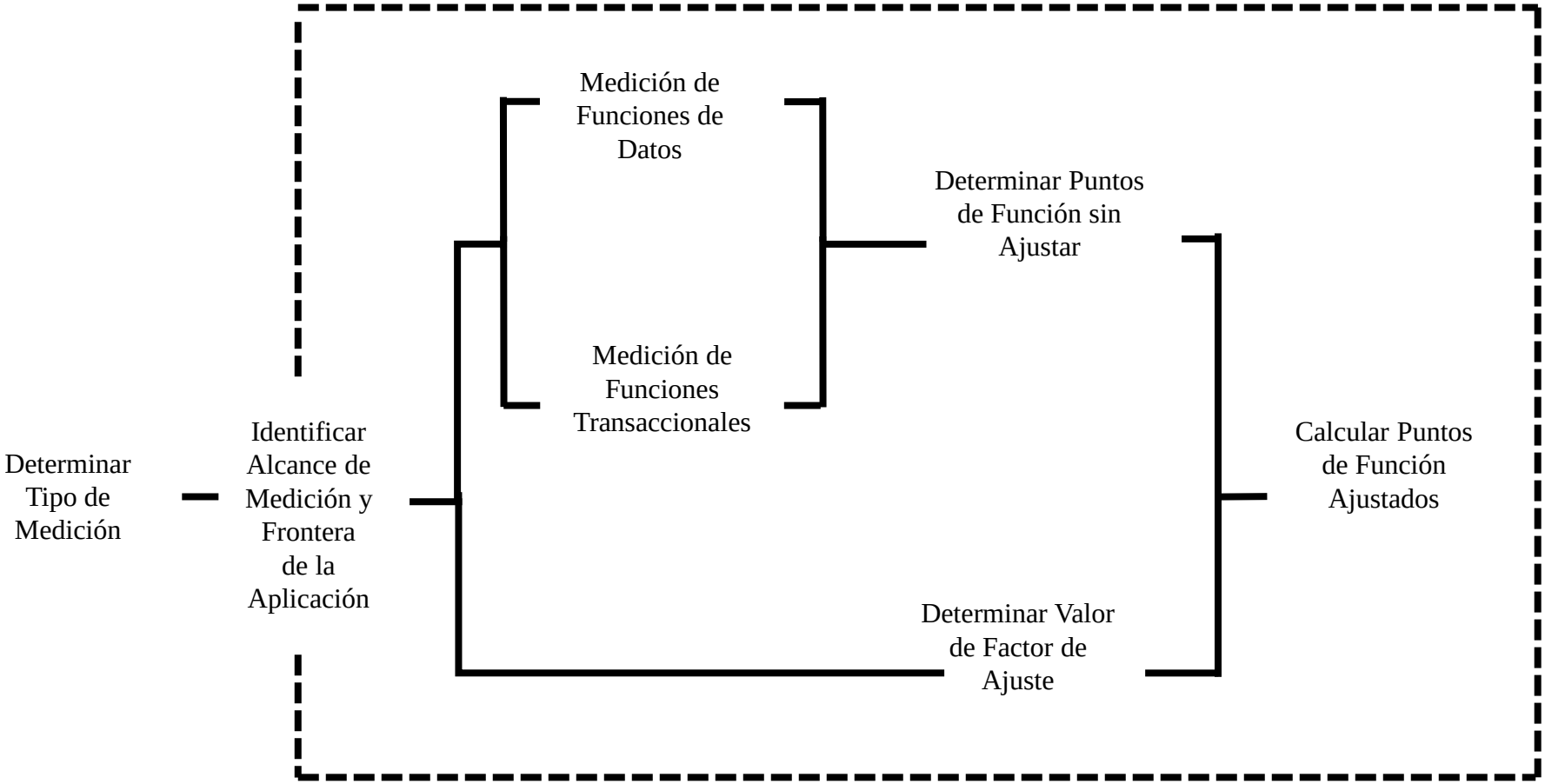
II. Proceso de Medición

III. Caso de estudio: Compra-Venta de Autos

IV. Medición IFPUG

Facultad de Ciencias

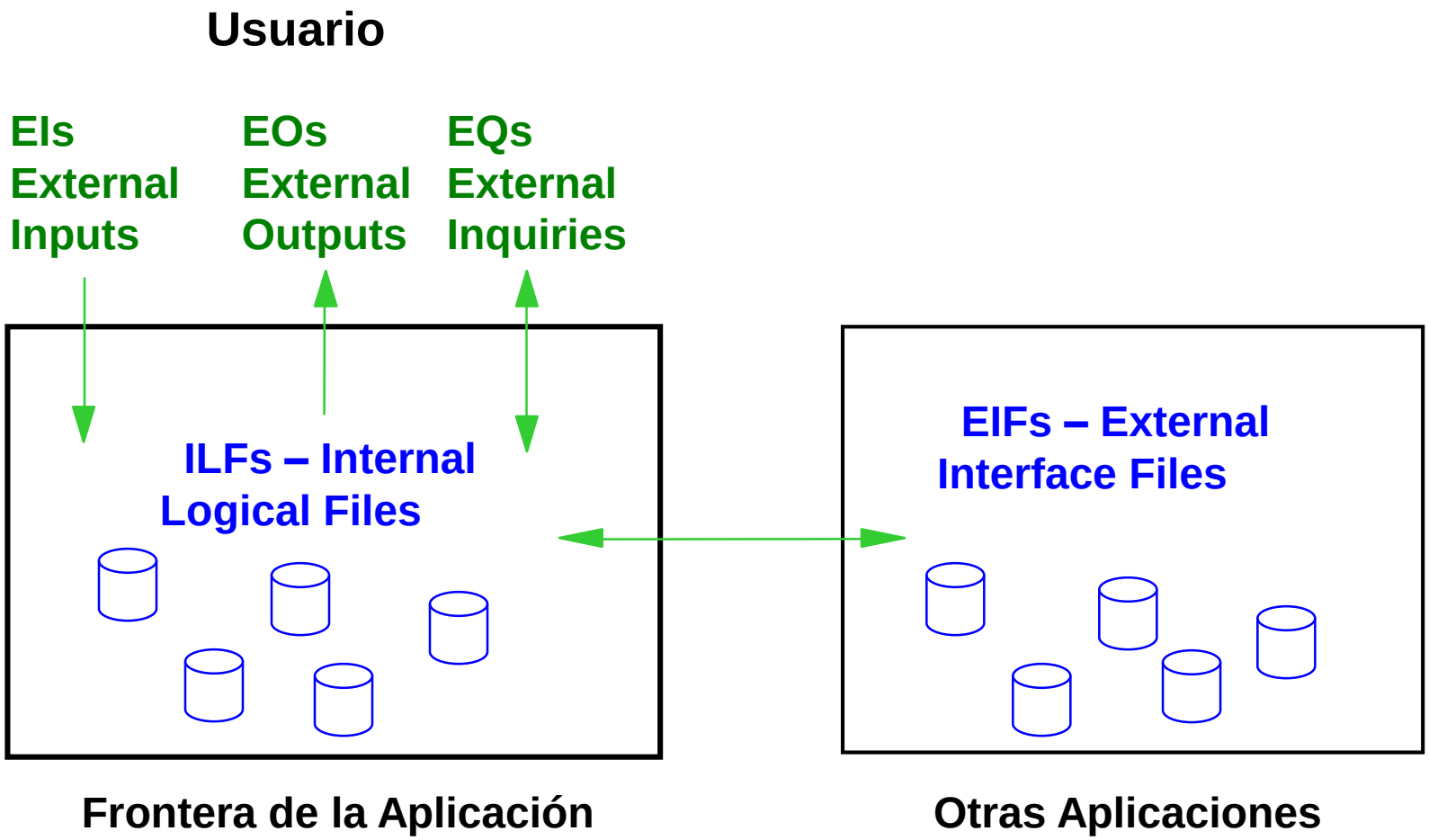
Proceso de medición



Punto de vista del usuario

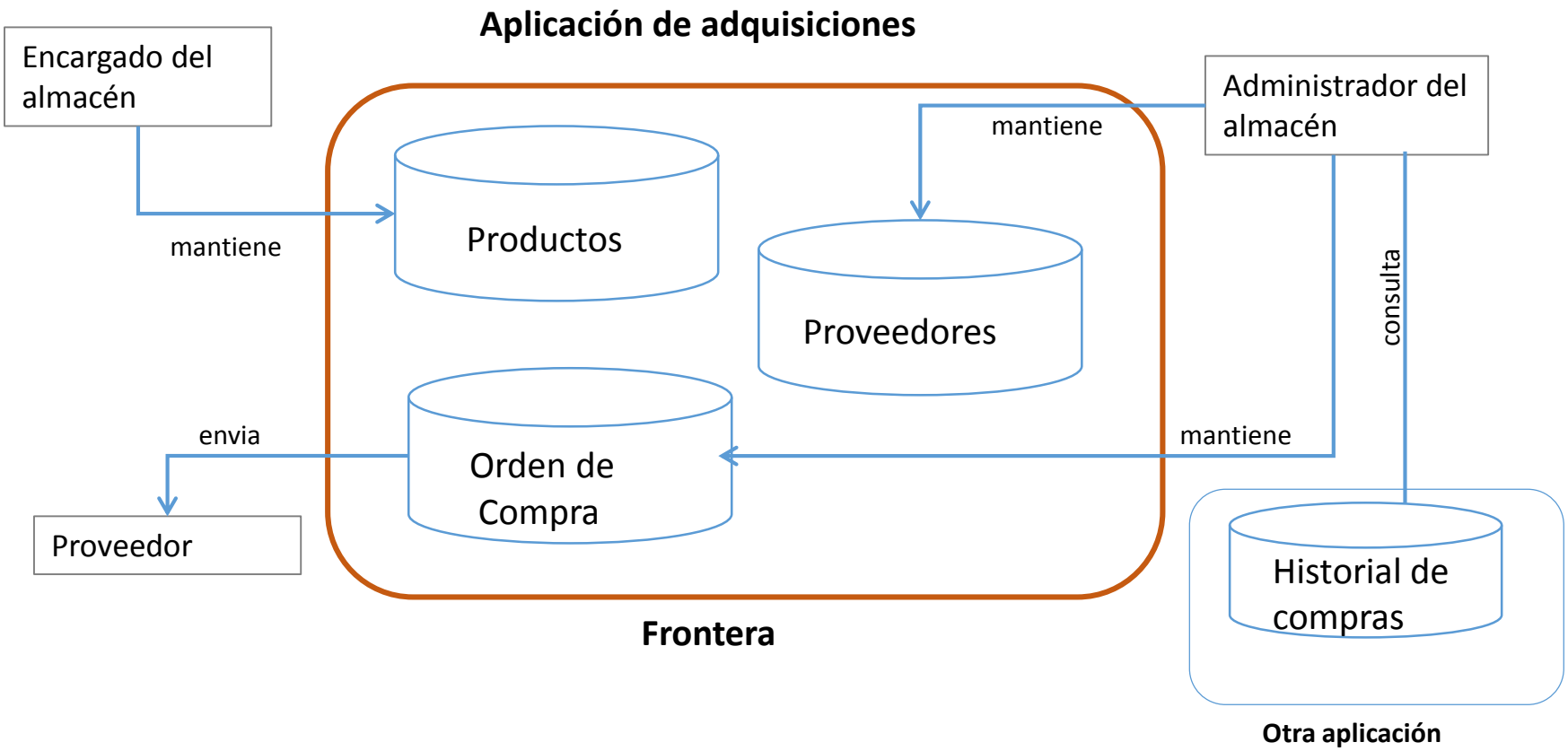
- El usuario es cualquier persona o cosa que comunica o interactúa con el software en cualquier momento.
- El punto de vista del usuario:
  - Es una descripción del las funciones del negocio de la aplicación
  - Son los requerimientos descritos por el usuario
  - Estos requerimientos son aprobados y reconocidos por el usuario
  - La medición de tamaño funcional es acompañada de la información que es entendida tanto para el(los) usuario(s) como para los desarrolladores
  - Ej. Documento de requerimientos, especificaciones detalladas de lo que el usuario necesita que la aplicación cumpla.

Concepto


Usuario
EIs
External
Inputs
EOs
External
Outputs
EQs
External
Inquiries
ILFs – Internal
Logical Files
EIFs – External
Interface Files
Frontera de la Aplicación
Otras Aplicaciones

Frontera de la Aplicación

Ejemplo:



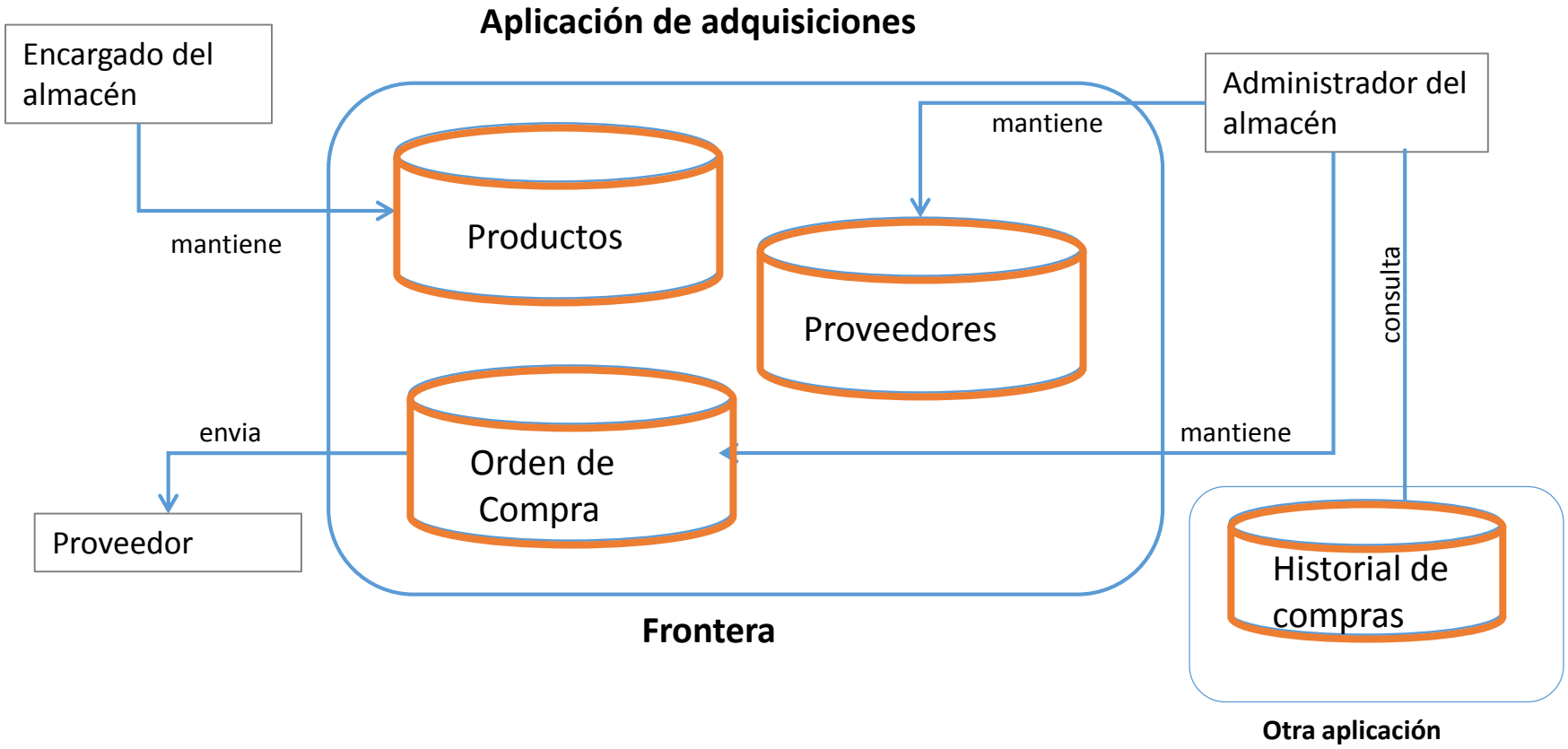
Medir la función de datos

- La **función de datos** representa la funcionalidad proporcionada al usuario para cumplir con los requerimientos de almacenamiento de datos internos y externos.
- La función de datos puede ser:
  - ILF Internal Logical File
  - EIF External Interface File

- Nota: El término file (archivo) no se refiere a un archivo físico o una tabla. En este caso se refiere a un grupo de datos relacionados de manera lógica y no a su implementación física.

Ejemplo de ILFs, EIFs

Ejemplo:



Complejidad de los ILFs/EIFs

- DET Data Element Type - Atributo único, no repetible, reconocido por el usuario.

- RET Record Element Type – Subgrupo de datos dentro de un ILF/EIF reconocido por el usuario

ILF – Internal Logical File
EIF – External Interface File

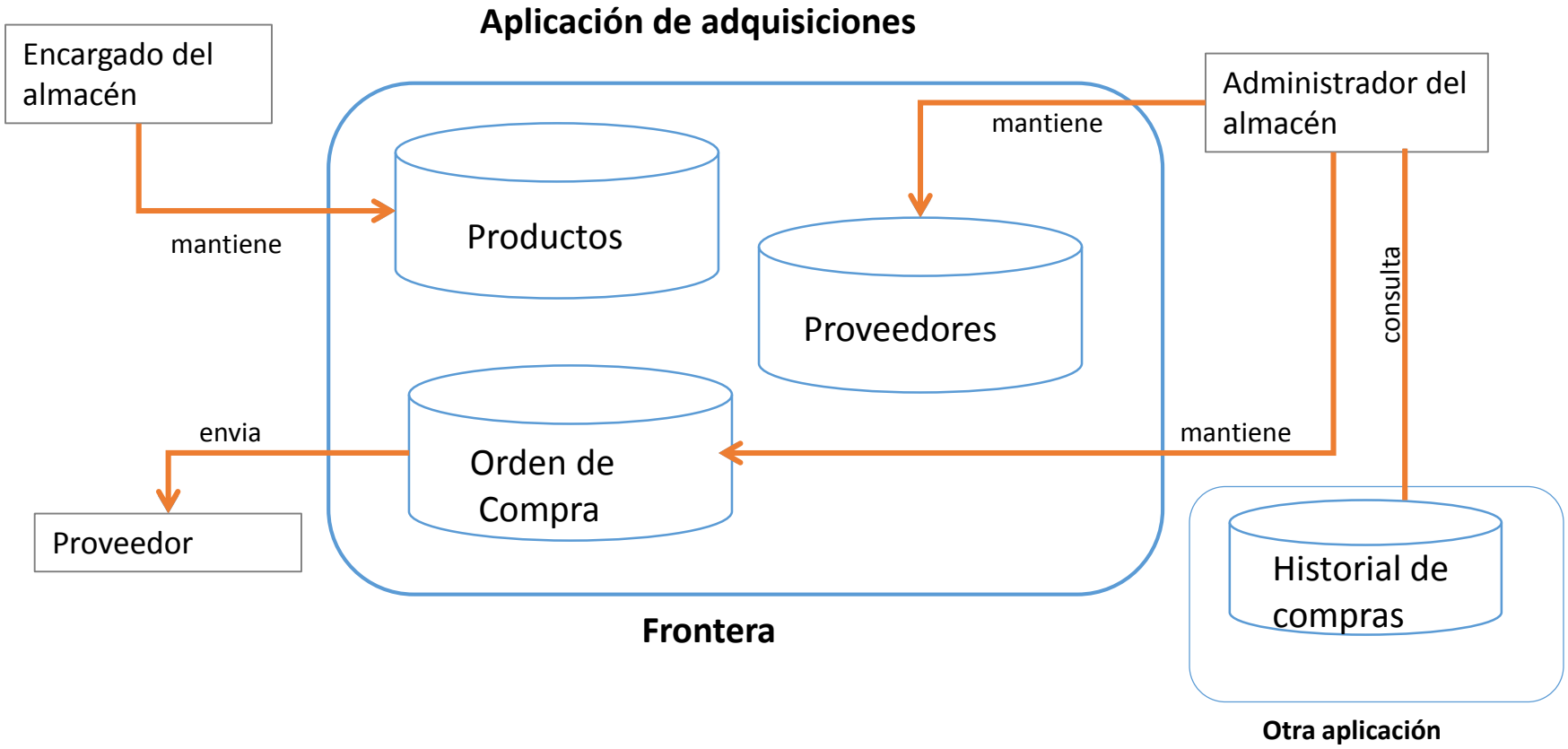
RETs	DETs		
	1 - 19	20 - 50	>50
1	P	P	M
2 - 5	P	M	G
>= 6	M	G	G

Transacciones

- Un EI External Input es un proceso elemental que recibe datos o información de control enviada desde fuera de la frontera de la aplicación
- Un EO External Output es un proceso elemental que envía datos o información de control fuera de la frontera de la aplicación e incluye procesos lógicos adicionales a los EQs.
- Un EI External Inquiry es un proceso elemental que envía datos o información de control fuera de la frontera de la aplicación.

Ejemplo de ILFs, EIFs

Ejemplo:



Complejidad de los EI/EO/EQ

EI – External Input

FTR	DETs		
	1 - 4	5 - 15	>15
0 – 1	P	P	M
2	P	M	G
>= 3	M	G	G

EO – External Output
EQ* – External Inquiry

FTR	DETs		
	1 - 4	5 - 15	>15
0 – 1	P	P	M
2 - 3	P	M	G
> 3	M	G	G

*EQ necesita al menos 1 FTR

Facultad de Ciencias

Matriz de Contribución

	Pequeño	Mediano	Grande	Totales
ILF	___x 7	___x 10	___x 15	_____
EIF	___x 5	___x 7	___x 10	_____
EI	___x 3	___x 4	___x 6	_____
EO	___x 4	___x 5	___x 7	_____
EQ	___x 3	___x 4	___x 6	_____
			UFP =	_____

Conteo Ajustado (VAF)

- Este factor de ajuste modifica el valor del conteo en un ±35% dependiendo de cómo influyen 14 características generales del sistema que califican a la aplicación como un todo en requerimientos no funcionales.
- Se evalúan cada una de estas 14 GSCs en una escala de 0 a 5 para determinar su influencia (DI)
- Se suman estos 14 grados de influencia: TDI = ∑ DI

VAF = (TDI * 0.01) + 0.65

Facultad de Ciencias

Escalas

Tipo de Escala		Transformación Admisible	Operaciones	Ejemplos
Nominal	(R,=)	f unique	Name distinguish	Colors, shapes
Ordinal	(R,>=)	f strictly increasing monotonic function	Rank, Order	Preferences, hardness
Interval	(R,>=,+)	f(x)= ax + b, a>0	Add	Calendar time, temperature, (degrees celsius)
Ratio	(R,>=,+)	f(x) = ax, a>0	Add, multiply, divide	Mass, distance, absolute temperatura, (degrees kelvin)
Absolute	(R,>=,+)	f (x) = x	Add, multiply, divide	Entity count

Facultad de Ciencias

Agenda

I. Historia de IFPUG
II. Proceso de Medición
III. Caso de estudio: Compra-Venta de Autos
IV. Medición IFPUG

Facultad de Ciencias

Requerimientos Compra-Venta de Autos

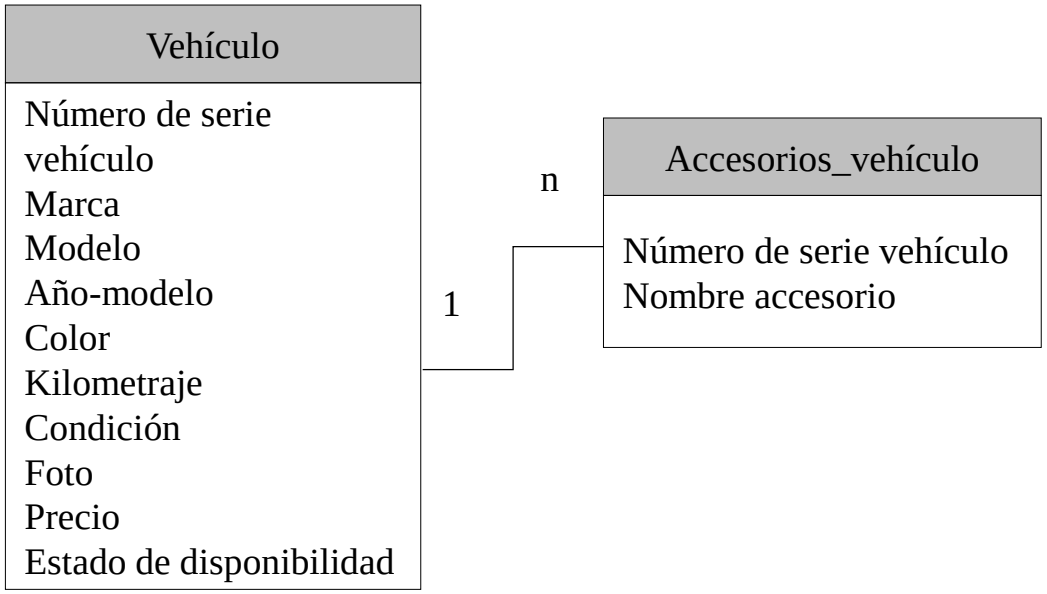
Una empresa de compra-venta de autos usados, ha decidido ofrecer en la red el servicio de venta de autos. Para ello ha pedido a una empresa desarrolladora de software, la ejecución del proyecto. De antemano, la empresa desarrolladora, prevé que los requerimientos y restricciones funcionales, serán de complejidad promedio y el desarrollo será en C#.

Como parte de los requerimientos funcionales, el cliente ha indicado lo siguiente:

1. Un posible comprador de vehículo podrá acceder a la página Web del distribuidor para conocer el precio y comprar un auto del inventario.

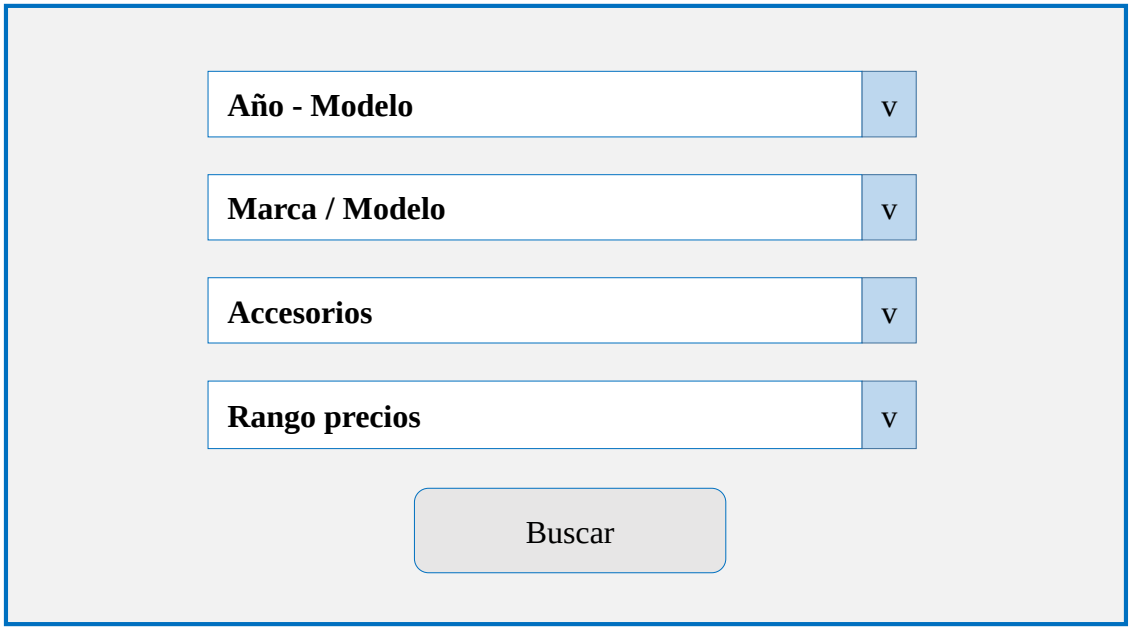
Requerimientos Compra-Venta de Autos

2. La aplicación Web utilizará la información que ya se mantiene dentro del Sistema de Inventario de Vehículos (SIV). Este sistema mantiene los vehículos y los posibles accesorios de cada uno de ellos. El modelo de datos que maneja es el siguiente:



Requerimientos Compra-Venta de Autos

3. Un comprador potencial, podrá buscar un vehículo de dos formas: podrá ver todo el inventario, o filtrar con base a una serie de características de los vehículos. La pantalla de consulta será de la forma siguiente:



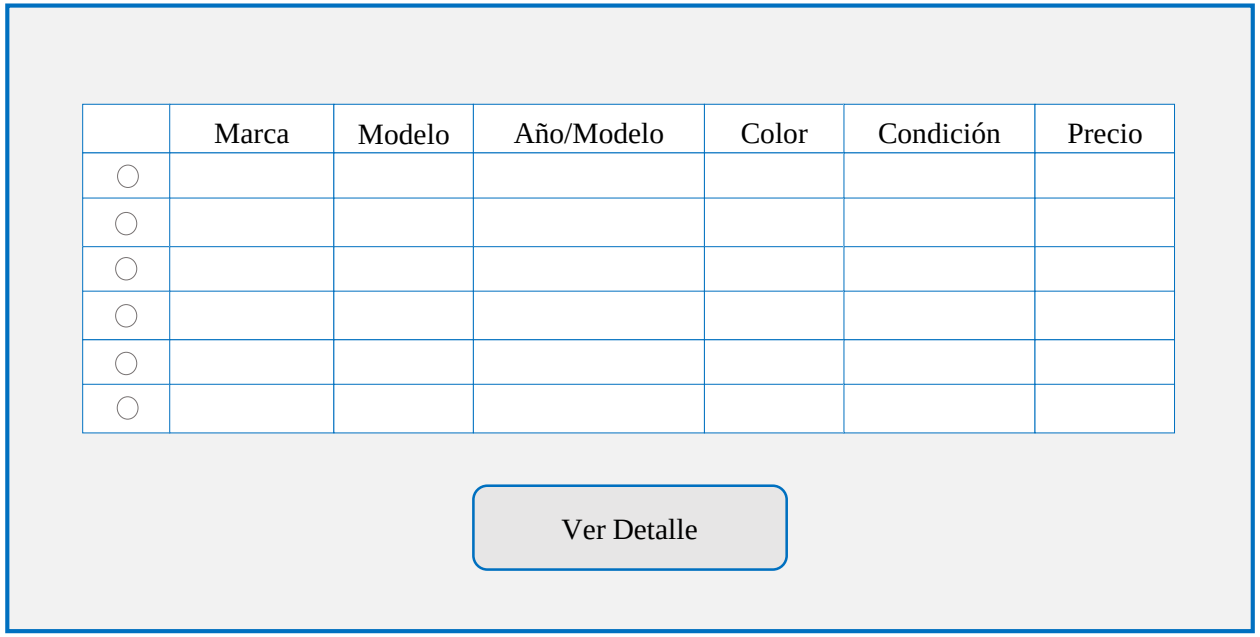
Requerimientos Compra-Venta de Autos

4. Los drop-downs del Año-Modelo, Marca/Modelo y Accesorios, serán llenados a partir de la información almacenada en el Sistema de Inventario de Vehículos. Adicionalmente, el drop-down de “Rango de Precios” ofrecerá 5 rangos de precio. Cada uno será calculado a partir del valor mínimo y máximo del precio de los autos que se tienen en inventario, y luego dichos valores serán concatenados para mostrarse de la manera siguiente:

- Entre 50,000 y 100,00
- Entre 100,001 y 150,000
- Entre 150,001 y 200,000
- …

Requerimientos Compra-Venta de Autos

5. Con base en los parámetros de consulta, será mostrado un listado con los vehículos disponibles en el inventario. Si no hay vehículos, el mensaje “No hay vehículos con esas características”, se desplegará.



Requerimientos Compra-Venta de Autos

6. El comprador podrá entonces verificar el listado, y si se ve un vehículo que le llame la atención, podrá seleccionarlo y hacer una consulta detallada, la cual se mostrará en la pantalla siguiente:

Comprar

Características del Vehículo	
Color:	
Kilometraje:	
Condiciones:	
Número de Serie:	
Precio de Venta:	
Accesorios:	

Requerimientos Compra-Venta de Autos

7. Si el posible comprador, decide comprar el vehículo, entonces presionará el botón “Comprar”, que lo llevará a la siguiente pantalla, en dónde registrará sus datos, los de su aseguradora y los de su institución financiera. Tal y como se muestra en la pantalla siguiente:

Captura Datos de Compra	
Nombre:	
Dirección:	
Teléfono:	
E-mail:	
Compañía de Seguros:	
Agente de Seguros:	
Teléfono Compañía de Seguros:	
Fax Compañía Seguros:	
Nombre Institución Financiera:	
Teléfono Institución Financiera	
Fax Institución Financiera:	

Comprar

Requerimientos Compra-Venta de Autos

8. Al término de esta operación, presionará el botón: “Comprar”, lo cual hará que el auto seleccionado quede como “comprado” en el inventario de vehículos. La información de compra se guardará en el almacenamiento de compras, y por último, se enviará un correo de confirmación al comprador, informando: el número de compra, nombre del comprador, nombre de la aseguradora, nombre de la financiera, marca del vehículo comprado, modelo, año-modelo, color, accesorios y precio del vehículo. Los datos quedarán almacenados en la siguiente tabla:

Compra
Número de Compra
Número de Serie Vehículo
Nombre Comprador
Teléfono Comprador
Dirección Comprador
E-mail Comprador
Compañía de Seguros
Agente de Compañía de Seguros
Teléfono Compañía Seguros
Fax Compañía Seguros
Nombre Institución Financiera
Teléfono Institución Financiera
Fax Institución Financiera

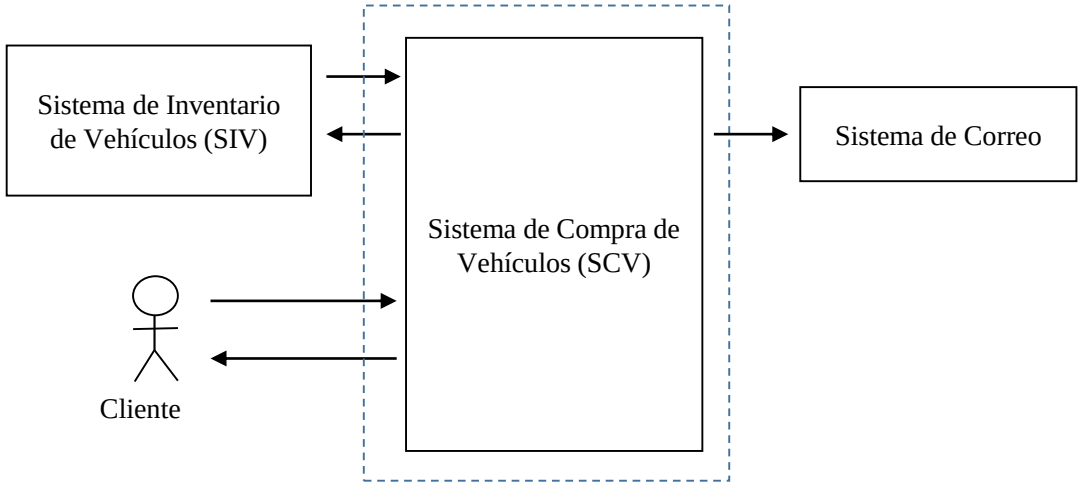
Facultad de Ciencias

Agenda

I. Historia de IFPUG
II. Proceso de Medición
III. Caso de estudio: Compra-Venta de Autos
IV. Medición IFPUG

Medición IFPUG

- Tipo de conteo: Nuevo Desarrollo
- Propósito: Determinar el conteo para posteriormente calcular el esfuerzo y costo del proyecto.
- Frontera del sistema:



Identificación de FTR's

- Identificación de FTR´s

SIV	- Vehículo - Accesorio_vehículo	EIF
SCV	- Compra	ILF
SCorreo	- Correo	ILF

- Complejidad EIF
  - Vehículo (Numero serie, Marca, Modelo, Año-modelo, Color, Kilometraje, Condición, Foto, Precio, Disponibilidad, Nombre accesorio)
  - Para 1 RET y 11 DET's la complejidad es Baja.

Identificación de FTR's

- Complejidad ILF
  - Compra (Número compra, Número serie, Nombre comprador, Dirección comprador, Tel comprador, E mail, Compañía seguros, Agente compañía seguros, Teléfono compañía seguros, Fax compañía seguros, Nombre inst. financiera, Teléfono inst. financiera, Fax inst. Financiera)
    - Para 1 RET y 13 DET's la complejidad es Baja.
  - Correo (Número compra, Nombre comprador, Nombre de la aseguradora, Nombre de la financiera, Marca del vehículo, Modelo, Año-modelo, Color, Accesorios, Precio)
    - Para 1 RET y 10 DET's la complejidad es Baja.

Identificación de Transacciones

- Identificación de Transacciones
  - Pantalla 1 (Drop-downs):
    - Año-modelo (flecha y campo)
      - Para 2 DET's y 1 FTR la complejidad para EQ es Baja.
    - Marca-modelo (flecha y campo)
      - Para 2 DET's y 1 FTR la complejidad para EQ es Baja.
    - Accesorios (flecha y campo)
      - Para 2 DET's y 1 FTR la complejidad para EQ es Baja.
    - Rango de precios (flecha y campo, se requiere cálculo)
      - Para 2 DET's y 1 FTR la complejidad para EO es Baja.

Facultad de Ciencias

Identificación de Transacciones

- Pantalla 2 (Marca, Modelo, Año-modelo, Precio, Color, Condición, Radio button, Botón “Ver Detalle”). Se obtienen todos los datos de un FTR (Vehículo):
  - Para 8 DET’s y 1 FTR la complejidad para EQ es Baja.
- Pantalla 3 (Color, Kilometraje, Condición, Numero serie, Precio, Accesorios (aunque se repite), Botón “Comprar”, Foto). Se obtienen todos los datos de un FTR (Vehículo):
  - Para 8 DET’s y 1 FTR la complejidad para EQ es Baja.

Identificación de Transacciones

- Pantalla 4 (Nombre comprador, Dirección comprador, Tel comprador, E mail, Compañía seguros, Agente compañía seguros, Teléfono compañía seguros, Fax compañía seguros, Nombre inst. financiera, Teléfono inst. financiera, Fax inst. Financiera, Botón “Comprar”, Número compra procesamiento del mail, Marca vehículo, Modelo, Año-modelo, Color, Accesorios, Precio). Se insertan los datos en Compra y se actualiza un Vehículo, = 2 FTR’s):
  - Para 19 DET’s y 2 FTR la complejidad para EI es Alta.

Facultad de Ciencias

Cálculo de PF no ajustados (UFP)

Nombre	Tipo	Complejidad	UFP
Consultar listado año-modelo	EQ	Baja	3
Consultar listado marca-modelo	EQ	Baja	3
Consultar listado accesorios	EQ	Baja	3
Mostrar listado rango precios	EO	Baja	4
Vehículo	EIF	Baja	5
Compra	ILF	Baja	7
Correo	ILF	Baja	7
Consultar listado vehículos	EQ	Baja	3
Consultar detalle vehículos	EQ	Baja	3
Insertar compra	EI	Alta	6
Total			**44**

Facultad de Ciencias

Cálculo del valor de factor de ajuste (VAF)

Característica general del sistema	Valor (Caso 1)	Valor (Caso 2)
Comunicación de Datos	5	0
Procesamiento de Datos Distribuidos	5	0
Rendimiento	5	0
Configuración de Uso Intensivo	5	0
Nivel de Transacciones	5	0
Entrada de Datos en Línea	5	0
Eficiencia del Usuario Final	5	0
Actualización en Línea	5	0
Procesamiento Complejo	5	0
Reutilización	5	0
Facilidad de Instalación	5	0
Facilidad de Operacional	5	0
Múltiples Sitios	5	0
Facilidad de Cambio	5	0
SUMA (TDI)	**70**	**0**
VAF = (TDI x 0.01) + 0.65	**1.35**	**0.65**

Cálculo de Function Points Ajustados

- Resultado de PF ajustados (AFP)

Una vez calculados los puntos de función sin ajustar y el valor de factor de ajuste se procede a calcular los puntos de función ajustados. En este ejemplo se calcularon dos valores de factor de ajuste con el propósito de ilustrar cuál es el rango en el que pueden estar los puntos de función al ajustarse. Resultado con factor de ajuste ± 35%:

Total UFP	44
AFP C1	59.4
AFP C2	28.6

Facultad de Ciencias

Contacto

- Jesús Iván Saavedra Martínez
- jism@ciencias.unam.mx

Universidad Nacional Autónoma de México

UNAM

Facultad de Ciencias

Métricas de Software

Introducción a la medición del software con COSMIC

Jesús Iván Saavedra Martínez

Octubre 2017

COSMIC

- Introducción
- Fase de estrategia de medición
- Fase de representación
- Fase de medición
- Informe de medidas

Facultad de Ciencias

Agenda

I. Evolución de los Métodos de Medición de Tamaño Funcional (FSM)

II. Aplicabilidad del Método COSMIC

III. Requisitos Funcionales y NO Funcionales de Usuario

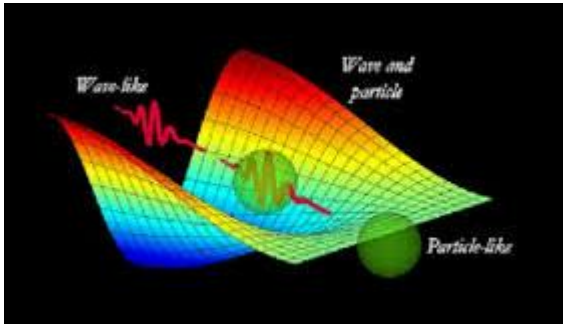
IV. Principios Fundamentales de COSMIC

V. El Proceso de Medición COSMIC

Problema de la medición del software

- Desarrollo de Software = Arte.
- Software = Producto Intelectual.
- Es más cualitativo que cuantitativo.
- Muy complejo.
- . . .

Características de Modelos de Medición



Wave model of light

- **CONCEPTO**: específico, bien definido, claro, represente la realidad.
- **PROCEDIMIENTO DE MEDICIÓN**: Generalmente aceptado, bien soportado, pueda ser replicado.
- **UNIDAD**: Consistente, clara, referencia (Etalón).

Una analogía

200 m2

400 m2

400 m2

¿Qué diferencia hay en construir las casas de 400 m^2? El tamaño es el mismo, el cómo y qué vas a utilizar para construirlo es la diferencia.

¿Cuál es el tamaño correcto?

Tamaño de proyecto:

Esfuerzo en HH / meses/ $$$

"Estándares Internacionales"

Tamaño del Software:

Tamaño Funcional

¿Qué se puede medir en el software?

Factores Técnicos

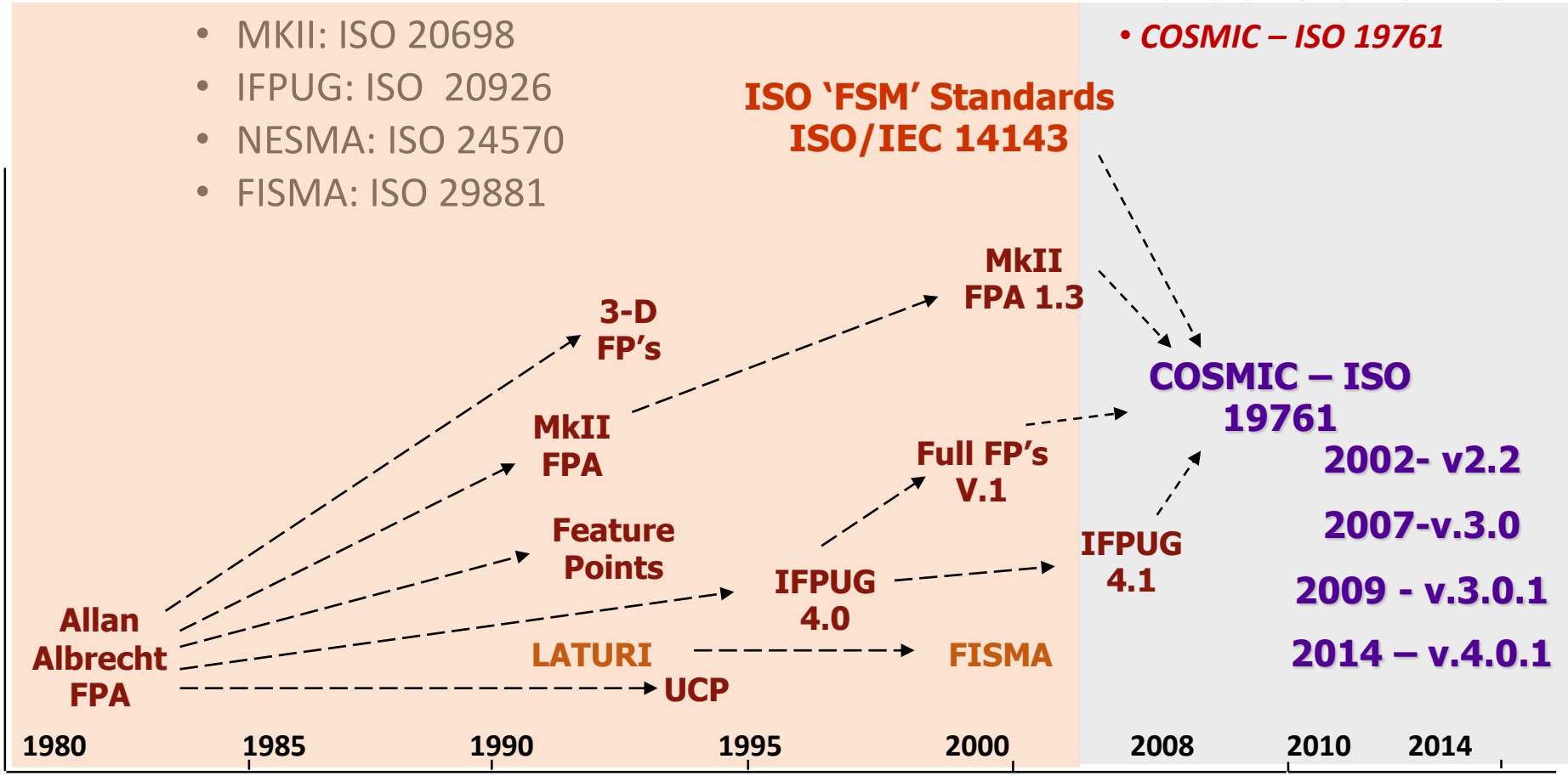
- Significante para desarrolladores pero no para usuarios y administración
- Pobre portabilidad
- Se conocen precisamente muy tarde
- No hay un estándar internacional

Factores Funcionales

- Significativo para los managers y usuarios y staff técnico
- Portable
- Puede ser medido temprano
- **Basado en estándar internacional!**

Evolución de los Métodos de Medición de Tamaño Funcional

- **1st Generation FSM**:
  - MKII: ISO 20698
  - IFPUG: ISO 20926
  - NESMA: ISO 24570
  - FISMA: ISO 29881
- **2nd Generation FSM**:
  - *COSMIC – ISO 19761*



COSMIC

- COSMIC fue diseñado para dar cumplimiento al estándar **ISO**

- COSMIC ha sido diseñado POR la industria CON la industria.

Facultad de Ciencias

Agenda

I. Evolución de los Métodos de Medición de Tamaño Funcional (FSM)

II. Aplicabilidad del Método COSMIC

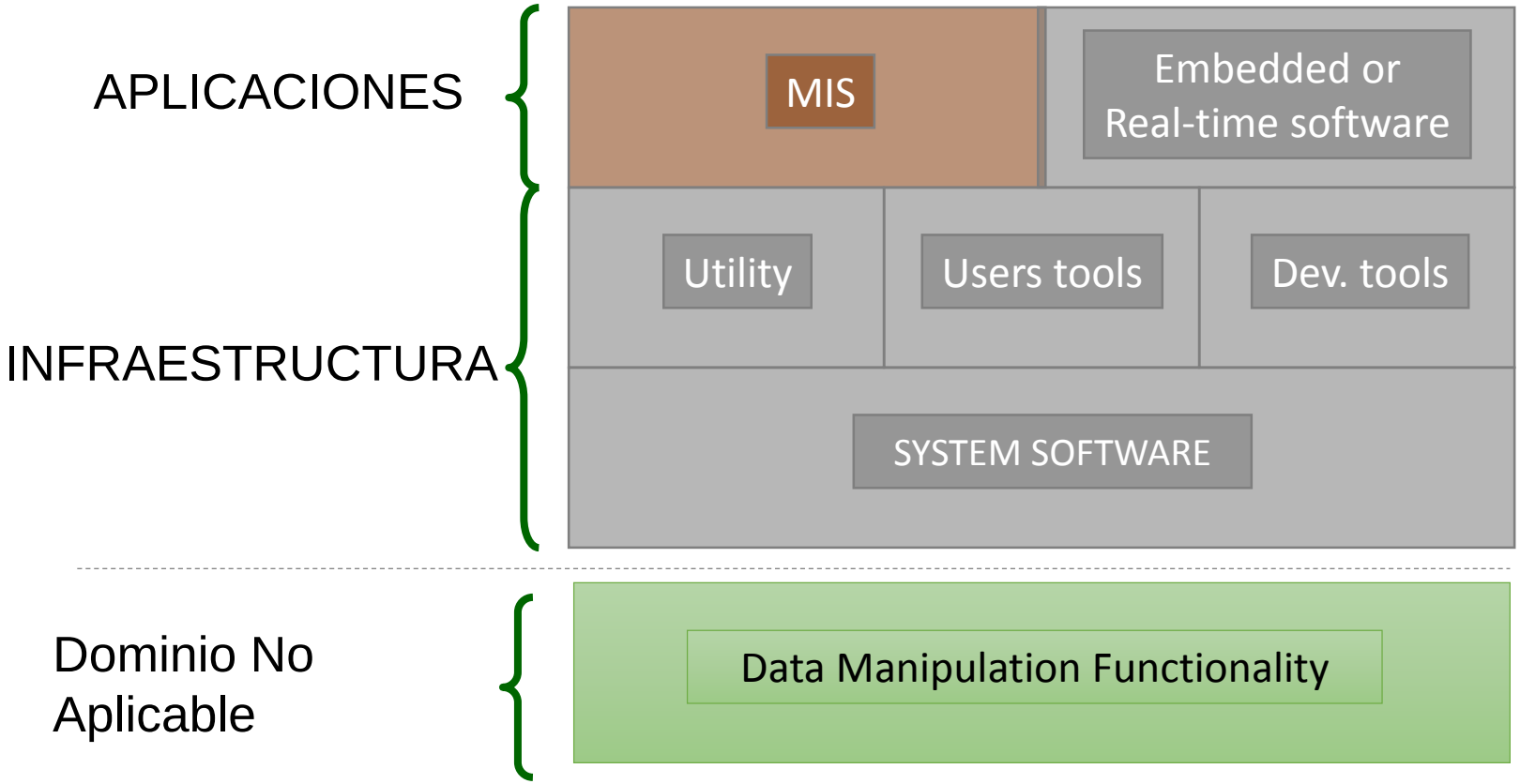
III. Requisitos Funcionales y NO Funcionales de Usuario

IV. Principios Fundamentales de COSMIC

V. El Proceso de Medición COSMIC

Ámbitos funcionales

1ª Generación: MIS - Management Information Systems
2ª Generación: Real-time and multi-layered software



Ámbitos funcionales

Dominio No Aplicable

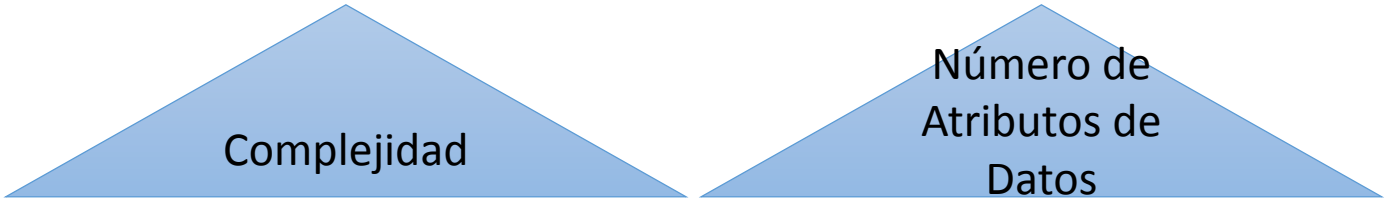
Data Manipulation Functionality

Si es posible, definir una extensión local al método de medición COSMIC

COSMIC Method Update Bulletin #8

Proposal to remove limitations on the scope of applicability of the COSMIC Method v3.0.1 as defined in the 'Measurement Manual'

- **Posibles** limitaciones en los factores que contribuyen al tamaño funcional



Funcionalidad de Software

- ‘El procesamiento de la información que el software debe realizar para sus usuarios’.

Requerimientos Técnicos

Requerimientos Funcionales

Definición de Tamaño Funcional

- ISO 14143 Information technology -- Software measurement -- Functional size measurement
  - Parte 1: Definición de conceptos
  - Parte 2: Evaluación de la conformidad
  - Parte 3: Guía de verificación
  - Parte 4: Modelo de referencia
  - Parte 5: Determinación de dominios funcionales
  - Parte 6: Guía para el uso de ISO/IEC 14143

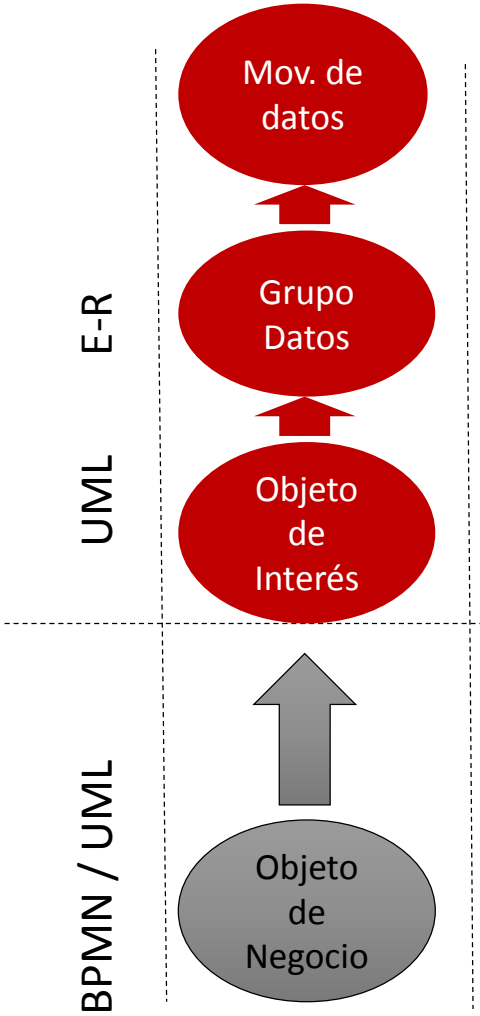
Tamaño Funcional es: "Un tamaño de software derivado de cuantificar los Requisitos Funcionales de Usuario – FUR"

Facultad de Ciencias

Agenda

I. Evolución de los Métodos de Medición de Tamaño Funcional (FSM)
II. Aplicabilidad del Método COSMIC
III. Requisitos Funcionales y No Funcionales de Usuario
IV. Principios Fundamentales de COSMIC
V. El Proceso de Medición COSMIC

Requisitos Funcionales de Usuario



DEFINICIÓN – Requisitos Funcionales de Usuario (Functional Users Requirements, FUR)

Subconjunto de los Requisitos de los Usuarios. Requisitos que describen lo que el software deberá hacer, en términos de tareas y servicios.

***IEEE* 830** Guide to *Software Requirements* Specifications

***IEEE* 1233** Guide for Developing System Requirements Specifications

IEEE 1362 Guide for Information Technology - System Definition - Concept of Operations (ConOps) Document

Requisitos Funcionales de Usuario

DEFINICIÓN - Requisitos Funcionales de Usuario (Functional Users Requirements, FUR)

- ✓ Los requisitos funcionales de los usuarios incluyen, pero no están limitados a:
  - Transferencia de datos (por ejemplo de entrada de datos de clientes; enviar señal de control)
  - Transformación de datos (por ejemplo calcular los intereses bancarios; derivar temperatura media)
  - Almacenamiento de datos (por ejemplo pedidos de clientes; recopilar datos de temperatura ambiente durante un tiempo)
  - Recuperación de datos (por ejemplo listas de los empleados actuales; recuperar la última posición de una aeronave)

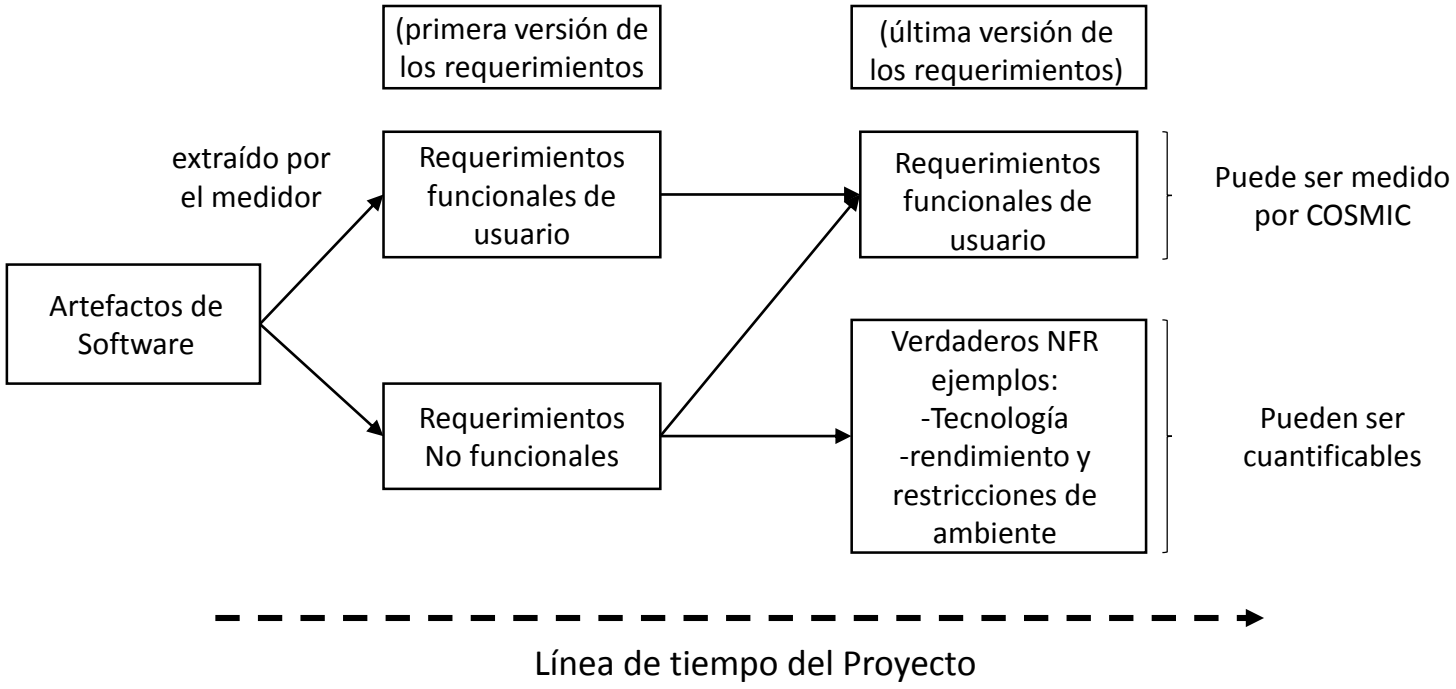
Requisitos No Funcionales de Usuario

DEFINICIÓN – Requisistos NO-Funcionales (Non-Functional Requirements) (de software)

- ✓ Cualquier requisito para la parte de un sistema de software/hardware o producto de software, incluyendo la forma en que debe ser desarrollado y mantenido, y cómo debe desempeñarse en la operación, excepto algún requisito funcional de usuario. NFR se refieren a:
  - la calidad de software;
  - El ambiente en que el software debe implementarse y debe servir;
  - los procesos y la tecnología que se utilizará para desarrollar y mantener el software;
  - la tecnología que se utiliza para la ejecución de software

- NOTA: Los requisitos del sistema o de software que se expresan inicialmente como No Funcional menudo evolucionan a medida que el proyecto avanza, total o parcialmente en FUR de un software.

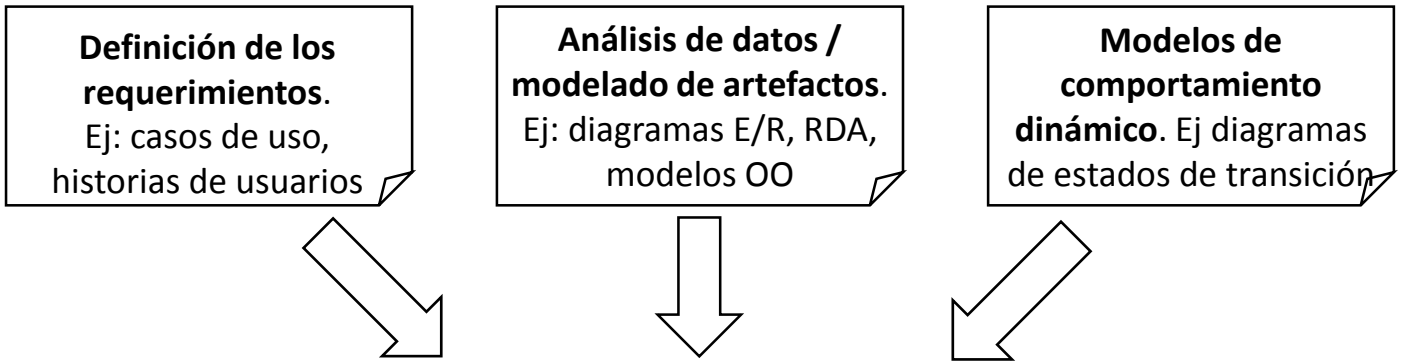
Requisitos No Funcionales de Usuario



Varios requisitos aparecen inicialmente como NFR evolucionan en FUR conforme avanza el proyecto

Extracción de los requisitos funcionales de usuario

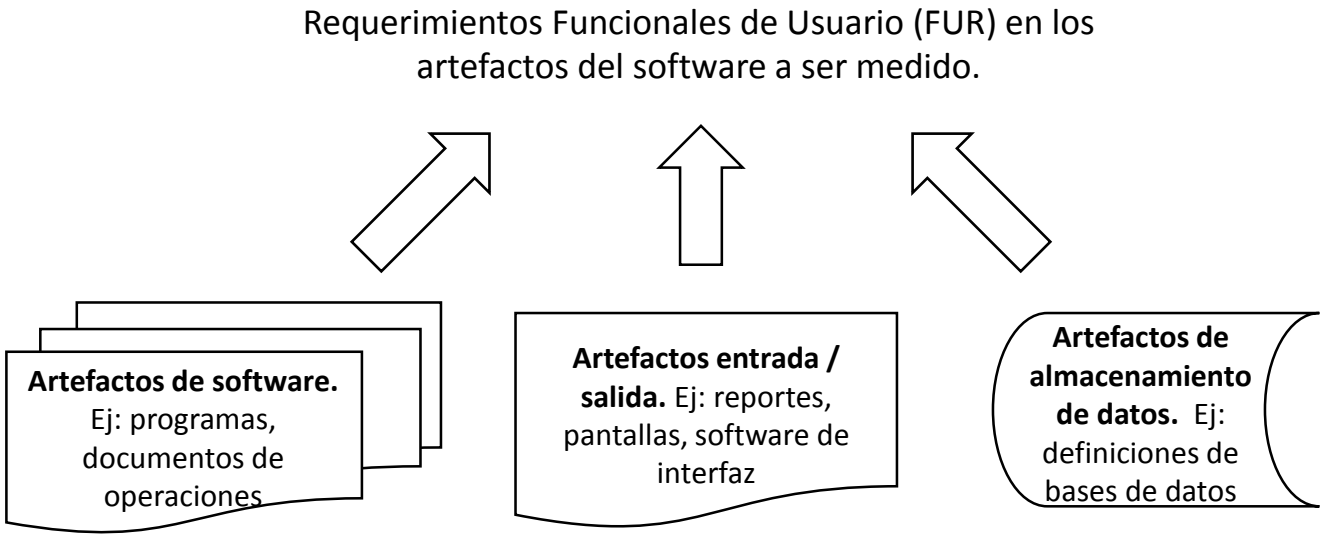
- Los FUR pueden ser derivados de artefactos de ingeniería de software que se producen **ANTES** de que el software exista.



Fuentes de Pre-implementación de los Requisitos Funcionales de Usuario

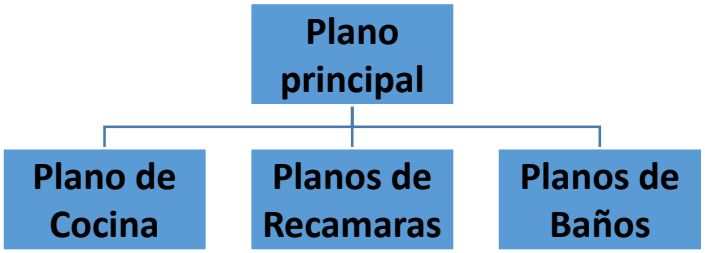
Derivando los requisitos funcionales de usuario

- Los FUR pueden ser identificados **DESPUÉS** de que el software ha sido construido.



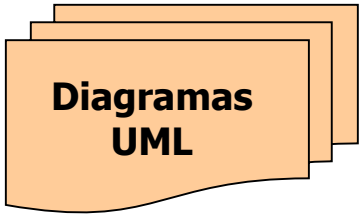
Fuentes de Post-implementación de los Requisitos Funcionales de Usuario

Una analogía…

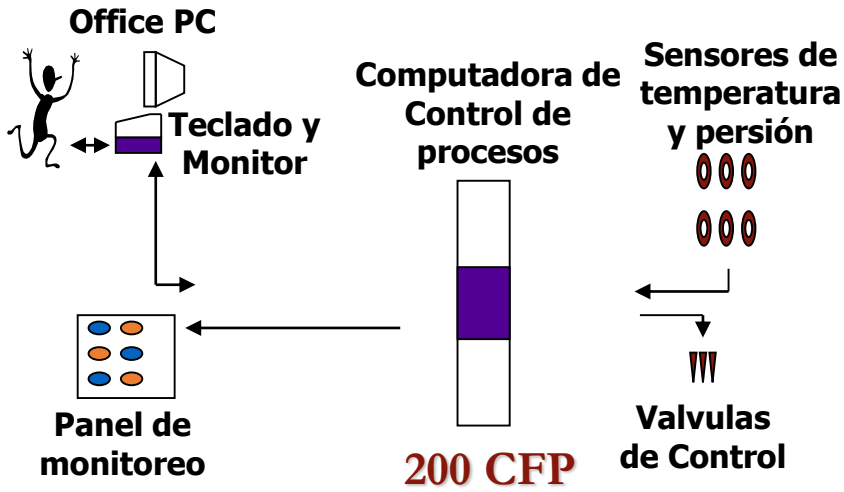


200 m^2

200 m^2



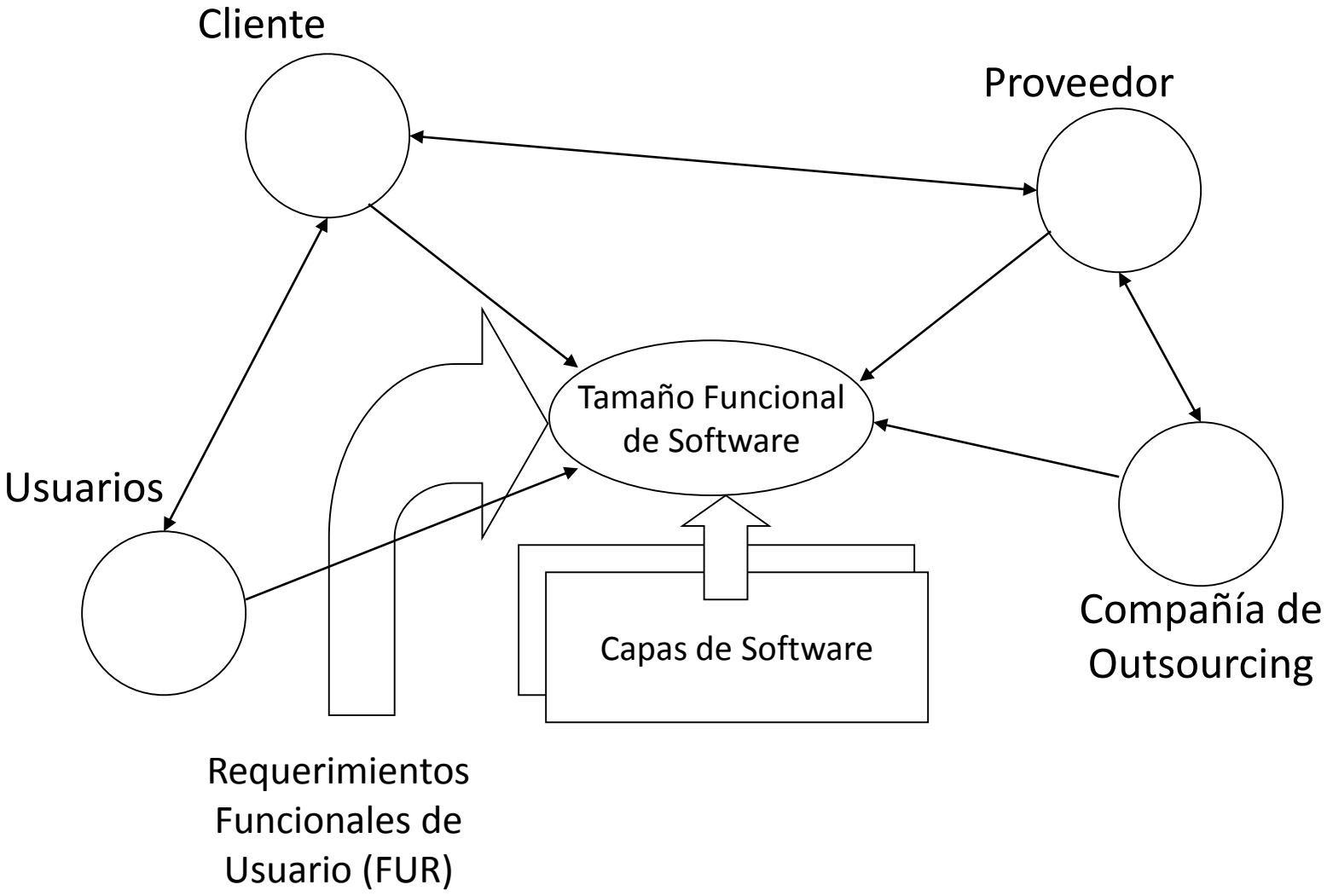
200 CFP



200 CFP

CFP =Unidad de Tamaño Funcional= COSMIC Function Points

Diferentes vistas de un mismo software



Facultad de Ciencias

Agenda

I. Evolución de los Métodos de Medición de Tamaño Funcional (FSM)
II. Aplicabilidad del Método COSMIC
III. Requisitos Funcionales y NO Funcionales de Usuario
IV. Principios Fundamentales de COSMIC
V. El Proceso de Medición COSMIC

Los Principios Fundamentales del Método COSMIC

- Basado en **Principios** no en **Reglas**
- **Unidad de medida** claramente definida
- No se deriva del esfuerzo
- No se deriva de una arquitectura específica
- Diseñado a partir del ISO 14143

Los Principios Fundamentales del Método COSMIC

4 Principios

3 Fases

Los Principios Fundamentales del Método COSMIC

1. La funcionalidad del software se conforma de **procesos funcionales**.
2. Los procesos funcionales consisten en **subprocesos** que ***mueven*** o ***manipulan*** datos
3. Cada movimiento de datos mueve un grupo de datos que describe una cosa
4. La manipulación de datos se considera incluida para cada movimiento de datos

Los Principios Fundamentales del Método COSMIC

- El método COSMIC se basa en sólidos principios de ingeniería de software. Estos principios se resumen en dos modelos.
  - 'Modelo Contextual de Software' permiten a un medidor definir el software a ser medido y lo que incluye la medición del tamaño.
  - 'Modelo Genérico de Software' define cómo los FUR del software que se va a medir se modelan para que puedan ser medidos

El Modelo Contextual de Software de COSMIC

Principios – Modelo Contextual de Software de COSMIC

a) El software está limitado por el hardware.

b) El software está típicamente estructurado en **capas**.

c) Una capa puede contener una o más aplicaciones de software semejante

d) Cualquier aplicación software que deba medirse, se define por su **alcance** de medición, que se limitará en su totalidad dentro de una sola capa.

e) El alcance de la aplicación de software debe medirse en función del propósito de la medición.

f) Los **usuarios funcionales** de una aplicación de software se identificarán a partir de los FUR de la aplicación software que se medirá como los emisores y/o destinatarios de los datos al/desde el software respectivamente.

El Modelo Contextual de Software de COSMIC

Principios – Modelo Contextual de Software de COSMIC

g) Los FUR de software pueden expresarse en distintos niveles de granularidad.

h) Una medición precisa en COSMIC del tamaño de una pieza de software requiere que sus FUR sean conocidos en un nivel de granularidad en el que se puedan identificar sus **procesos funcionales** y subprocesos.

i) Una medición aproximada en COSMIC de una pieza de software es posible si sus FUR se miden a un alto nivel de granularidad por un enfoque de aproximación y es escalado al nivel de granularidad de procesos funcionales y subprocesos.

El Modelo Genérico de Software de COSMIC

Principios – Modelo Genérico de Software de COSMIC

a) Una pieza de software interactúa con sus usuarios funcionales a través de una **frontera**, y con un **almacenamiento persistente** que existe dentro de esa frontera.

b) Las requisitos funcionales de los usuarios de la aplicación software a medir pueden representarse como procesos funcionales únicos.

c) Cada proceso funcional se conforma por sub-procesos.

d) Un sub-proceso puede ser un **movimiento de datos** o una **manipulación de datos**.

e) Un movimiento de datos mueve un sólo **grupo de datos**.

El Modelo Genérico de Software de COSMIC

Principios – Modelo Genérico de Software de COSMIC

f) Existen cuatro tipos de movimientos de datos:

- Una **Entrada**, mueve un grupo de datos hacia un proceso funcional desde un usuario funcional.
- Una **Salida** mueve un grupo de datos fuera de un proceso funcional hacia un usuario funcional.
- Una **Escritura** mueve un grupo de datos desde un proceso funcional hacia un almacén persistente.
- Una **Lectura** mueve un grupo de datos desde un almacén persistente hacia un proceso funcional

g) Un grupo de datos consiste en un conjunto único de **atributos de datos** que describen un único **objeto de interés**.

El Modelo Genérico de Software de COSMIC

Principios – Modelo Genérico de Software de COSMIC

h) Cada proceso funcional es activado por un **movimiento desencadenante de entrada** de datos. El grupo de datos movido por el evento desencadenante es generado por un usuario funcional en respuesta a un **evento desencadenante.**

i) Un proceso funcional deberá incluir al menos un movimiento de entrada y también un movimiento de Salida o Escritura de datos, es decir que deberá incluir un mínimo de dos movimientos de datos. No existe un límite superior para el número de movimientos de datos en un proceso funcional.

j) Como una aproximación para fines de medición, los sub-procesos de manipulación de datos no se miden por separado; la funcionalidad de cualquier manipulación de datos se supone que se tomada en cuenta por el movimiento de datos con el que está asociado

Tipos contra Ocurrencias

- El ‘tipo’ de una cosa es una clase abstracta de todas las cosas que comparten alguna característica común. (Sinónimos de ‘tipo’ son ‘categoría’ o ‘clase’)

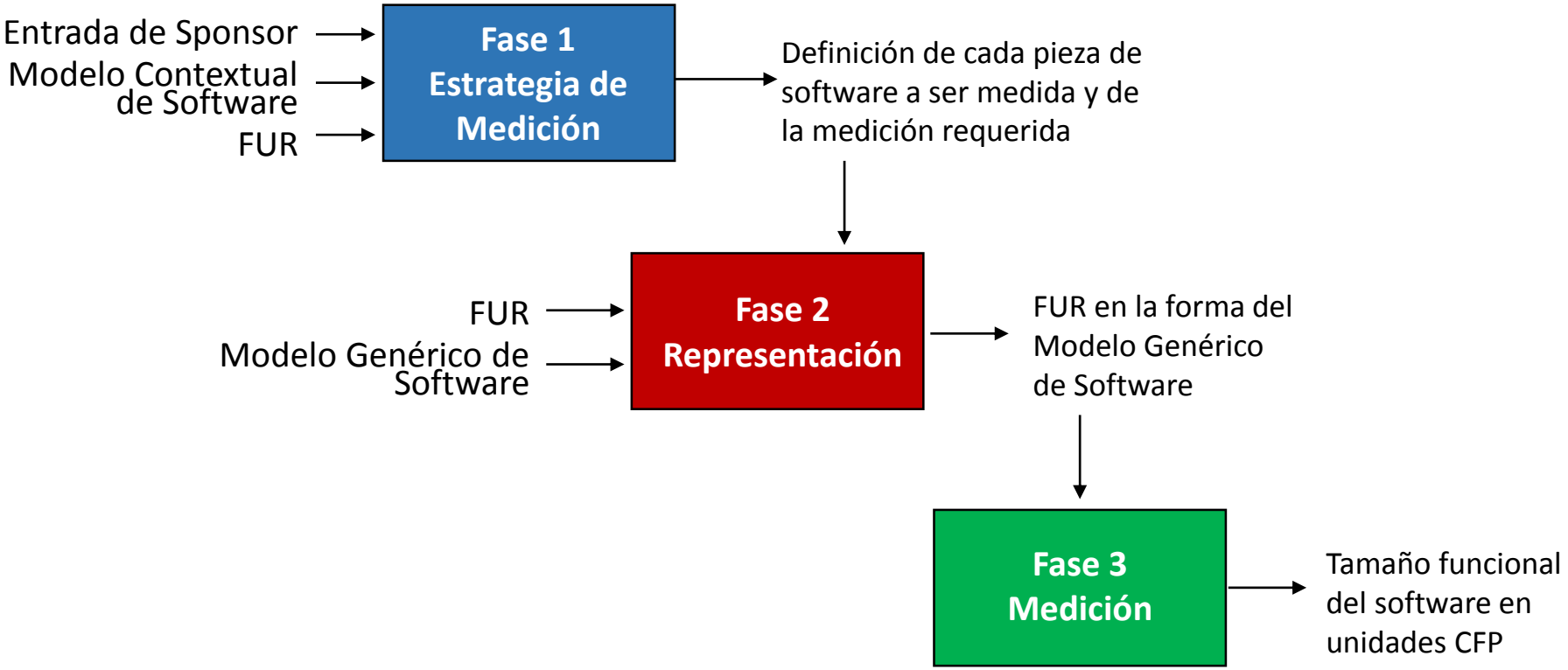
- Una ‘ocurrencia’ de una cosa es cuando la cosa aparece en la práctica, por ejemplo, en un contexto del mundo real, o cuando se produce un evento, o cuando un proceso se ejecuta por una persona o una computadora.

Facultad de Ciencias

Agenda

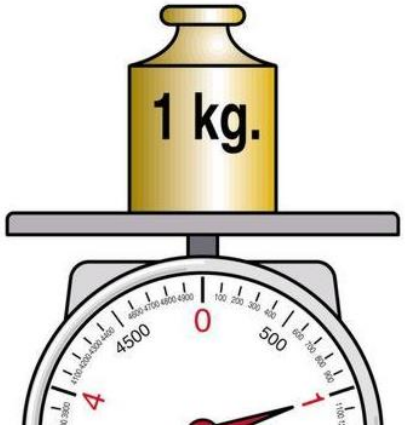
I. Evolución de los Métodos de Medición de Tamaño Funcional (FSM)
II. Aplicabilidad del Método COSMIC
III. Requisitos Funcionales y NO Funcionales de Usuario
IV. Principios Fundamentales de COSMIC
V. El Proceso de Medición COSMIC

El Proceso de Medición COSMIC



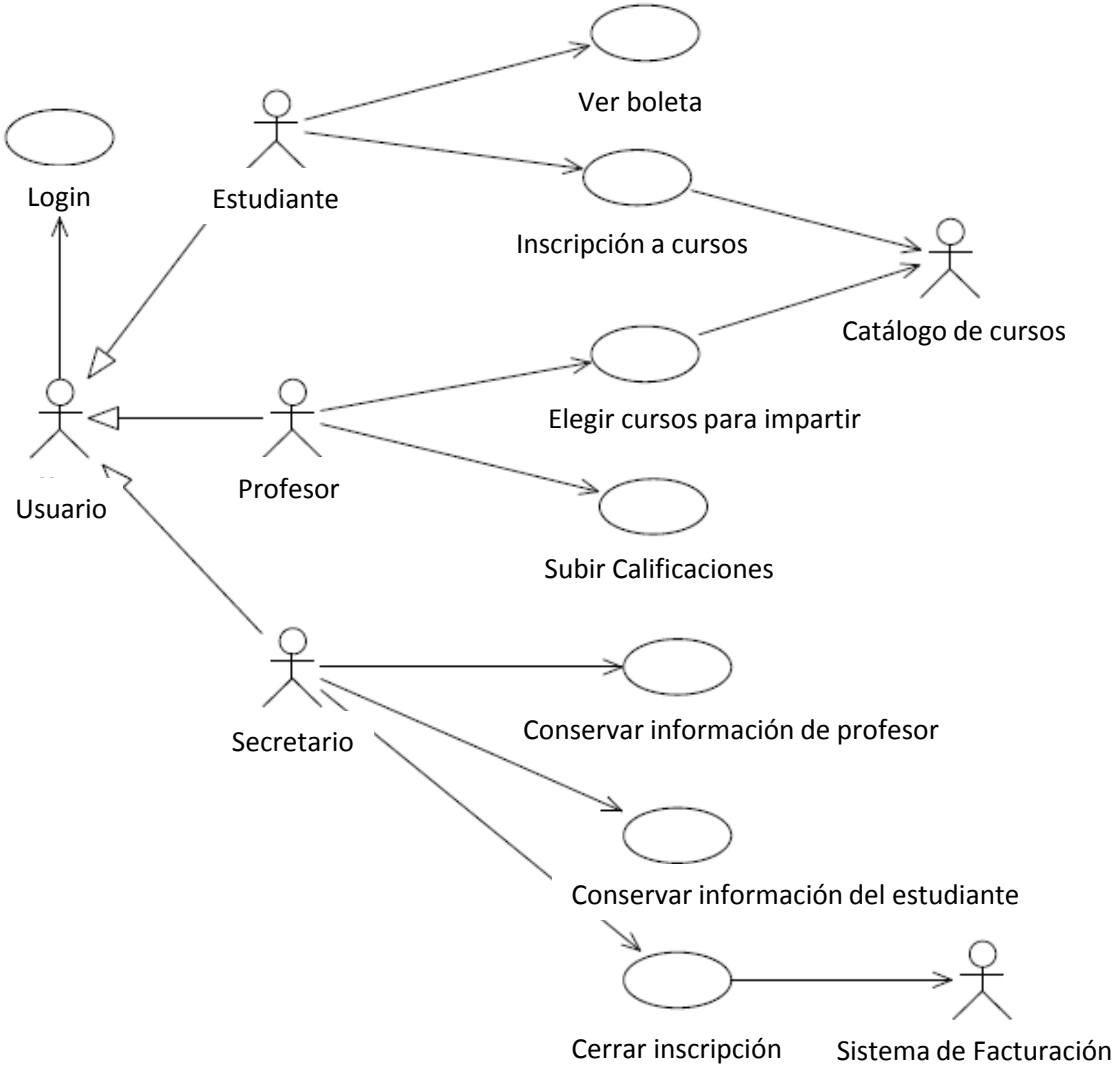
Unidad de Medida COSMIC

- 1 CFP (Cosmic Function Point)



Ejemplo CRS

C-Registration System



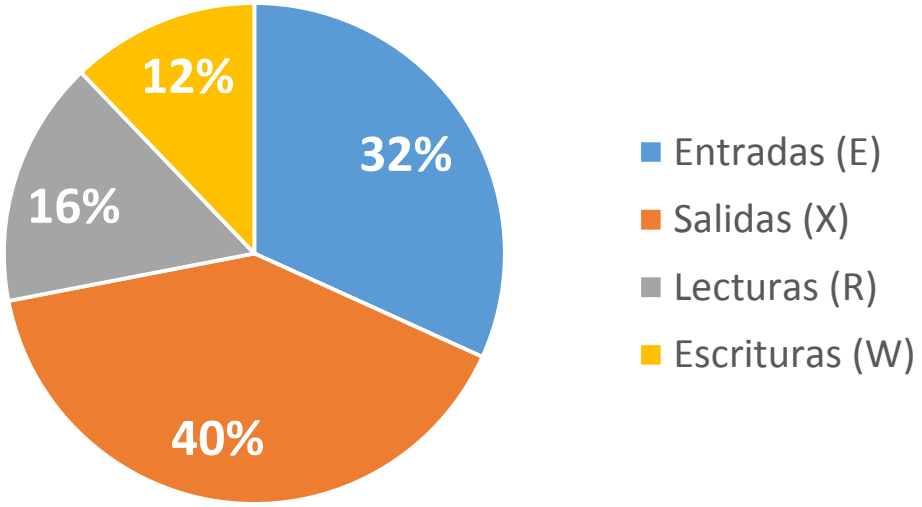
Facultad de Ciencias

Ejemplo CRS

No.	ID de requisitos	Procesos funcionales	CFP	%
1	1.2	Entrar	3	3
2	2.2.1	Agregar un profesor	5	5
3	2.2.2.1	Modificar un profesor	6	6
4	2.2.2.2	Eliminar un profesor	6	6
5	3.2	Elegir cursos para impartir	9	9
6	4.2.1	Agregar un estudiante	4	4
7	4.2.2.1	Modificar un estudiante	6	6
8	4.2.2.2	Eliminar un estudiante	6	6
9	5.2.1	Crear un itinerario	13	13
10	5.2.2.1	Modificar un itinerario	15	15
11	5.2.2.2	Eliminar un itinerario	7	7
12	7.2	Cerrar inscripciones	9	8
13	8.2	Presentar calificaciones	12	8
14	9.2	Ver boleta de calificaciones	6	4
	TOTAL	14	107 CFP	100%

Ejemplo CRS

Tipos de movimientos de datos	CFP	%
Entradas (E)	34	32
Salidas (X)	43	40
Lecturas (R)	17	16
Escrituras (W)	13	12
Total	107 (CFP)	100 (%)



Puntos importantes

Common Software Measurement International Consortium (COSMIC)

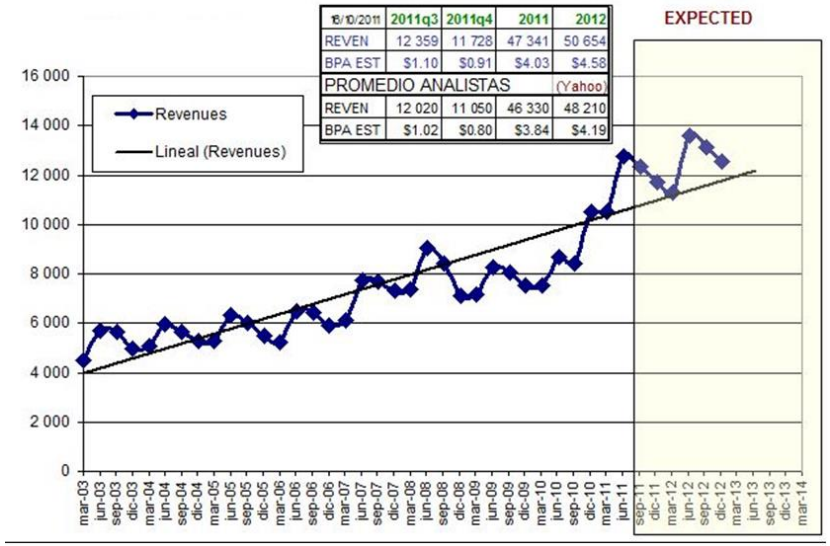
- Es un método de medición de tamaño funcional de software
- Es el único método de segunda generación.
- Está basado en la representación del software: E, X, W, R. No en datos estadísticos como en la primera generación.
- Tiene su equivalente en Norma Mexicana (NMX), que está incluida en el MAAGTICSI

Estructura de la Documentación

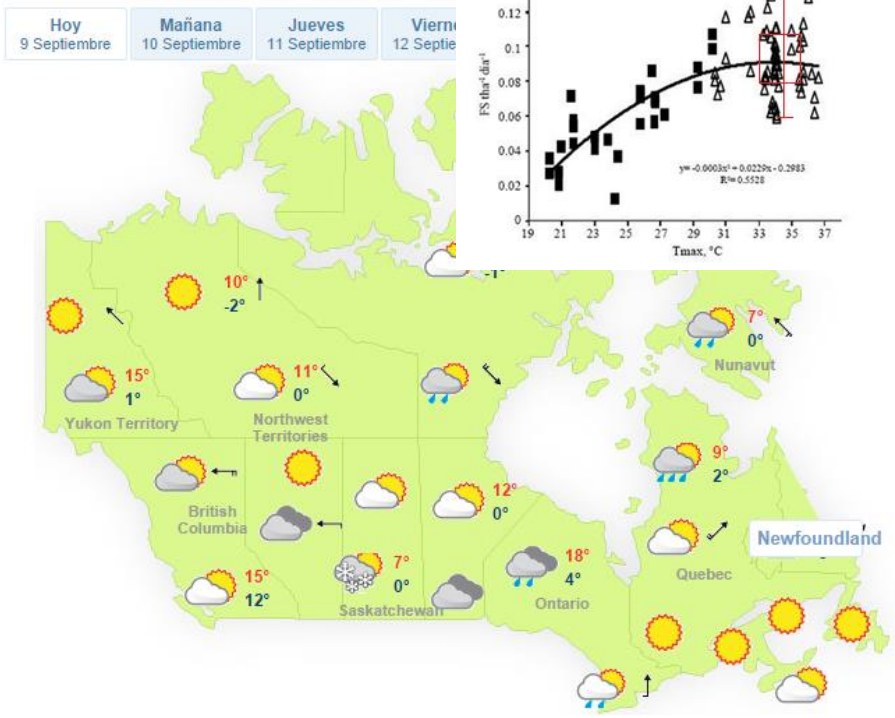
Visión General	Introducción al método COSMIC	Guía de referencias rápidas
Estándares	Estándar ISO 19761	Manual de Medición v4.0
Directrices:		
Dominio-específico	Tiempo-real, Negocios, Data Warehouse, SOA, etc.	
Método de Desarrollo específico	Agile, UML*, etc.	
Soporte de medición	Precisión de la Medición, Patrones de Medición, Aproximación de la Medición*, Convertibilidad, etc.	
Aplicación de la medición	Estimando*, Contabilizando NFR*, etc.	
Casos de estudio:	Tiempo-real, Negocios, Aplicaciones Web, etc.	

Importancia de la Medición del Tamaño de Software - Estimación

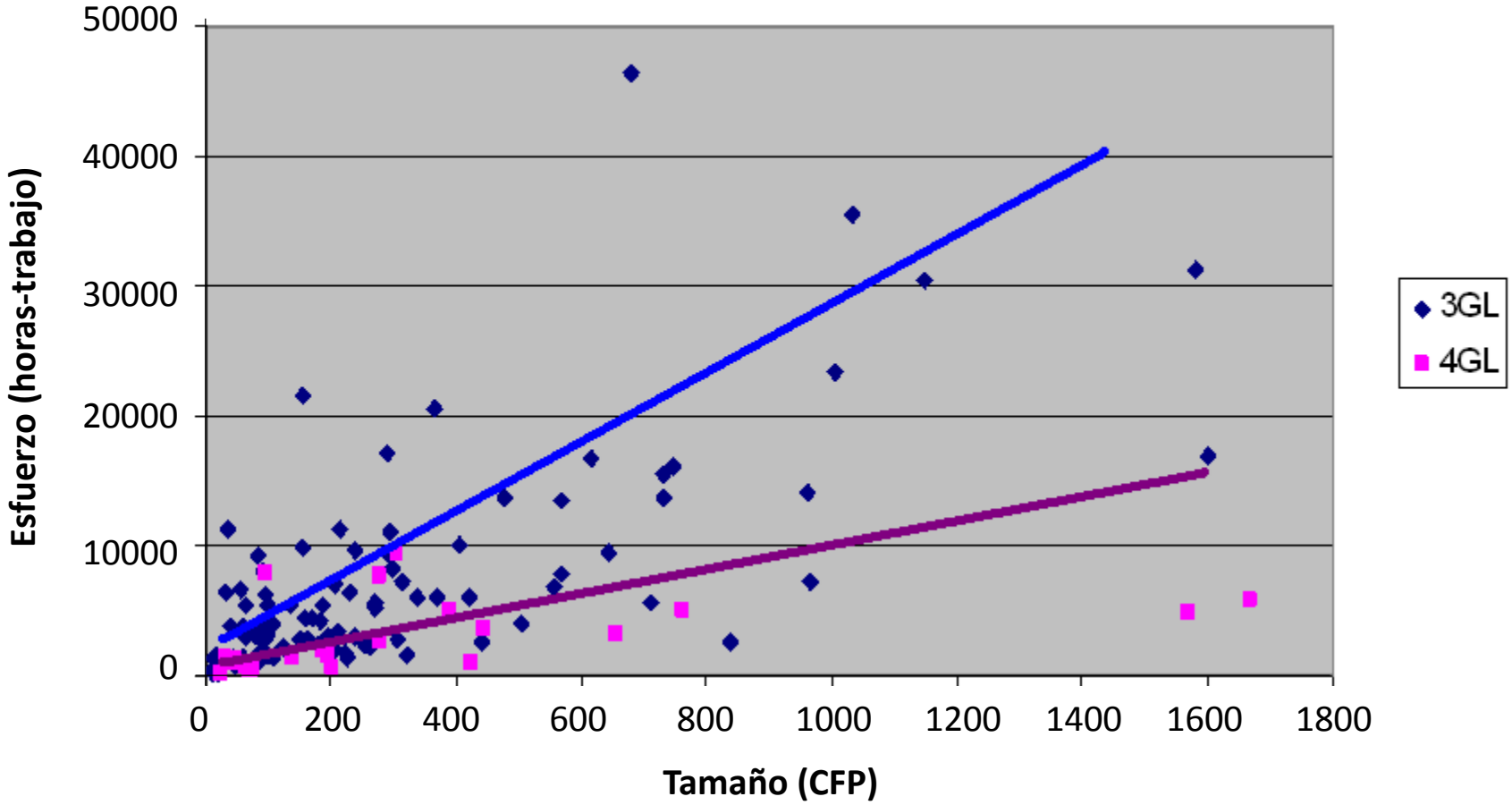
- Medición (GIGO - Std)



- Confianza



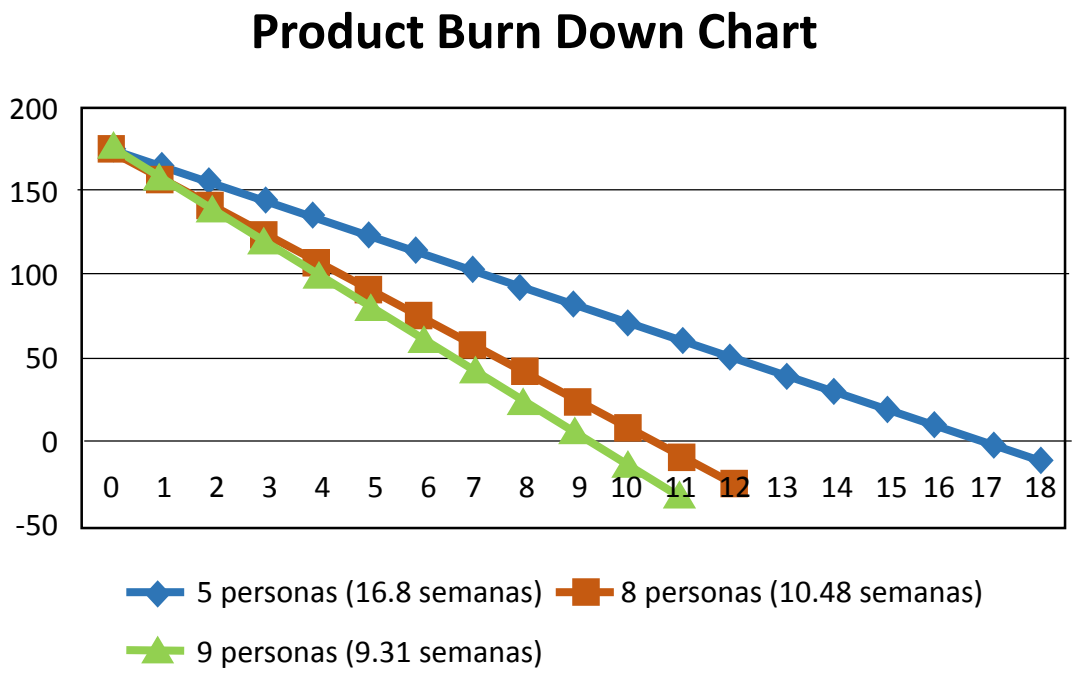
Estimación de Software



Importancia de la Medición del Tamaño de Software -> Desempeño proyecto

- En prácticamente cualquier industria, la capacidad de medir el desempeño es fundamental para comprender y mejorar la manera de operar.
- ‘Productividad' = 'Tamaño del software' / 'esfuerzo del proyecto'.
- ‘Velocidad de entrega’ = ‘Tamaño Software’ / tiempo transcurrido
- ‘Densidad de defectos'= Número de defectos identificados en un determinado período / ‘Tamaño del software'.
- Medidas de desempeño para comparación (benchmarking) externo/interno.

Importancia de la Medición del Tamaño de Software -> Control de Alcance



Tamaño Software	176
Esfuerzo Total	2934
Productividad Promedio	0.06
Duración de Iteración	1
Integrantes Equipo Desarrollo	5
Horas Disponibles x Semana	175
Horas Disponibles x Semana x Desarrollador	35

Importancia de la Medición del Tamaño de Software -> Evaluación de activos SW

- La mayoría de las grandes organizaciones, tanto en los sectores públicos o privados ya han acumulado grandes cantidades de software a través de la segunda mitad del siglo XX
- Determinar un tamaño de este activo o parte de el puede ser valioso a la hora de estimar el costo de remplazo.
- Algunas organizaciones, especialmente en los servicios financieros, utilizan una medida de tamaño para ayudar a valorar sus activos para que pueda aparecer en el balance de la organización.

Importancia de la Medición del Tamaño de Software -> Control de contratos SW

- Si su trabajo es controlar un contrato de suministro de software, entonces puede ser de vital importancia para poder dimensionar el software para todos los fines descritos anteriormente.

Visión Estratégica de Uso de COSMIC

Frank Vogelezang, 1er Congreso Nacional de Medición y Estimación de Software, México 2015

Facultad de Ciencias

Contacto

- Jesús Iván Saavedra Martínez
- jism@ciencias.unam.mx

Métricas de Software

Fase de Estrategia de Medición

Jesús Iván Saavedra Martínez

Octubre 2017

COSMIC

- Introducción
- Fase de estrategia de medición
- Fase de representación
- Fase de medición
- Informe de medidas

Fase de estrategia de medición

- Fase de estrategia de medición
  - Introducción
  - Definir el propósito de la medición
  - Definir el alcance de la medición
  - Identificando usuarios funcionales y el almacenamiento persistente
  - Identificando el nivel de granularidad
  - Observaciones finales

Parámetros clave

- El *propósito* de la medición, es decir, para lo que será utilizado el resultado.
- El *alcance global* del software a ser medido y, si el software se compone de más de una parte que deba ser medida por separado.
- Los *usuarios funcionales* de cada pieza de software a medir. Estos son los emisores y destinatarios de los datos a/desde el software que se desea medir.
- El *nivel de granularidad* de los artefactos disponibles del software que se va a medir

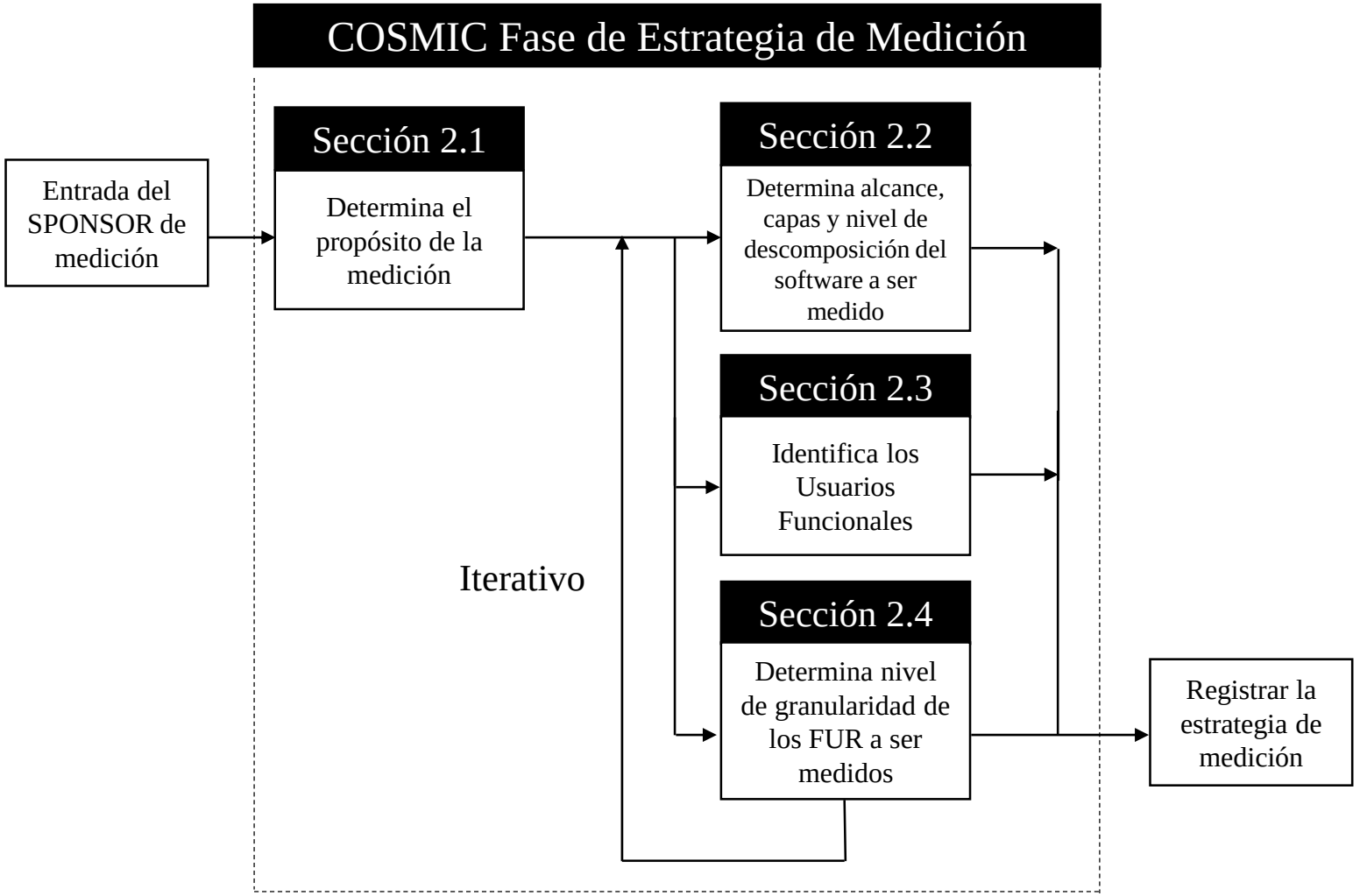
Parámetros clave

¿Para qué nos sirve determinar estos parámetros?

- Determinar estos parámetros nos ayuda a responder a las preguntas de ‘qué tamaño debe medirse’, ‘qué tan preciso queremos la medición’.

- Estos parámetros no son específicos de COSMIC pero requiere sean considerados con más cuidado que con otros métodos FSM

Proceso para determinar la Fase de Estrategia de Medición



Patrones de Estrategia de Medición

DEFINICIÓN - Patrón de Medición

- ✓ Una plantilla estándar que se puede aplicar en la medición de una pieza de software desde un ámbito funcional de software determinado, que define los tipos de usuario funcional que pueden interactuar con el software, el nivel de descomposición del software y los tipos de movimientos de datos que el software puede manejar.

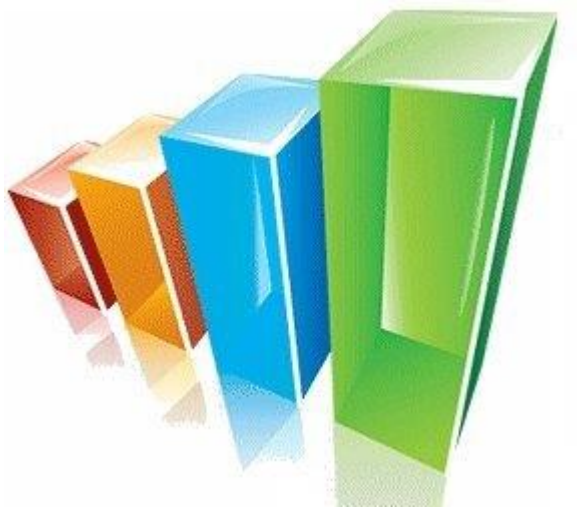
Fase de estrategia de medición

- Fase de estrategia de medición
  - Introducción
  - Definir el propósito de la medición
  - Definir el alcance de la medición
  - Identificando usuarios funcionales y el almacenamiento persistente
  - Identificando el nivel de granularidad
  - Observaciones finales

Propósito de la medición

DEFINICIÓN – Propósito de la medición
✓ Una declaración que define por qué una medición es necesaria, y para qué se utilizará el resultado.

- El propósito de una medición determina el tamaño que será medido



Propósito de la medición

El Medidor de una pieza de software debe decidir, en función de la finalidad de la medición:

- *cuándo* medir (antes, durante o después del desarrollo)
- *qué* medir (por ejemplo, todo el software que se entrega en un proyecto, o excluir el software reutilizado)
- *cuáles artefactos* utilizar para derivar los FUR a ser medidos (por ejemplo, una especificación de requisitos o el software instalado).

Ejemplos de propósitos de medición típicos

Para medir el tamaño de los FUR:

- a medida que evolucionan, como entrada a un proceso para estimar el esfuerzo de desarrollo.
- del software entregado, y del software que fue desarrollado, con el fin de obtener una medida de la reutilización funcional.
- del software existente como entrada para la medición del desempeño del grupo responsable de mantener y dar soporte al software.

Ejemplos de propósitos de medición típicos

- Para medir el tamaño de los cambios a los FUR después de que se hayan establecido inicialmente, con el fin de gestionar el alcance.
- Para medir el tamaño del subconjunto de la funcionalidad total del software que debe ser desarrollado.

La importancia del propósito

El propósito ayuda a los medidores a determinar:

- El alcance que debe medirse y por lo tanto los aparatos que serán necesarios para la medición.
- Los usuarios funcionales.
- El momento en el ciclo de vida del proyecto en el que la medición se llevará a cabo
- La precisión necesaria de la medición, y por lo tanto, si debería utilizarse la medición con el método COSMIC estándar, o si se debería utilizar una aproximación del método COSMIC

Fase de estrategia de medición

- Fase de estrategia de medición
  - Introducción
  - Definir el propósito de la medición
  - Definir el alcance de la medición
  - Identificando usuarios funcionales y el almacenamiento persistente
  - Identificando el nivel de granularidad
  - Observaciones finales

Alcance de la medición

DEFINICIÓN – Alcance de una medición
✓ El conjunto de los FUR que deben incluirse en un determinado ejercicio de medición de tamaño funcional.

- NOTA: (específico para el método COSMIC) Se debe hacer una distinción entre el ‘alcance global’, es decir, todo el software que debe medirse de acuerdo al propósito, y el ‘alcance’ de cualquier pieza de software individual dentro del alcance global, cuyo tamaño debe medirse por separado.

Alcance de la medición

REGLAS – Alcance de la medición

a) El alcance de cualquier pieza de software a ser medida se deriva del propósito de la medición.

b) El alcance de cualquier medición no se extenderá por más de una capa de software que se desea medir.

Derivando el alcance a partir del propósito de medición

- El software definido por un *alcance global* puede ser sub-dividida en piezas individuales de software cada uno con su propio *alcance de medición* definido de muchas maneras, dependiendo del propósito de la medición.

Ejemplo

Alcance global:

- Supongamos que un alcance global se define como ‘el portafolio de aplicaciones de una organización’ o como 'todas las piezas de software que se entregarán por un proyecto'. Las subdivisiones se podrían hacer debido a:
  - El software en diferentes capas (debido a la regla b)
  - La necesidad de distinguir los diferentes entregables para la medición del desempeño, la estimación de esfuerzo o con fines contractuales del software.

Alcance de la medición

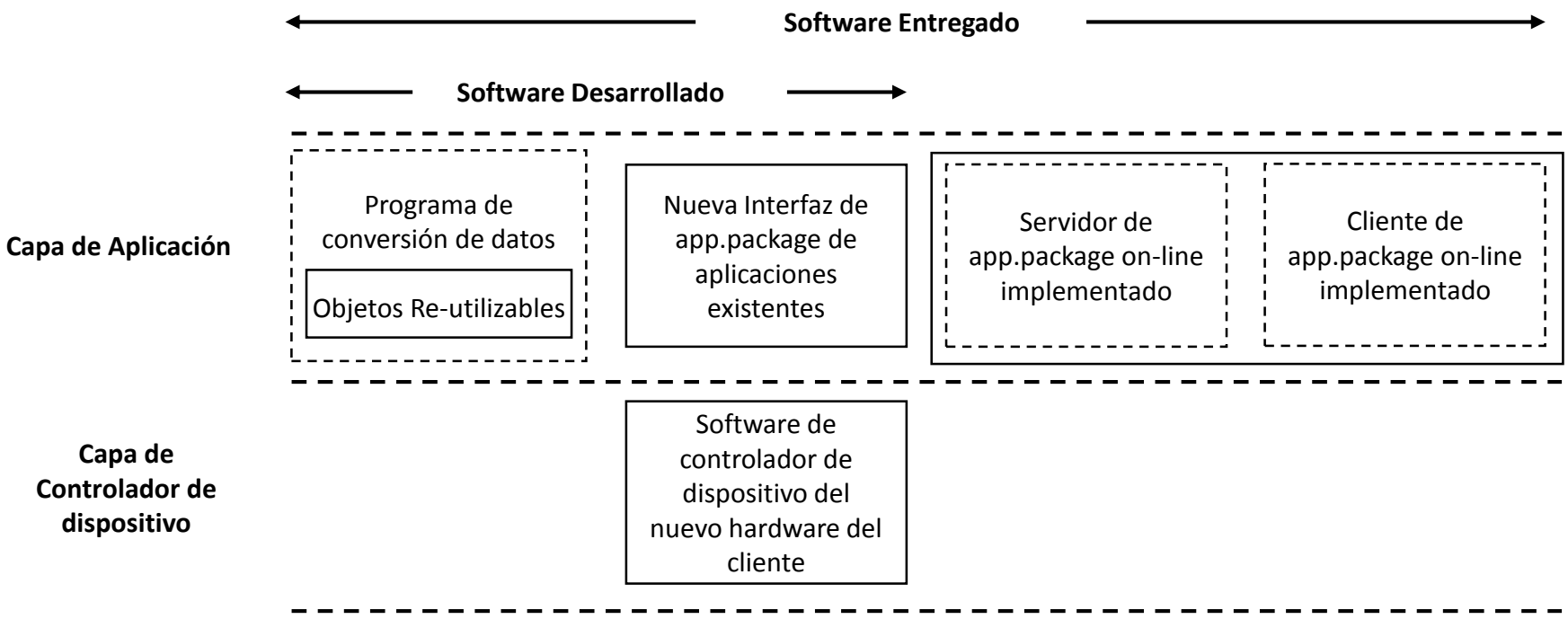
En resumen, el propósito de la medición debe ser siempre utilizado para determinar:

a) qué software es incluido o excluido del alcance global

b) la forma en que el software incluido puede que sea necesario dividirlo en partes separadas, cada una con su propio alcance, que deben medirse por separado.

- En la práctica, una especificación de alcance debe ser explícita en lugar de genérica.

Ejemplo



El alcance general de los entregables de un proyecto de software y los alcances de medición individuales

Capas

DEFINICIÓN – Capa
✓ Una partición funcional de una arquitectura de un sistema de software.

- Dado que el alcance de una pieza de software que debe medirse debe limitarse a una sola capa de software, el proceso para definir el alcance podrá requerir que el medidor primero tenga que decidir cuáles son las capas de la arquitectura del software.

Capas

PRINCIPIOS – Capas

- El software en una capa proporciona un conjunto de servicios que es coherente de acuerdo con algún criterio definido, y ese software puede utilizarse en otras capas sin saber cómo se implementan estos servicios.

- La relación entre el software en cualquiera de dos capas se define por una 'regla de correspondencia' que puede ser
  - Jerárquica: el software en la capa A tiene permitido utilizar el software en la capa B, y no viceversa
  - Bidireccional: el software en la capa A tiene permitido utilizar el software en la capa B, y viceversa.

Capas

PRINCIPIOS – Capas

- El Software en una capa intercambia grupos de datos con el software en otra capa a través de sus respectivos procesos funcionales.
- El Software en una capa no necesariamente utiliza todos los servicios funcionales proporcionados un software en otra capa.
- El Software en una capa de una arquitectura de software definida puede ser dividido en otras capas de acuerdo a diferentes arquitecturas de software definidas.

Componentes semejantes de Software o pares

DEFINICIÓN – Componentes semejantes de Software o pares
✓ Dos piezas de software son semejantes una a la otra sí se encuentran en la misma capa

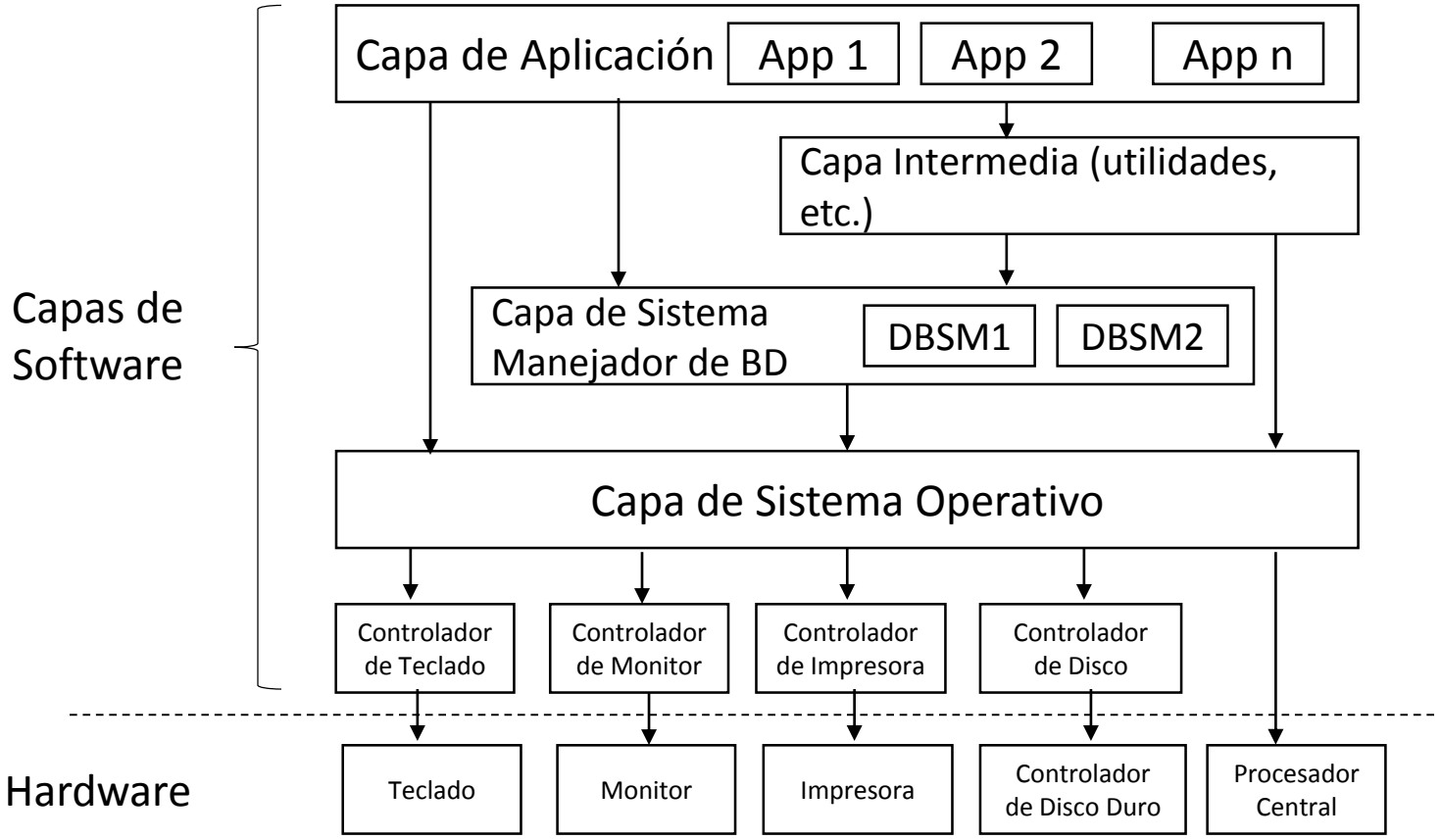
- Una medición puede referirse a dos o más piezas 'semejantes' de software, donde estas piezas se encuentran dentro de una misma capa en la arquitectura de software

Ejemplo 1. Tiempo real

El modelo OSI de 7 capas para las telecomunicaciones.

- Esto define una arquitectura en capas para que las reglas jerárquicas correspondientes para las capas del software de recepción de mensajes son inversas a las reglas para las capas del software de transmisión de mensajes.

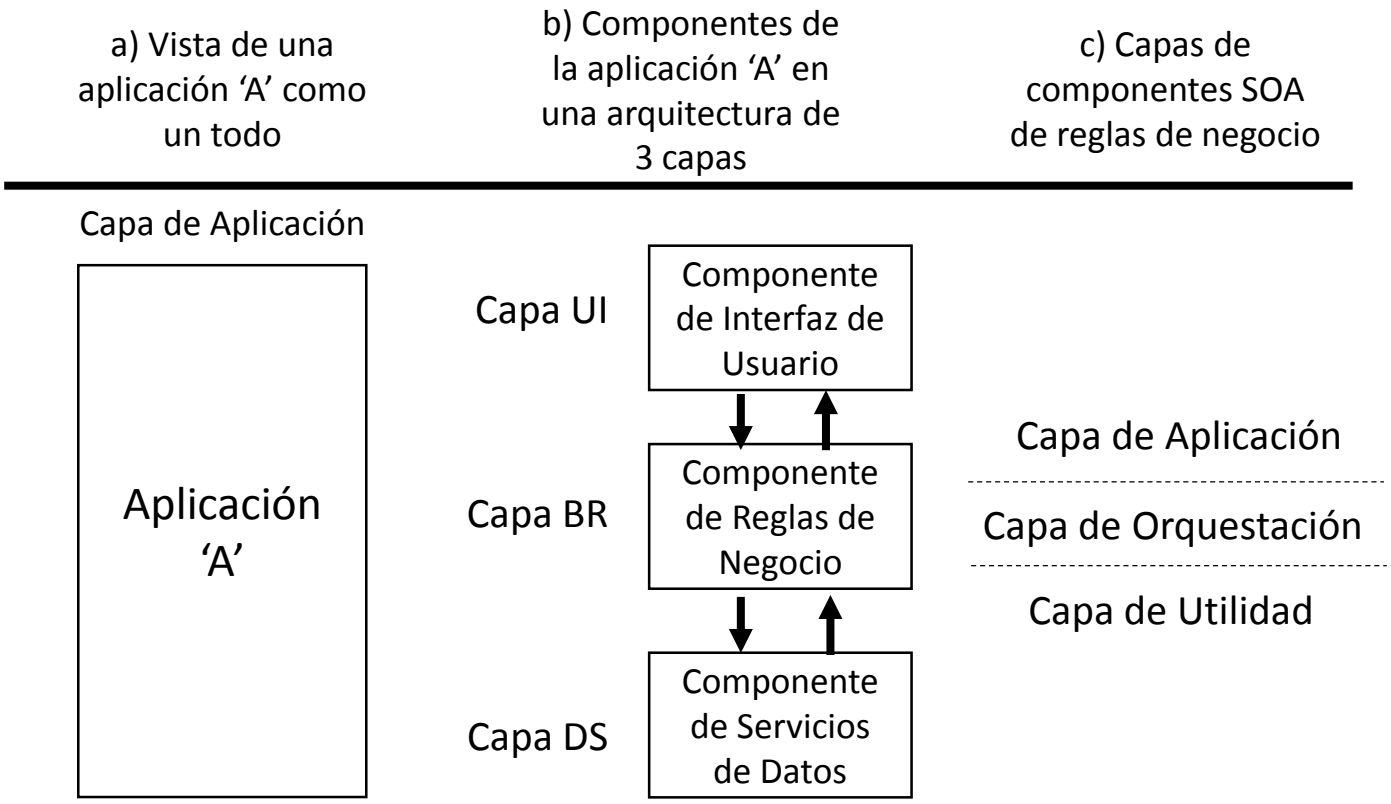
Ejemplo 2. Aplicación de Negocio



Típica arquitectura de software en capas de una aplicación de negocios

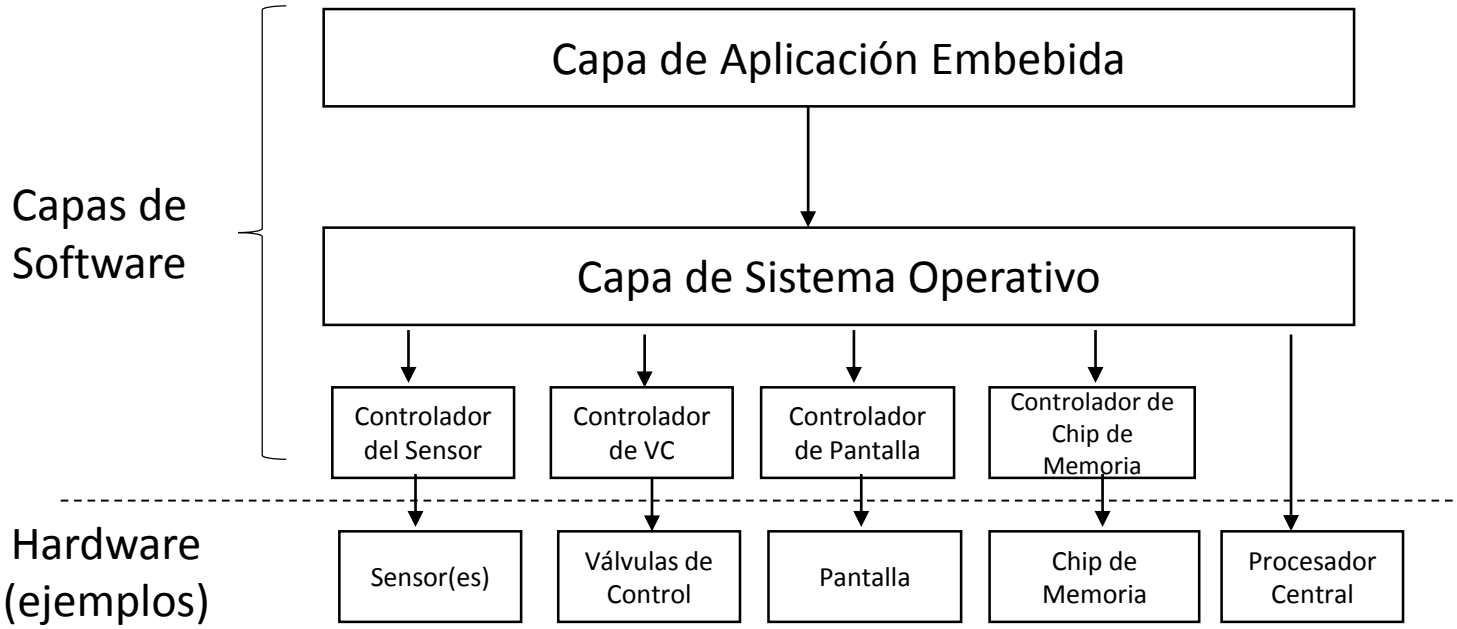
Ejemplo 3. Aplicación de Negocio

Una arquitectura de software puede presentar diferentes capas en función de la 'vista' de la arquitectura.



Tres vistas de capas de una aplicación

Ejemplo 4. Tiempo real



Típica arquitectura en capas de un sistema software embebido de tiempo real

Niveles de descomposición

DEFINICIÓN – Nivel de descomposición
✓ Cualquier nivel resultante de dividir una pieza de software en componentes (‘Nivel 1’, por ejemplo), después dividir dichos componentes en subcomponentes (‘Nivel 2’), y después dividir estos subcomponentes en sub-sub componentes (‘Nivel 3’), etc.

- NOTA 1: No debe confundirse con ‘nivel de granularidad’.
- NOTA 2: Las mediciones de tamaño de componentes de una pieza de software sólo pueden ser comparadas con componentes semejantes, por ejemplo, componentes del mismo nivel de descomposición.

Fase de estrategia de medición

- Fase de estrategia de medición
  - Introducción
  - Definir el propósito de la medición
  - Definir el alcance de la medición
  - Identificando usuarios funcionales y el almacenamiento persistente
  - Identificando el nivel de granularidad
  - Observaciones finales

Usuarios Funcionales

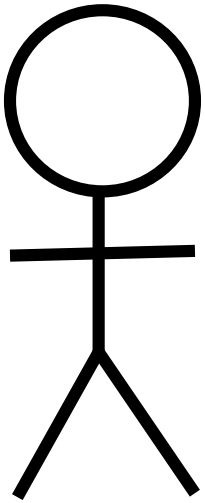
- Los diferentes tipos de usuarios pueden ‘ver’ una funcionalidad diferente y por lo tanto pueden obtener diferentes tamaños.

- En el caso del software, diferentes (tipos de) usuarios funcionales pueden requerir (a través de los FUR) diferente funcionalidad y por lo tanto los tamaños funcionales variarán con la elección de los usuarios funcionales.

Usuarios Funcionales

- Un ‘usuario’ se define como ‘cualquier cosa que interactúa con el software que se está midiendo’.

- Esta definición es demasiado amplia para las necesidades del método COSMIC, ya que la elección del usuario (o usuarios) está determinado por los FUR que deben medirse.



Usuarios Funcionales

DEFINICIÓN – Usuario Funcional
✓ Un tipo de usuario que es un emisor y/o un destinatario de los datos en los Requisitos Funcionales de Usuario de una pieza de software.

- En el método COSMIC es esencial distinguir a los usuarios funcionales de una pieza del software que debe ser medido de entre todos sus posibles usuarios.

Ejemplo 1

- Considere una aplicación empresarial, sus usuarios funcionales normalmente incluirían humanos y otras aplicaciones semejantes con las que la aplicación interactúa.
- Para una aplicación en tiempo real, los usuarios funcionales normalmente serían dispositivos hardware u otros software semejantes que interactúen con ella.

Reglas de Usuarios Funcionales

REGLAS – Usuarios Funcionales

a) Los usuarios funcionales de una pieza de software a ser medida, deben ser derivados del propósito de la medición

b) Para los usuarios funcionales que son funcionalmente idénticos según los FUR, se identifica un solo tipo de usuario funcional.

c) Cuando el propósito de una medición está relacionado con el esfuerzo para desarrollar o modificar el software, entonces los usuarios funcionales deben ser todos aquellos emisores y/o receptores de datos hacia/desde la nueva funcionalidad o la modificada.

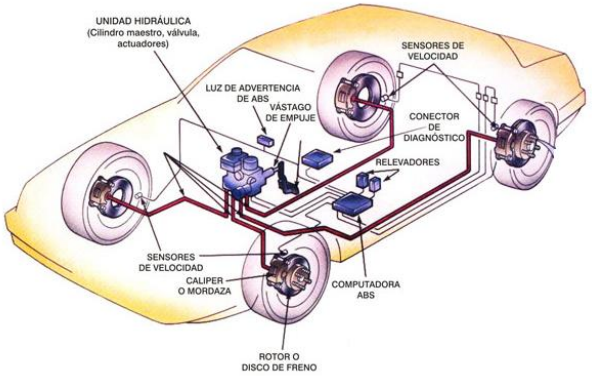
Ejemplo 2 – Aplicación de Negocio

- En un sistema de pedidos un número de empleados (usuarios funcionales) mantienen los datos del pedido.

- Se identificar un solo tipo de usuario funcional ‘empleado’.
- Ejemplo que ilustra la regla b)

Ejemplo 3 – Tiempo Real

- Cada rueda de un coche tiene un sensor que obtiene la presión de aire. Si la presión es demasiado baja o demasiado alta.
- Los valores están en el software - el software activa el aviso correspondiente LED rojo(s) en el tablero.
- Los usuarios funcionales son los cuatro sensores y los cuatro LEDs.



- ✓ Tanto los cuatro sensores y los cuatro LEDs son funcionalmente idénticos, identificar a un usuario funcional de tipo 'sensor' y de un tipo de usuario funcional ‘LED’.
- ✓ Ejemplo que ilustra la regla b)

Frontera

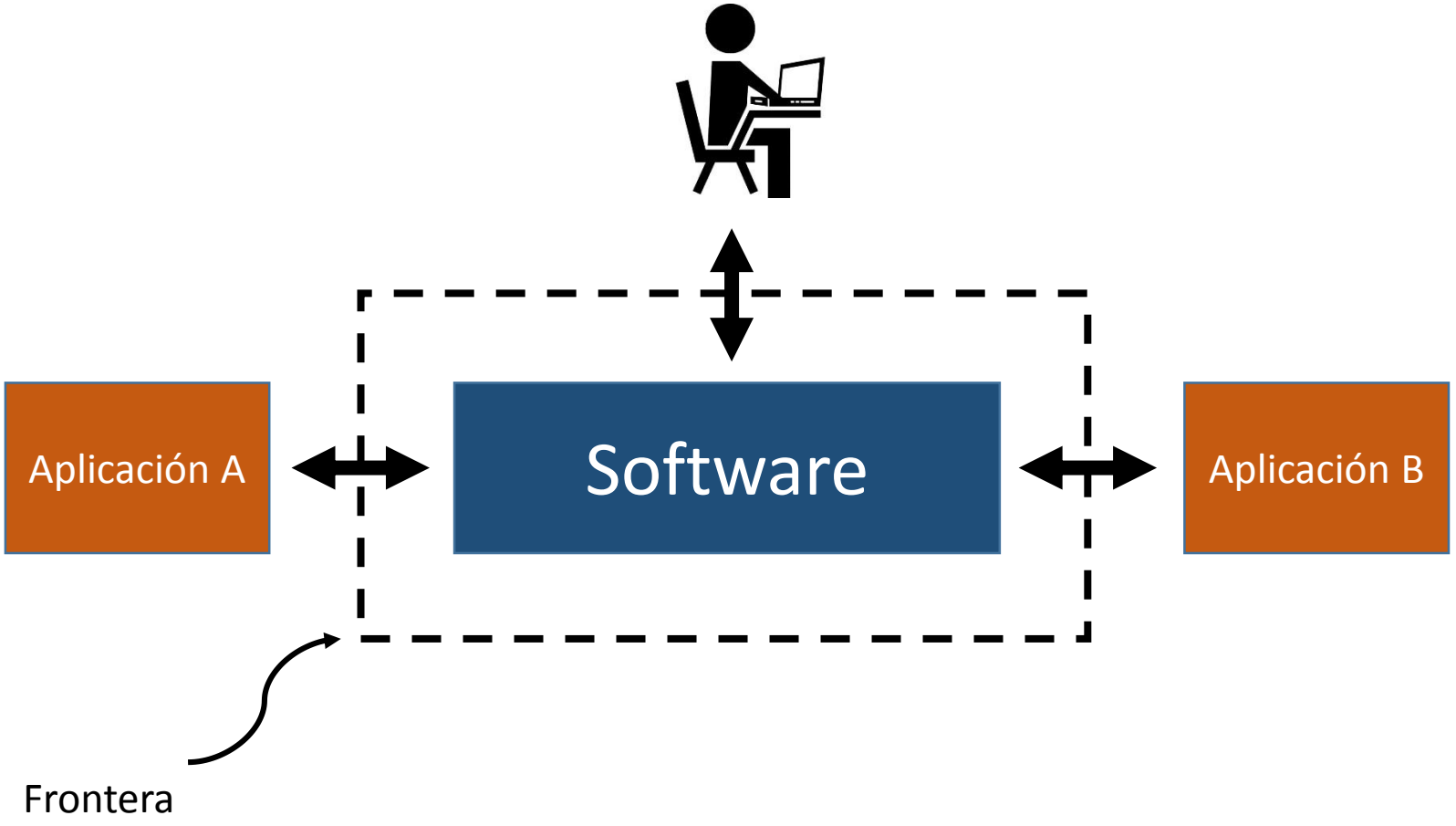
DEFINICIÓN – Frontera
✓ Una interfaz conceptual entre el software que está siendo medido y sus usuarios funcionales.

- NOTA: Se desprende de la definición que hay una frontera entre dos piezas de software en las mismas o diferentes capas que intercambian datos donde una pieza de software es un usuario funcional de la otra, y/o viceversa

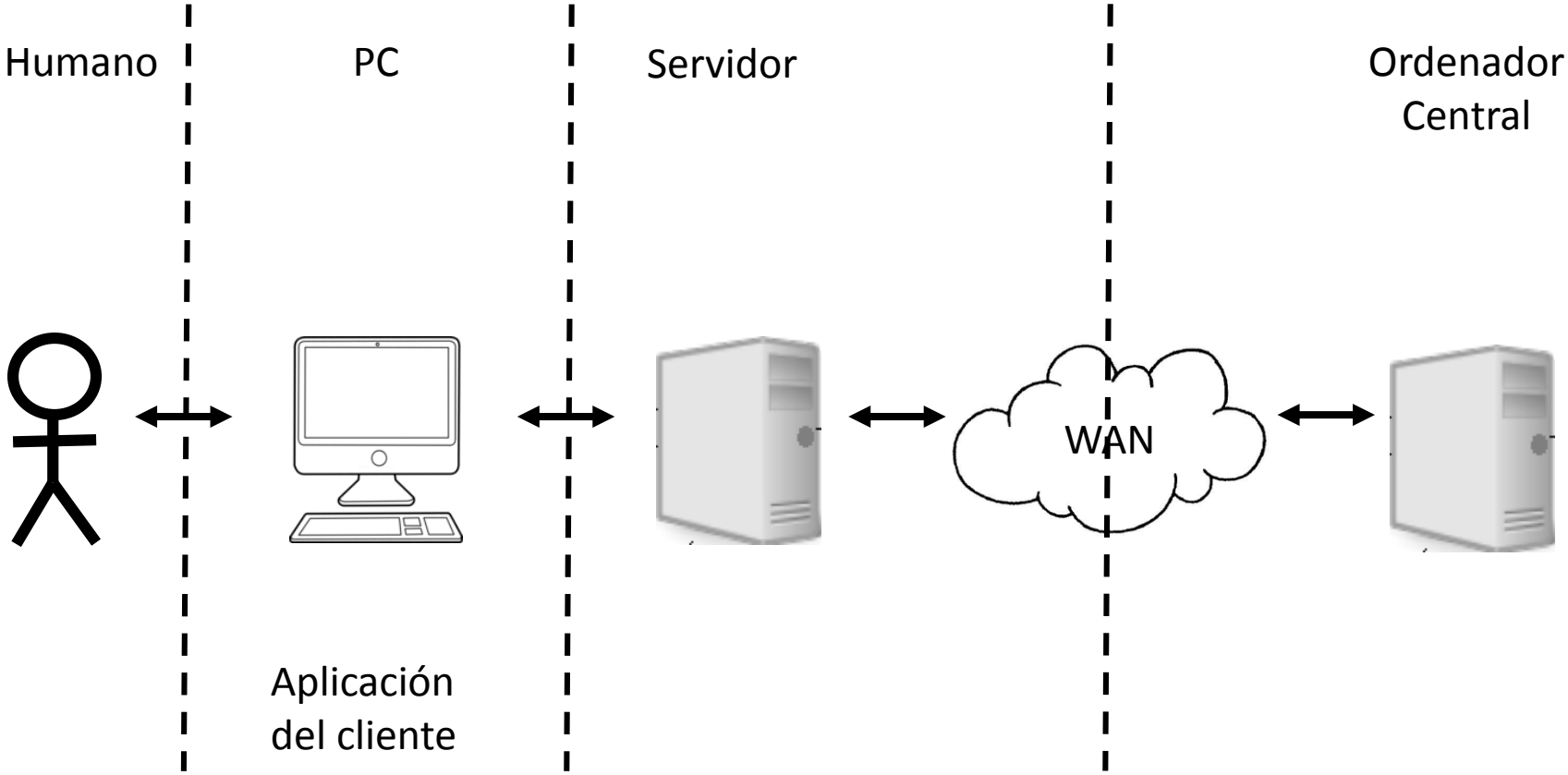
Frontera

- Esta definición de ‘frontera’ se toma de ISO / IEC 14143/1: 2007, modificado por la adición de ‘funcional’ para calificar al ‘usuario’.
- La frontera no se debe confundir con cualquier línea que podría ser dibujada alrededor de algún tipo de software que debiera medirse para definir el alcance de la medición.
- La frontera no se utiliza para definir el alcance de la medición

Frontera



Frontera en una aplicación de negocio



Almacén Persistente

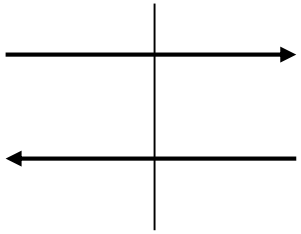
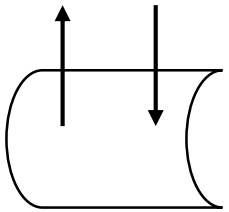
DEFINICIÓN – Almacén Persistente
✓ Es un almacén que permite a un proceso funcional almacenar un grupo de datos más allá de la vida del proceso funcional y/o del cual un proceso funcional puede recuperar un grupo de datos almacenado por otro proceso funcional, o almacenado por una ocurrencia anterior del mismo proceso funcional.

- NOTA: En el modelo COSMIC, el almacén persistente es un concepto que existe solamente dentro de la frontera del software que se está midiendo, no es considerado como un usuario funcional del software que se está midiendo.

Diagramas de contexto

- Puede ser muy útil cuando se define un alcance de medición y los usuarios funcionales dibujar un ‘diagrama de contexto’ para el software que se está midiendo.
- Se utilizan los diagramas de contexto para mostrar el alcance de una pieza de software que se va a medir dentro de su contexto de usuarios funcionales y los movimientos de datos entre ellos.
- Un diagrama de contexto es efectivamente una instancia de un patrón de medición aplicada al software que se mide. Los símbolos clave utilizados en los diagramas de contexto son:

Diagramas de contexto

Símbolo	Interpretación
	La pieza de software que se desea medir (caja con contorno continuo y grueso), es decir, la definición de un alcance de medición
	Cualquier usuario funcional del software que se desea medir
	Las flechas representan todos los movimientos de los datos que cruzan la frontera (la línea de puntos) entre un usuario funcional y el software que se mide.
	Las flechas representan todos los movimientos de datos entre el software que se esta midiendo y el 'almacén persistente'. (El símbolo de diagrama de flujo estándar para 'almacenamiento de datos' hace hincapié en que el almacenamiento persistente es un concepto abstracto. El uso de este símbolo indica que el software no interactúa directamente con el almacenamiento del hardware físico.)

Ejemplo – Aplicación de Negocio

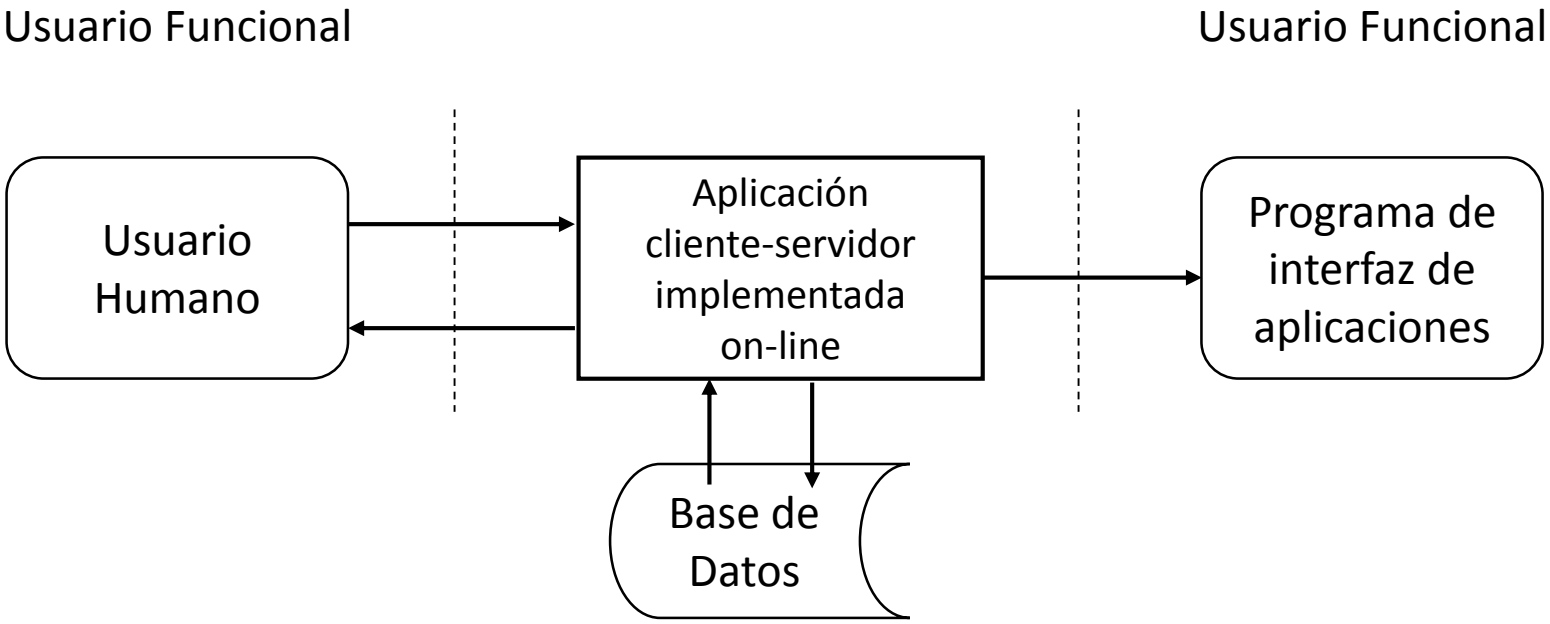


Diagrama de contexto para una aplicación cliente-servidor

Ejemplo – Aplicación en tiempo real

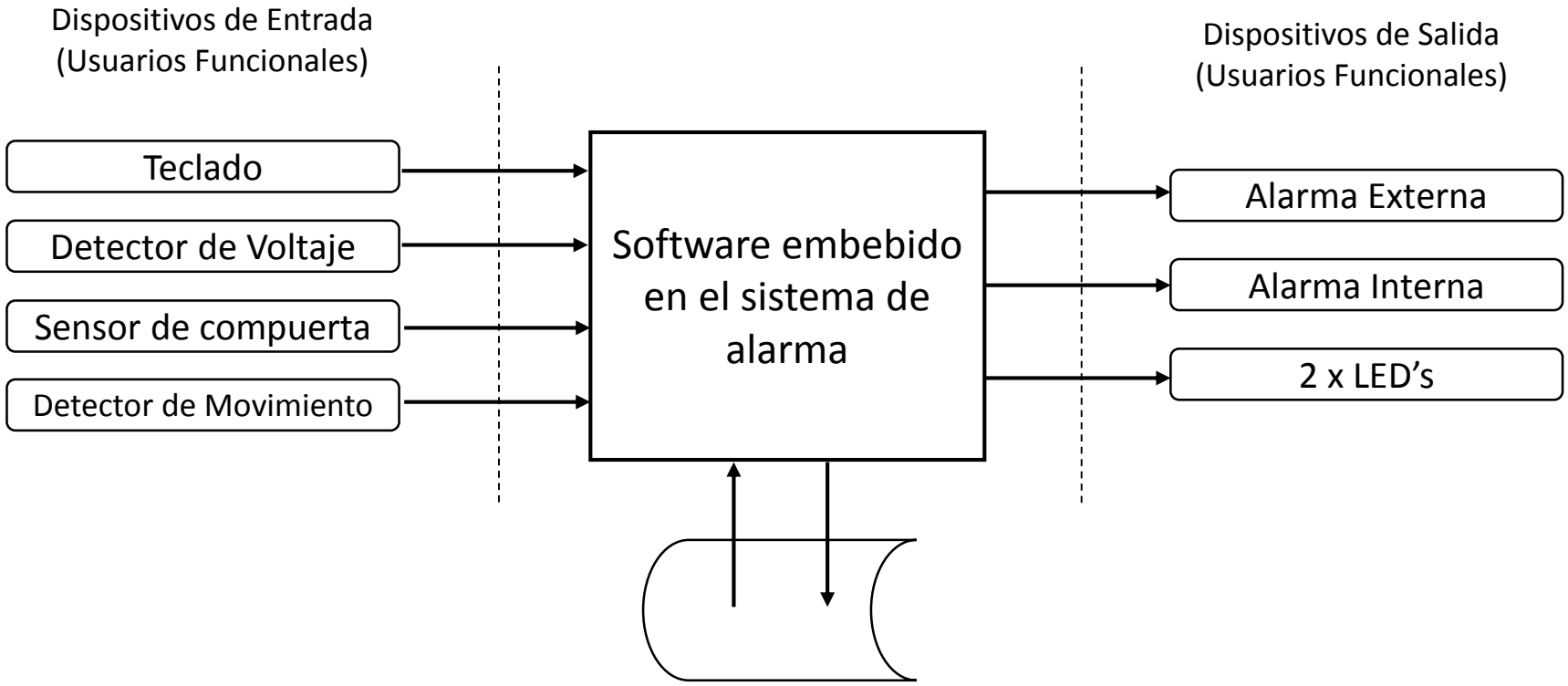
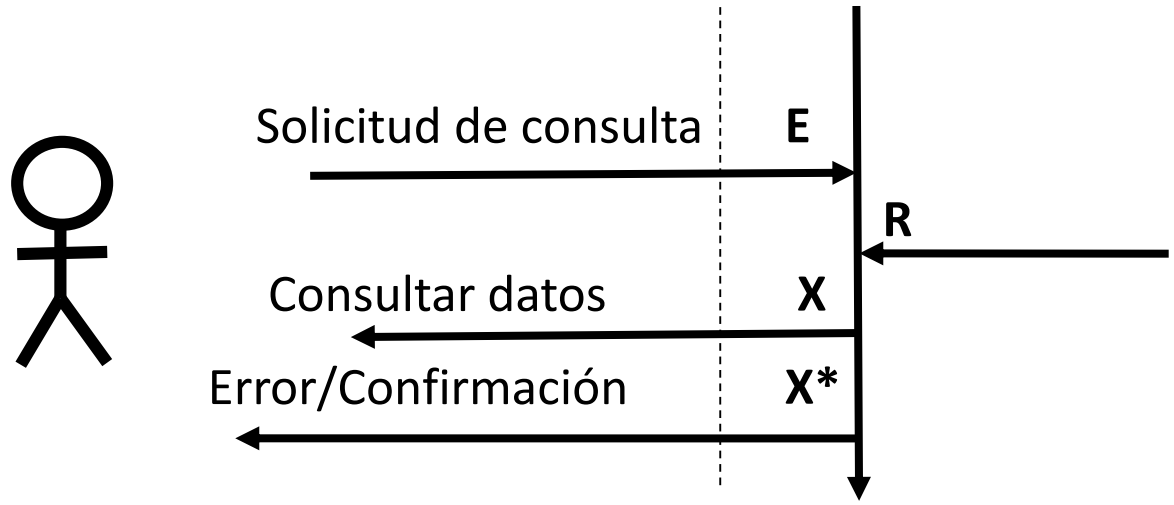


Diagrama de Contexto para el software embebido de un sistema de alarma de intrusos

Diagrama de Secuencia de Mensajes

Software A en el ejemplo de capa de aplicación



* Si lo requieren los FUR

Fase de estrategia de medición

- Fase de estrategia de medición
  - Introducción
  - Definir el propósito de la medición
  - Definir el alcance de la medición
  - Identificando usuarios funcionales y el almacenamiento persistente
  - Identificando el nivel de granularidad
  - Observaciones finales

Identificando el nivel de granularidad

- En las etapas iniciales de un proyecto de desarrollo de software, los requisitos funcionales de los usuarios (FUR) se especifican en ‘alto nivel’, es decir, se obtiene un esbozo, o con pocos detalles.
- A medida que el proyecto progresa, los FUR, son refinados, revelando más detalle a un ‘menor nivel’. Estos diferentes grados de detalle de los requisitos actuales son conocidos como diferentes ‘niveles de granularidad’.

Nivel de granularidad

DEFINICIÓN – Nivel de granularidad
✓ Cualquier nivel de expansión de la descripción de una pieza de software de tal manera que a cada aumento del nivel de expansión, la descripción de la funcionalidad de la pieza de software está en un nivel de detalle mayor y uniforme.

- NOTA: Los medidores deben ser conscientes de que cuando los requisitos van apareciendo a principios de la vida de un proyecto de software, en cualquier momento diferentes partes de la funcionalidad necesaria del software normalmente se habrán documentado en los diferentes niveles de granularidad.

Analogía

Un conjunto de mapas de carreteras revela los detalles de una red nacional de carreteras en tres niveles de granularidad:

- Mapa A muestra sólo las autopistas y carreteras principales.
- Mapa B muestra todas las autopistas, carreteras principales y secundarias.
- Mapa C muestra todas las carreteras secundarias con sus nombres.

- Si no reconocemos el fenómeno de los diferentes niveles de granularidad, parece que se exponen tres mapas diferentes, con mapas de carreteras todos reconocemos los diferentes niveles de detalle y hay escalas estándar para interpretar el tamaño de la red.

Nivel de granularidad

- El concepto abstracto de nivel de granularidad se esconde detrás de las escalas de los FUR utilizados para realizar la medición.
- Para la medición del software, sólo hay un nivel de granularidad estándar que es posible definir sin ambigüedades. Es el nivel de granularidad en el cual se han identificado los distintos procesos funcionales y se han definido sus movimientos de datos. Las mediciones deben hacerse a este nivel o a escala de este nivel siempre que sea posible.

Nivel de granularidad

Detallar los FUR implica la descripción del software de un nivel de granularidad ‘superior’ a uno ‘inferior’ revelando más detalles pero *sin modificar su alcance*.

No confundir con:

- Detallar un software con el fin de revelar sus componentes, subcomponentes, etc.
- La evolución de la descripción de algún tipo de software a medida que avanza a través de su ciclo de desarrollo,

Nivel de granularidad

- El concepto de ‘nivel de granularidad’ se aplica únicamente a los requisitos de los usuarios funcionales de software (FUR).

- Mediciones de tamaño funcional COSMIC exactos requieren que el FUR a ser medido existe a un nivel de granularidad en el que los procesos funcionales y sus movimientos de datos pueden ser identificados.

Nivel de granularidad de un proceso funcional

DEFINICIÓN – Nivel de granularidad de un Proceso Funcional

- ✓ Un nivel de granularidad de la descripción de una pieza de software en el que los usuarios funcionales:
  - Son seres humanos individuales o dispositivos de ingeniería o elementos de software (y no grupos de estos)
  - Detectan ocurrencias únicas de eventos a los que la aplicación software debe responder (y no cualquier nivel en el cual se han definido grupos de eventos)

Nivel de granularidad de un proceso funcional

- NOTA 1: En la práctica, la documentación de software y por lo tanto los FUR a menudo describe la funcionalidad en diferentes niveles de granularidad.
- NOTA 2: ‘Grupos de estos’ podrían ser, un ‘departamento’ cuyos miembros manejan muchos tipos de procesos funcionales.
- NOTA 3: Un grupo de eventos puede indicarse en una especificación de FUR en un alto nivel de granularidad por un flujo de entrada a un sistema de software de aeronáutica, etiquetado como ‘comandos del piloto’.

Reglas de Nivel de granularidad de un proceso funcional

REGLAS – Nivel de granularidad de un proceso funcional

a) Una medición precisa de tamaño funcional de una pieza de software requiere que sus FUR sean conocidos a nivel de granularidad en la que se pueden identificar sus procesos funcionales y sub-procesos de movimiento de datos.

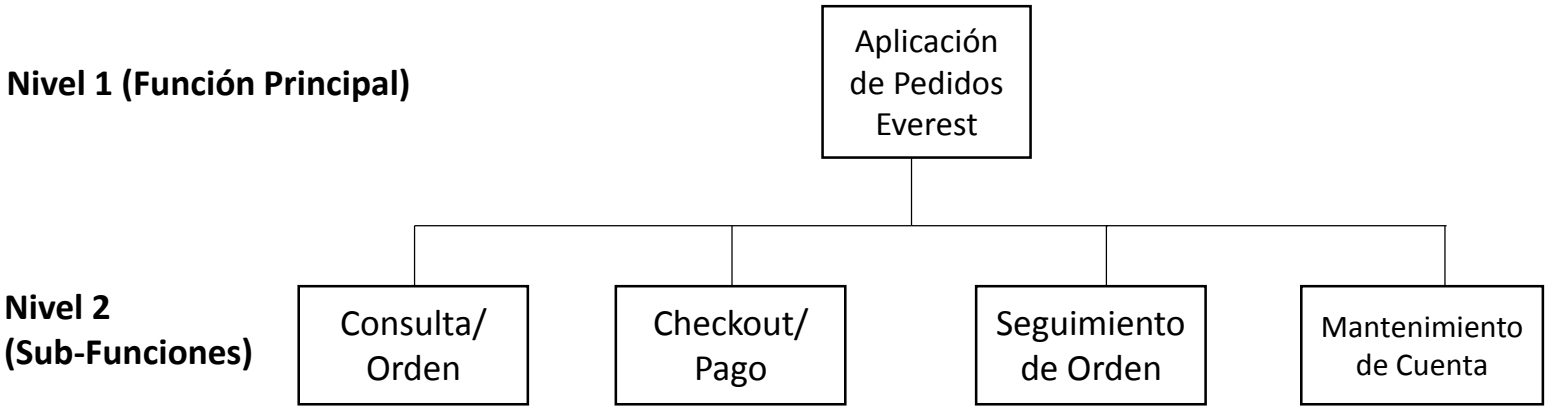
b) Si algunos requisitos deben medirse antes de que se hayan definido con suficiente detalle para una medición precisa, los requisitos se pueden medir utilizando un enfoque de aproximación. Estos enfoques definen cómo los requisitos pueden ser medidos en niveles de granularidad superiores.

Ejemplo – Aplicación de Negocio

Se tiene un conocido sistema de comprar de productos a través de Internet, que llamaremos el sistema ‘Aplicación de Pedidos Everest’.

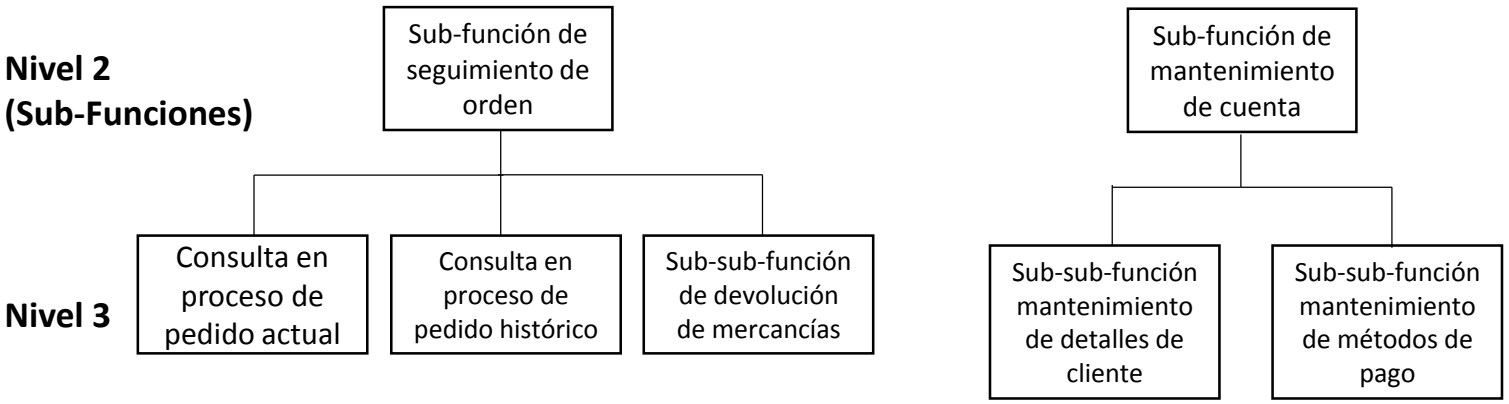
- Si quisiéramos medir esta aplicación, podemos asumir el propósito de la medición es determinar el tamaño funcional de la parte de la aplicación disponible para los clientes
- Tendríamos que definir el alcance de la medida como ‘las partes de la aplicación del Everest accesibles a los clientes para encargar productos a través de Internet".

Ejemplo – Aplicación de Negocio



Análisis del sistema de pedidos Everest: los dos primeros niveles de granularidad

Ejemplo – Aplicación de Negocio



La descomposición del sub-sistema del Seguimiento y del sub-sistema del Mantenimiento de la cuenta

Fase de estrategia de medición

- Fase de estrategia de medición
  - Introducción
  - Definir el propósito de la medición
  - Definir el alcance de la medición
  - Identificando usuarios funcionales y el almacenamiento persistente
  - Identificando el nivel de granularidad
  - Observaciones finales

Observaciones finales

- Es esencial para determinar y documentar los parámetros de la estrategia de medición a fin de garantizar que el tamaño resultante puede entenderse y utilizarse en el futuro correctamente
- La gran mayoría de las mediciones de tamaño funcionales se llevan a cabo para un propósito que está relacionado con el esfuerzo de desarrollo de alguna manera, por ejemplo, para la medición de los resultados de desempeño del proyecto, o para la estimación del proyecto.

Facultad de Ciencias

Contacto

- Jesús Iván Saavedra Martínez
- jism@ciencias.unam.mx

Métricas de Software

Fase de Representación

Jesús Iván Saavedra Martínez

Octubre 2017

COSMIC

- Introducción
- Fase de estrategia de medición
- Fase de representación
- Fase de medición
- Informe de medidas

Fase de Representación

- Fase de Representación
  - Introducción
  - Identificación de los procesos funcionales
  - Identificación de objetos de interés y grupos de datos
  - Identificación de los atributos de datos
  - Identificando los movimientos de datos

Introducción

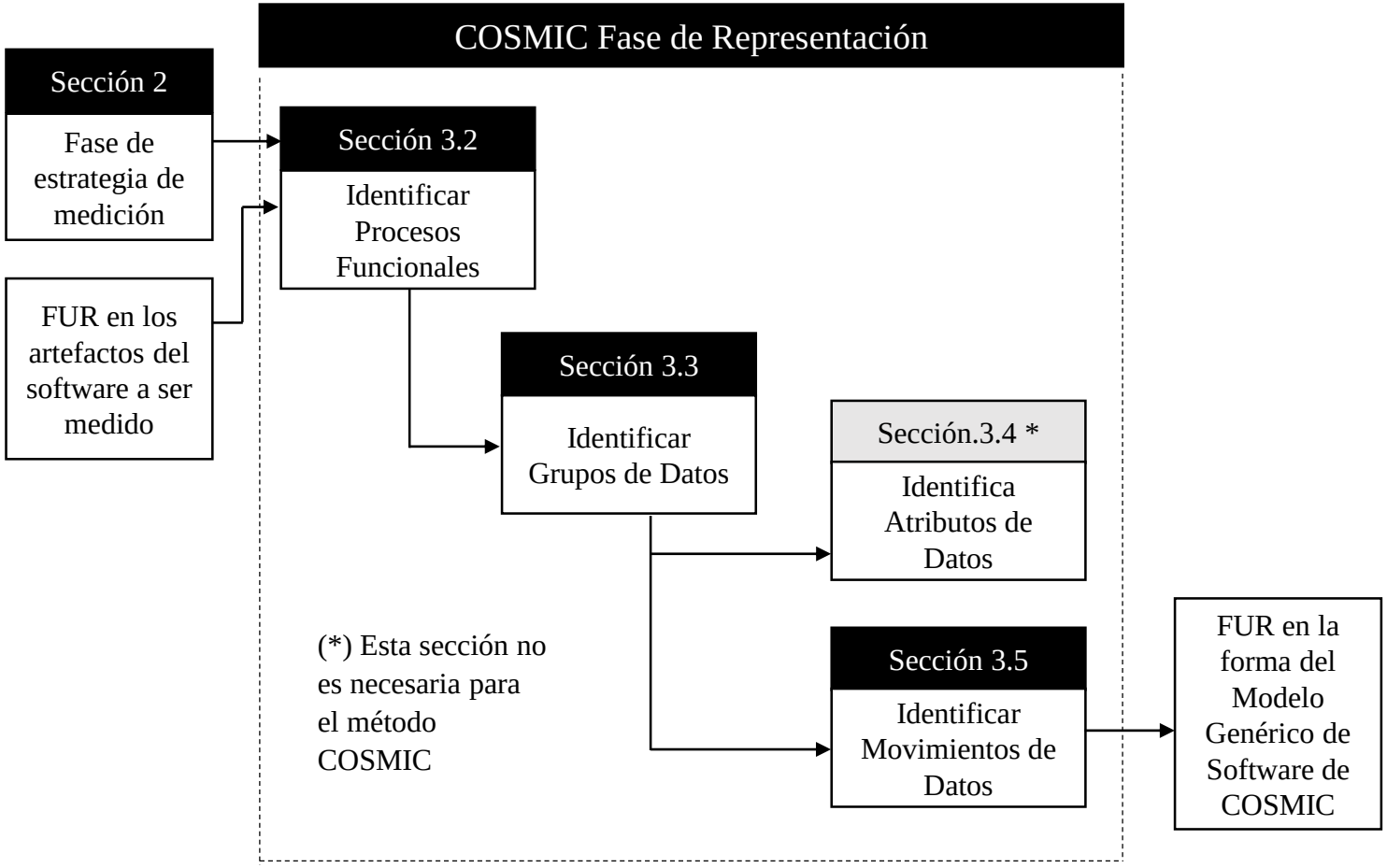
- La fase de representación trata de la medición mediante la definición de los conceptos clave del Modelo Genérico de Software y el proceso a seguir en la representación de los FUR, con la finalidad de que los FUR se puedan medir.

- *Evento desencadenante y proceso funcional*
- *Movimientos de datos o manipulación de datos*
- *Grupo de datos, atributos de datos y objeto de interés*

Introducción

- Un evento hace que un *usuario funcional* solicite un servicio de la pieza de software que se está midiendo.
- Los procesos funcionales se componen de dos tipos de subprocesos que, o bien mueven datos ('*movimientos de datos*') o manipulan datos ('*manipulación de datos*').
- Un movimiento de datos mueve un '*grupo de datos*'. Un grupo de datos se compone de '*atributos de datos*' que todos describen un '*objeto de interés*'.
- Hay cuatro tipos de movimientos de datos: *Entradas, Salidas, Lecturas* y *Escrituras*

Representación de los FUR al Modelo Genérico de Software



Método general del proceso de representación COSMIC

Fase de Representación

- Fase de Representación
  - Introducción
  - Identificación de los procesos funcionales
  - Identificación de objetos de interés y grupos de datos
  - Identificación de los atributos de datos
  - Identificando los movimientos de datos

Evento desencadenante

DEFINICIÓN – Evento
✓ Algo que sucede

DEFINICIÓN – Evento desencadenante
✓Un evento que genera que un usuario funcional del componente de software desencadene (“trigger”) uno o más procesos funcionales. En un conjunto de requisitos funcionales de usuario, cada evento que causa que un usuario funcional desencadene un proceso funcional

Evento desencadenante

- Un evento que hace que uno o más usuarios funcionales de genere uno o más grupos de datos, cada uno de los cuales posteriormente serán movidos por una Entrada desencadenante.

- Un evento desencadenante no puede ser subdividido y sucede o no sucede.
- NOTA: los eventos de reloj y temporización pueden ser eventos desencadenantes.

Proceso funcional

DEFINICIÓN – Proceso funcional
✓ Un conjunto de movimientos de datos, que representa una parte elemental de los Requisitos Funcionales de Usuario para el software que se está midiendo, que es único dentro de estos FUR y que se puede ser definido de manera independiente de cualquier otro proceso funcional en estos FUR.

- El conjunto de todos los movimientos de datos de un proceso funcional es el conjunto que se necesita para cumplir con los FUR para todas las posibles respuestas a la Entrada desencadenante.

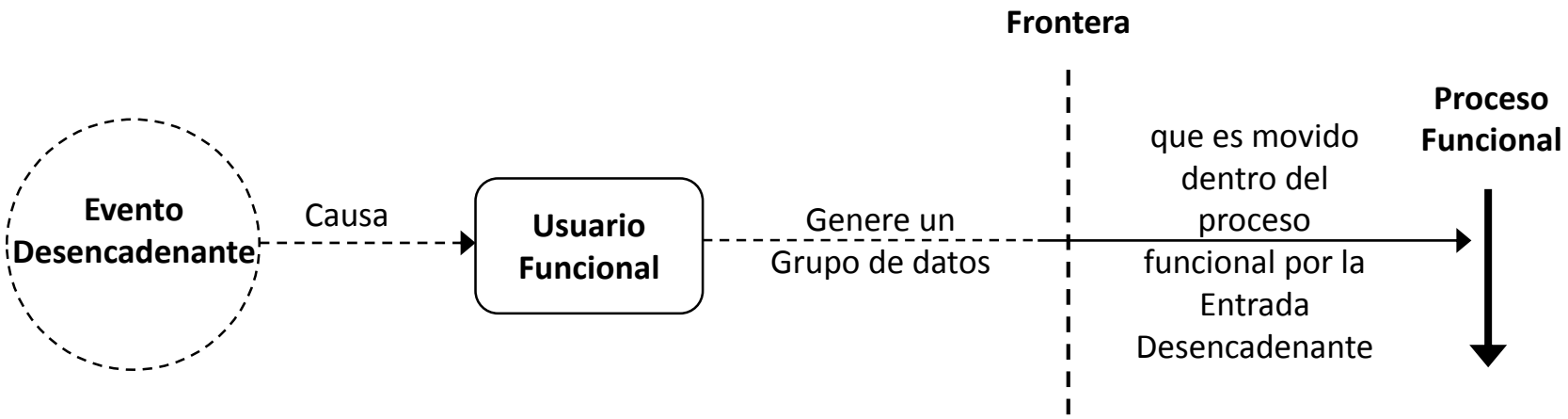
Proceso funcional

- Un proceso funcional puede tener sólo una Entrada desencadenante.
- Cada proceso funcional inicia con la recepción de un grupo de datos movidos de una Entrada desencadenante.
- El FUR para un proceso funcional puede requerir una o más Entradas distintas, además de la entrada desencadenante.
- Si un usuario funcional envía un grupo de datos con errores, por lo general es la tarea del proceso funcional para determinar si el evento realmente ocurrió y/o si los datos introducidos son realmente válidos y cómo responder.

Entrada desencadenante

DEFINICIÓN – Entrada desencadenante
✓ El movimiento de Entrada de datos de un proceso funcional que mueve un grupo de datos generados por un usuario funcional que el proceso funcional necesita para iniciar el procesamiento.

Relaciones



- Todas las relaciones pueden ser uno a muchos, muchos a uno, o muchos a muchos, con una excepción, que el grupo de datos movido por cualquier Entrada desencadenante puede iniciar sólo un proceso funcional.

Ejemplos de cardinalidades

1. Un evento desencadenante puede ser detectado por muchos usuarios funcionales, por ejemplo, un terremoto es detectado por muchos sensores.
2. Una aplicación de software A que es un usuario funcional de otra aplicación B, puede 'llamar' la aplicación B para diversos propósitos. Le corresponde a la aplicación A la generación de un evento desencadenante separado para cada llamada.
3. Algunos procesos funcionales pueden ser iniciados por diferentes usuarios funcionales, por ejemplo, un componente de software puede ser llamado por cualquiera de sus usuarios funcionales.

Ejemplos

- *Aplicación de negocio:*
  - Un proceso funcional de un sistema de software de personal puede ser iniciado por la entrada desencadenante que mueve un grupo de datos que describe un nuevo empleado. El grupo de datos es generado por un usuario funcional humano del software que introduce los datos.
- *Tiempo real:*
  - Un proceso funcional puede ser iniciado por su Entrada desencadenante informando al proceso funcional que un reloj (usuario funcional) ha marcado. El grupo de datos que se movió contiene datos (la marca de tiempo) que informa únicamente de que ha ocurrido un evento.

Proceso de identificación de los procesos funcionales

1. Identificar en el mundo de los usuarios funcionales los eventos por separado a los que el software que se está midiendo debe responder- los ‘eventos desencadenantes’.
2. Identificar qué usuario(s) funcional(es) del software puede responder a cada evento desencadenante.
3. Identifique la entrada (o entradas) desencadenante(s) que cada usuario funcional puede iniciar en respuesta al evento.
4. Identificar el proceso funcional iniciado por cada entrada desencadenante.

Reglas para comprobar los procesos funcionales candidatos

a) Un proceso funcional pertenecerá íntegramente al alcance de la medición de una pieza de software en una, y sólo una, capa.

b) Cualquiera Entrada desencadenante de una pieza de software puede iniciar sólo un proceso funcional en ese software.

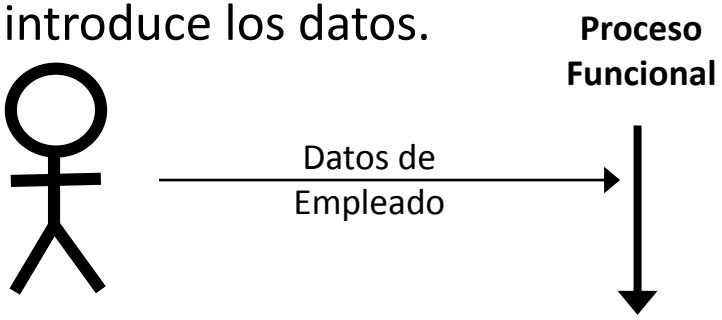
c) Un proceso funcional comprenderá al menos dos movimientos de datos, una Entrada y además una Salida o una Escritura. No hay límite superior para el número de movimientos de datos en un proceso funcional.

d) Un proceso funcional en ejecución se debe considerar terminado cuando se ha satisfecho su FUR para la respuesta a su Entrada desencadenante. Una pausa durante la tramitación no se considerará como la terminación del proceso funcional.

Eventos disparadores y procesos funcionales - aplicaciones de negocio

a) Eventos desencadenantes de una aplicación de negocio on-line normalmente ocurren en el mundo de los usuarios funcionales. El usuario humano comunica la incidencia de un evento a un proceso funcional introduciendo datos sobre el evento.

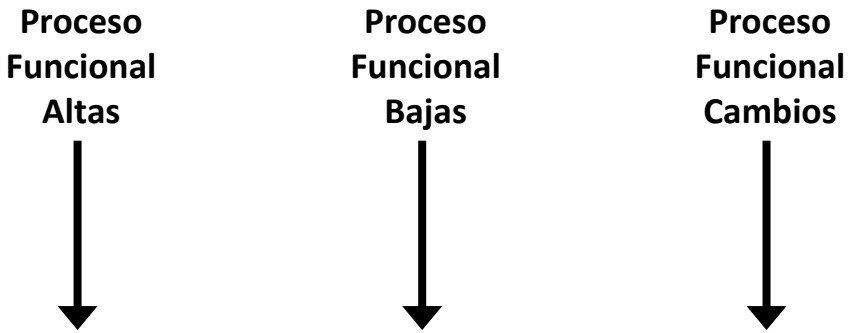
- Ejemplo:
  - Un proceso funcional de un sistema de software de personal puede ser iniciado por la entrada desencadenante que mueve un grupo de datos que describe un nuevo empleado.
  - ✓ El grupo de datos es generado por un usuario funcional humano del software que introduce los datos.



Eventos disparadores y procesos funcionales - aplicaciones de negocio

b) Eventos desencadenantes separados y por lo tanto procesos funcionales separados deben distinguirse en los siguientes casos:

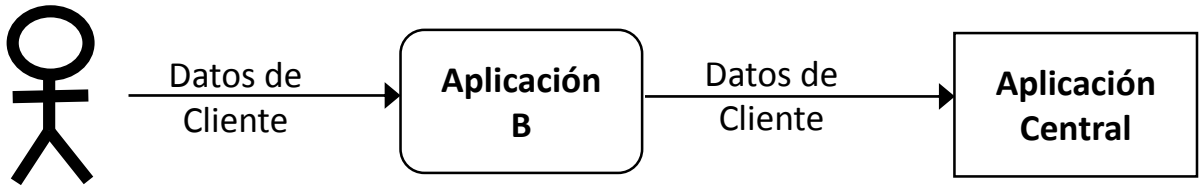
- Cuando un usuario funcional humano toma decisiones fuera del software sobre 'qué hacer después' que son independientes en el tiempo y requieren respuestas separadas del software, cada decisión por separado es un evento desencadenante para el cual el software debe proporcionar un proceso funcional distinto.
- Cuando las responsabilidades para las actividades son separadas.



Eventos disparadores y procesos funcionales - aplicaciones de negocio

c) Para una aplicación A podría haber una aplicación semejante B que necesite enviar u obtener datos de la aplicación A. En este caso, la aplicación B desencadena un proceso funcional de la aplicación A cuando necesita enviar u obtener datos, entonces la aplicación B es un usuario funcional de la aplicación A.

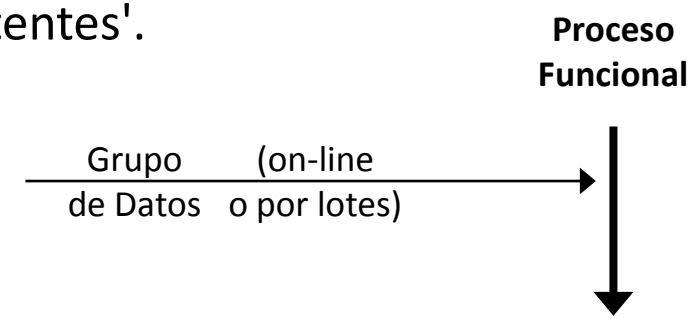
- Ejemplo:
  - Se requiere que se envíen detalles del cliente a una aplicación central de registro de clientes, lo cual se está midiendo.
  - ✓ Ahora la aplicación que procesa la orden se ha convertido en un usuario funcional de la aplicación central.



Eventos disparadores y procesos funcionales - aplicaciones de negocio

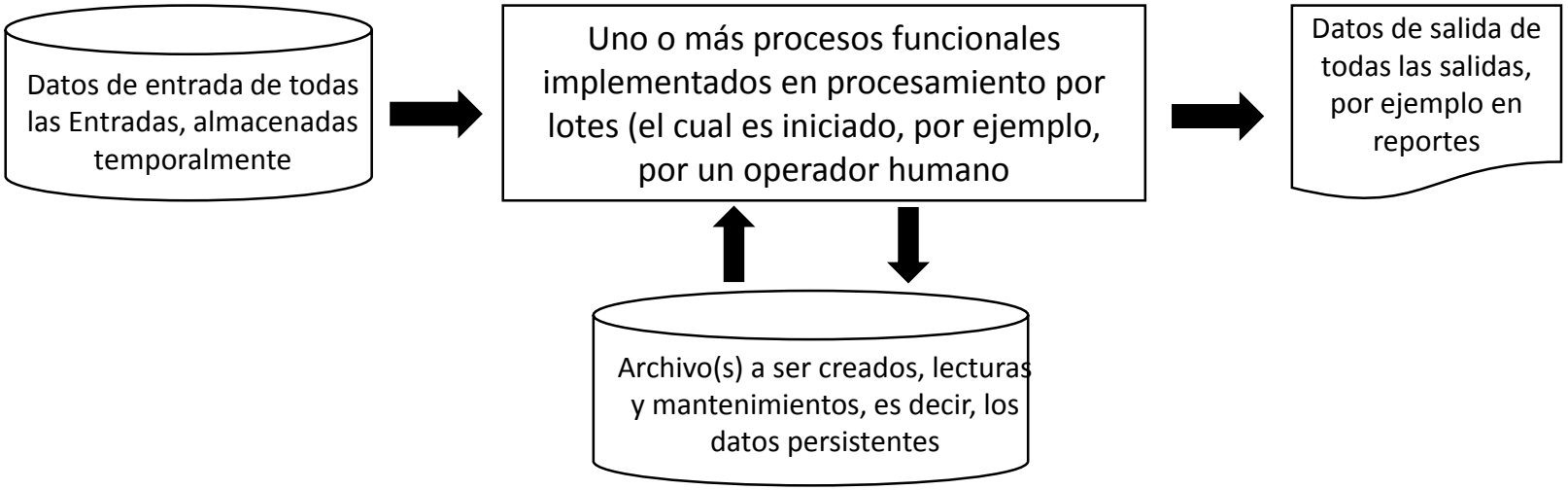
d) No hay diferencia en el principio para el análisis de un proceso funcional si se requiere un procesamiento on-line o un procesamiento por lotes.

- Todos los datos que han sido introducidos como entrada para el procesamiento por lotes deben ser almacenados temporalmente en algún lugar antes de que el proceso puede comenzar.
- Los datos de entrada almacenados temporalmente, deben ser analizados en la misma forma que si se está ingresando directamente a la aplicación. Estos datos de entrada no es 'de datos persistentes'.



Eventos disparadores y procesos funcionales - aplicaciones de negocio

Un ‘trabajo’ por procesamiento por lotes, implementación de un conjunto de procesos funcionales

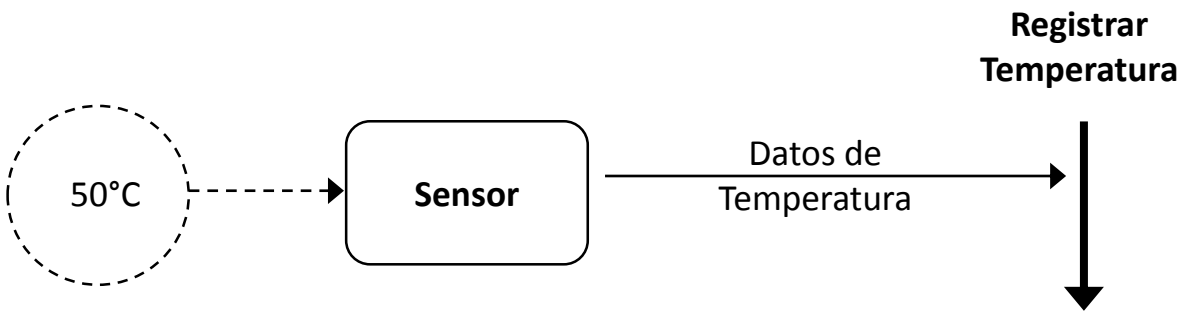


NOTA: El requisito de que algunos datos de entrada sean procesados por lotes es un requisito no funcional (NFR).

Eventos disparadores y procesos funcionales - aplicaciones de tiempo real

a) Un evento desencadenante es detectado típicamente por un sensor.

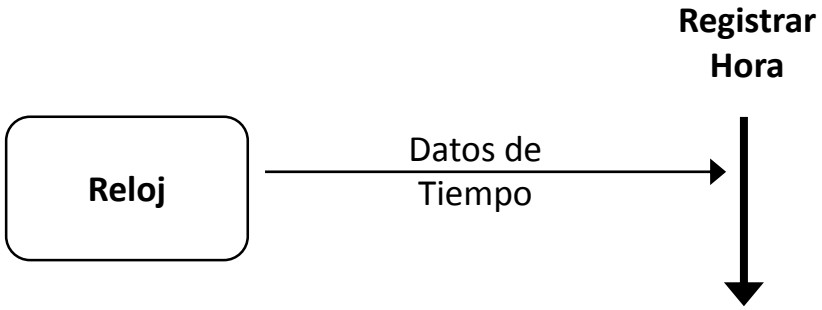
- Ejemplo:
  - Cuando un sensor (usuario funcional) detecta que la temperatura alcanza un valor determinado (evento desencadenante), se produce que el sensor (usuario funcional) envíe una señal iniciando un movimiento de datos de Entrada desencadenante de un proceso funcional para apagar un calefactor.



Eventos disparadores y procesos funcionales - aplicaciones de tiempo real

b) Señales periódicas de una reloj (marca de tiempo (ticks) de reloj) pueden desencadenar un proceso funcional.

- Ejemplo:
  - Un proceso funcional puede ser iniciado por su Entrada desencadenante informando al proceso funcional que un reloj (usuario funcional) ha marcado. El grupo de datos que se movió contiene datos (la marca de tiempo) que informa únicamente de que ha ocurrido un evento.



Procesos Funcionales Separados

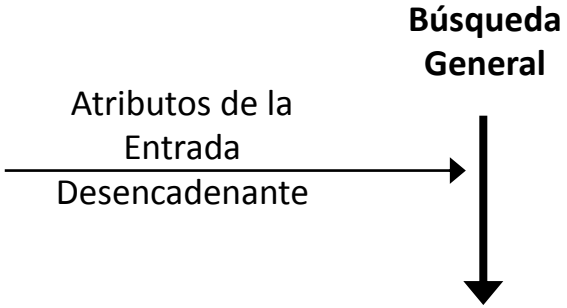
- El software diferencia eventos y proporciona los correspondientes procesos funcionales dependiendo solo de sus FUR.
  - Al medir el software puede a veces es difícil decidir qué eventos separados se requiere que el software reconozca.
  - El proceso funcional debe ser capaz de hacer frente a todas las posibles ocurrencias de valores movido por su entrada desencadenante, incluyendo valores de datos válidos y no válidos, e incluso valores de datos faltantes.

Ejemplo 1

- *Aplicación de negocios:*
  - Hay un requisito para dos tipos de créditos, ‘crédito adicional’ y ‘crédito fiscal de trabajo’.
  - Estos son requisitos a los que el software debe responder como dos eventos separados en el mundo de los usuarios funcionales humanos.
  - Por tanto, debería haber dos procesos funcionales, aunque solo se haya utilizado un formulario de impuestos para recoger los datos en ambos casos.

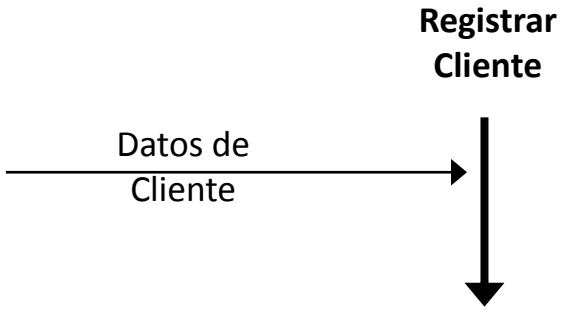
Ejemplo 2

- *Aplicación de negocios:*
  - Un proceso funcional que ofrece una capacidad de búsqueda general contra una base de datos puede ser obligado a aceptar hasta cuatro parámetros de búsqueda.
  - Pero el mismo proceso funcional funcionará si se introducen los valores de sólo uno, dos, tres o cuatro parámetros de búsqueda.



Ejemplo 3

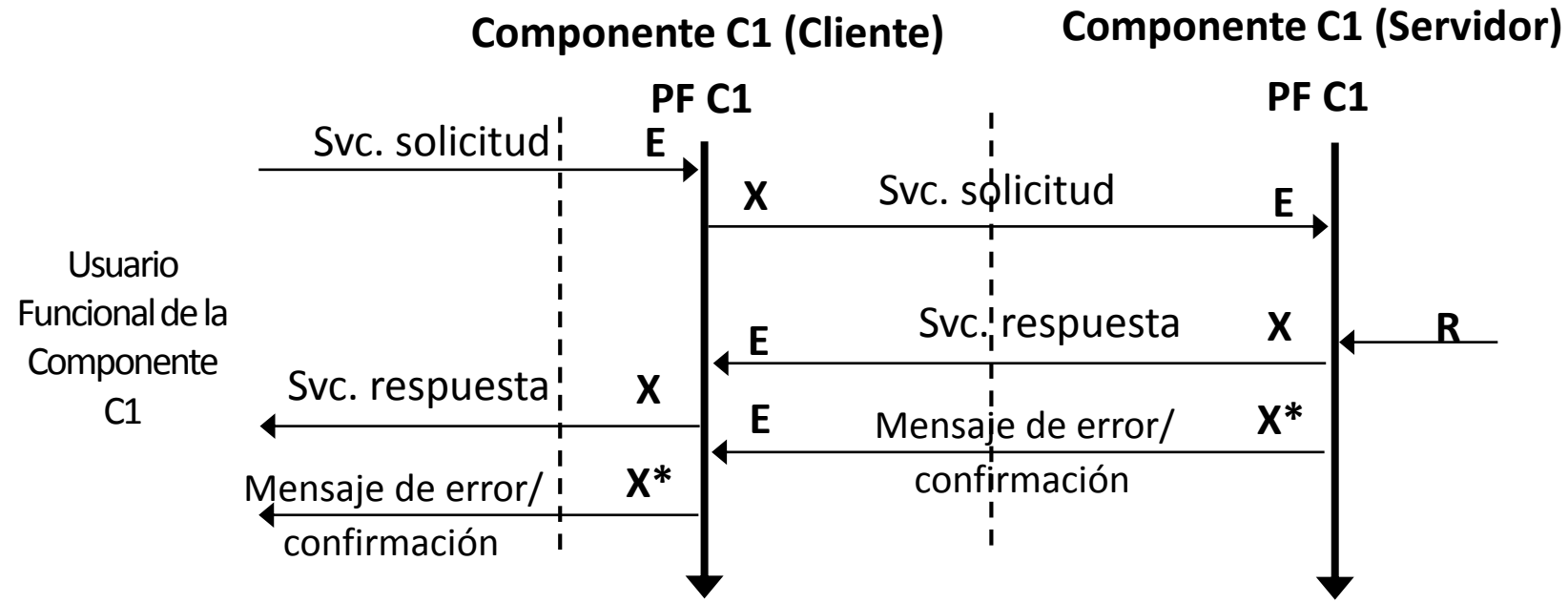
- *Aplicación de negocios:*
  - Para un proceso funcional que registra un nuevo cliente de una empresa, es obligatorio introducir los datos para la mayoría de los atributos de datos, pero algunos son opcionales y puede dejarse en blanco.
  - Independientemente de si se introducen todos o un subconjunto de estos atributos sólo hay un proceso funcional para registrar un nuevo cliente.



La medición de los componentes de un sistema de software distribuido

- Cuando el propósito de una medición es medir por separado el tamaño de cada componente de un sistema de software distribuido, un alcance de la medición independiente debe definirse para cada componente.
  - En tal caso, el dimensionamiento de los procesos funcionales de cada componente sigue todas las reglas normales como ya se ha descrito.
  - El usuario funcional de cada componente se determina examinando donde ocurren los eventos que hacen que se desencadenen los procesos funcionales en el componente que está siendo examinado.

La medición de los componentes de un sistema de software distribuido



*Si es necesario

Los intercambios de datos entre los componentes cliente y servidor

Similitud funcional

DEFINICIÓN – Similitud Funcional
✓ Dos o más procesos funcionales en el mismo software que se está midiendo puede tener alguna funcionalidad que es idéntica o muy similar en cada proceso. Este fenómeno se conoce como ‘funcionalidad coincidente’, o ‘similitud funcional’.

- Cualquier funcionalidad requerida que es común o similares a través de dos o más procesos funcionales en el mismo software que se está midiendo debe tenerse en cuenta en cada uno de estos procesos funcionales.

Ejemplos

- Aplicación de negocio:
  - Varios procesos funcionales en el mismo software pueden necesitar la misma funcionalidad de validación, por ejemplo ‘fecha de la orden’, o puede tener que acceder a los mismos datos persistentes, o puede que tenga que realizar el mismo cálculo de intereses.
- *Tiempo real:*
  - En varios procesos funcionales del mismo software puede ser necesario obtener datos del mismo sensor (movimiento común de un mismo grupo de datos) o pueden necesitar la misma escala para llevar a cabo el cálculo de una conversión, por ejemplo, de Fahrenheit a Centígrados (manipulación de datos común)

Eventos que desencadenan la ejecución de un sistema de software

- Al medir el tamaño de una pieza de software, se identifican sólo los eventos y las entradas que desencadenan los procesos funcionales
  - La funcionalidad necesaria para la puesta en marcha del software en sí mismo no es parte de estos procesos funcionales y deben ser ignorados (o medirse por separado, si es necesario).
- Ejemplo:
  - Para un sistema operativo de la computadora, el usuario funcional que inicia el sistema operativo es un programa de arranque que se inicia cuando la alimentación del ordenador está encendido.

Tips de verificación

Revisar que cada proceso funcional:

1) Tenga al menos una Entrada (E)
2) Tenga al menos una Salida (X) o una Escritura (W)
3) No contenga subprocesos duplicados

- Una vez identificados los procesos funcionales, el siguiente objetivo principal debe ser la identificación de sus movimientos de datos.

Fase de Representación

- Fase de Representación
  - Introducción
  - Identificación de los procesos funcionales
  - Identificación de objetos de interés y grupos de datos
  - Identificación de los atributos de datos
  - Identificando los movimientos de datos

Objeto de interés

DEFINICIÓN – Objeto de Interés
✓ Cualquier ‘cosa’ en el mundo del usuario funcional que se identifica en los Requisitos Funcionales de Usuario, sobre la que se requiere que el software procese y/o almacene datos. Puede ser cualquier cosa física, así como cualquier objeto conceptual o parte de un objeto conceptual.

- Nota 1: ‘procesar’ puede significar cualquier operación para mover y/o manipular los datos.
- Nota 2: El término 'objeto de interés' se utiliza para evitar los términos relacionados con métodos de ingeniería de software específicos, no implica ‘objetos’ en el sentido utilizado en métodos orientados a objetos.

Grupo de datos

DEFINICIÓN – Grupo de datos
✓ Un grupo de datos es un conjunto único, no vacío, no ordenado y no redundante de atributos de datos donde cada atributo de datos incluido describe un aspecto complementario del mismo objeto de interés.

- Cada grupo de datos identificado deberá ser único y distinguible a través de su colección única de atributos de datos.

Materialización de un grupo de datos

a) Como una estructura física grabada en un dispositivo de almacenamiento continuo (archivo, base de datos, memoria ROM, etc.)

b) Como una estructura física en la memoria volátil del ordenador (estructura de datos asignada dinámicamente o a través un bloque pre-asignado de espacio de memoria)

c) Como la presentación agrupada de atributos de datos relacionados funcionalmente en un dispositivo de entrada/salida (monitor, informe impreso, etc.)

d) Como un mensaje transmitido entre un dispositivo y un ordenador, o sobre una red, etc.

Identificación de objetos de interés y grupos de datos

- La necesidad de analizar un grupo de atributos de datos *es de crítica importancia,* decidir si todos los atributos comunican datos sobre un único 'objeto de interés', determina el número de 'grupos de datos' separados.
- Ejemplo:
  - Si los atributos de datos que van a ser los datos de entrada de un proceso funcional son atributos de tres objetos de interés separados, entonces necesitamos identificar tres movimientos de datos de Entradas separadas.

Ejemplos

- *Aplicación de negocio:*
  - Un objeto de interés podría ser ‘empleado’ (físico).
    - Normalmente desde el FUR de empleados se identifican dos objetos de interés: ‘empleado’ y ’dirección del empleado’.
    - Los grupos de datos correspondientes podrían llamarse ‘datos de empleado’ y ‘ dirección del empleado’.
- *Tiempo real:*
  - Un software de intercambio de mensajes puede recibir un grupo de datos de mensajes como una Entrada y guiarlo adelante sin cambios como una Salida.
    - Los atributos del grupo de datos de mensajes podrían ser, por ejemplo, “id del mensaje, el que lo envía, el que recibe, y contenido.
    - El objeto de interés del mensaje es ‘mensaje’.

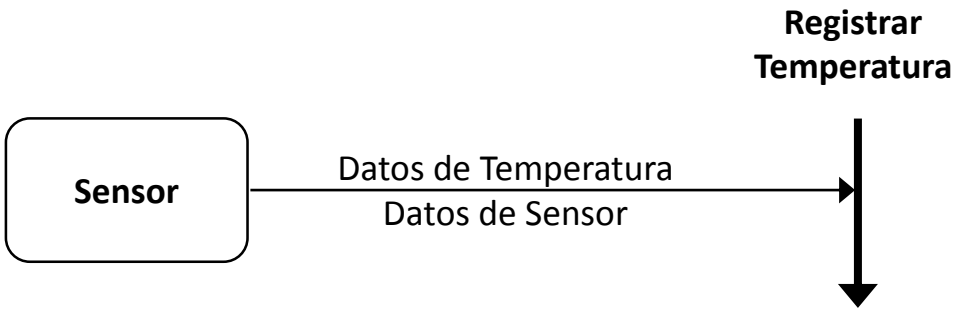
Grupos de datos o datos que NO son candidatos a movimientos de datos

Datos cualesquiera que aparezca en pantallas de entrada o salida o en informes y que no están relacionados con un objeto de interés para un usuario funcional NO deberían ser identificados indicando un movimiento de datos, por tanto no deberían medirse.

- Ejemplo:
  - Datos generales de la aplicación tales como encabezados y pies de página (nombre de la compañía, nombre de la aplicación, fecha del sistema, etc.) que aparecen en todas las pantallas.

Usuario Funcional como Objeto de Interés

- En ocasiones la Fase de Estrategia puede mostrar un dispositivo físico (un sensor) como un usuario funcional. Más tarde, en la Fase de Representación, este dispositivo puede ser identificado como un objeto de interés.
  - Esto es porque el dispositivo físico 'interactúa con el software' y, al mismo tiempo que es una "cosa" que requiere el software para que procese y/o almacene datos'.



Fase de Representación

- Fase de Representación
  - Introducción
  - Identificación de los procesos funcionales
  - Identificación de objetos de interés y grupos de datos
  - Identificación de los atributos de datos
  - Identificando los movimientos de datos

Atributo de datos

DEFINICIÓN – Atributo de datos
✓ La pieza más pequeña de información, dentro de un grupo de datos identificados, que tiene un significado desde la perspectiva de los requisitos funcionales del software.

- En el método COSMIC no es obligatorio identificar los atributos de los datos. Sin embargo, la comprensión del concepto de 'atributo de datos' es necesario para comprender la sección sobre ‘cambios de medición', donde un FUR que cambia un atributo de datos puede resultar en un movimiento de datos debiendo ser medido

Ejemplos

- *Aplicación de negocio:*
  - Atributos de datos son por ejemplo elementos de datos registrados en el diccionario de datos y atributos de datos que aparecen en un modelo de datos conceptual o lógico.
- *Tiempo real:*
  - Atributos de datos son por ejemplo ‘señal’, representando una señal recibida desde un sensor y ‘Emisor, Destinatario, Contenido’, representando partes de un mensaje en transmisión.

Facultad de Ciencias

Contacto

- Jesús Iván Saavedra Martínez
- jism@ciencias.unam.mx

Métricas de Software

Fase de Representación

Jesús Iván Saavedra Martínez

Octubre 2017

Fase de Representación

- Fase de Representación
  - Introducción
  - Identificación de los procesos funcionales
  - Identificación de objetos de interés y grupos de datos
  - Identificación de los atributos de datos
  - Identificando los movimientos de datos

Movimiento de datos

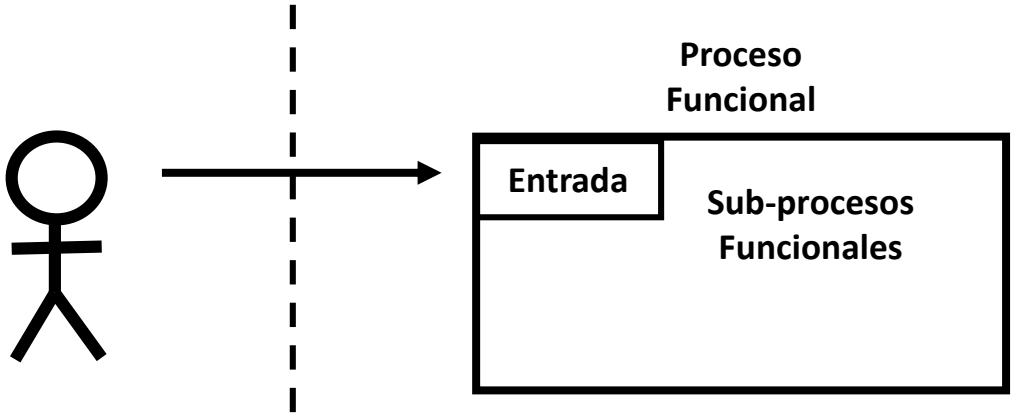
DEFINICIÓN – Movimiento de datos
✓ Un componente funcional base que mueve un único tipo de grupo de datos.

- Hay cuatro subtipos de movimiento de datos llamados: Entrada, Salida, Lectura y Escritura.
- Cada tipo de movimiento de datos se considera que incluye ciertas manipulaciones de datos asociadas.
- Es una *ocurrencia* de un movimiento de datos, no un tipo de movimiento de datos, que realmente mueve las *ocurrencias* del grupo de datos (no los tipos).

Entrada (Entry, E)

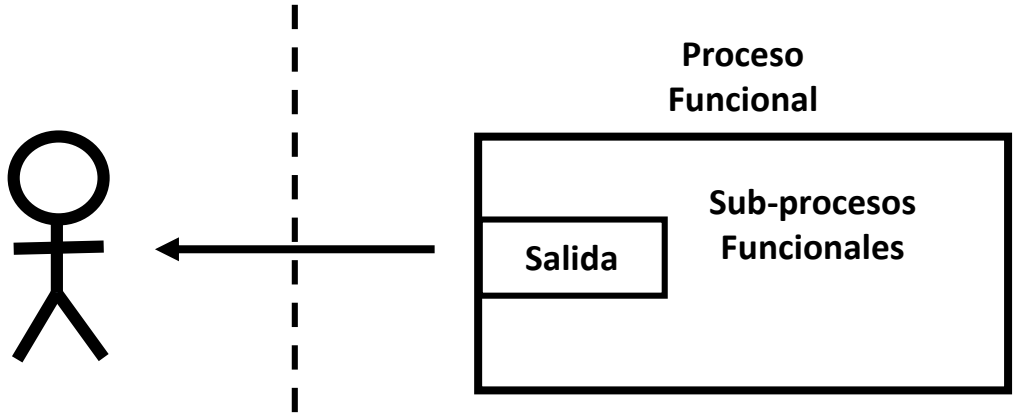
DEFINICIÓN – Entrada (E)

- ✓ Un movimiento de datos que mueve un grupo de datos desde un usuario funcional a través de la frontera hacia el proceso funcional donde se necesita



Salida (Exit, X)

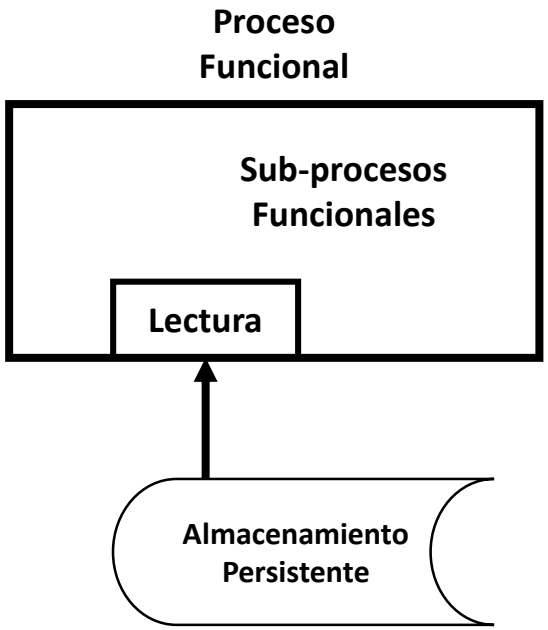
DEFINICIÓN – Salida (X)
✓ Un movimiento de datos que mueve un grupo de datos desde un proceso funcional a través de la frontera hacía el usuario funcional que los requiere.



Lectura (Read, R)

DEFINICIÓN – Lectura (R)

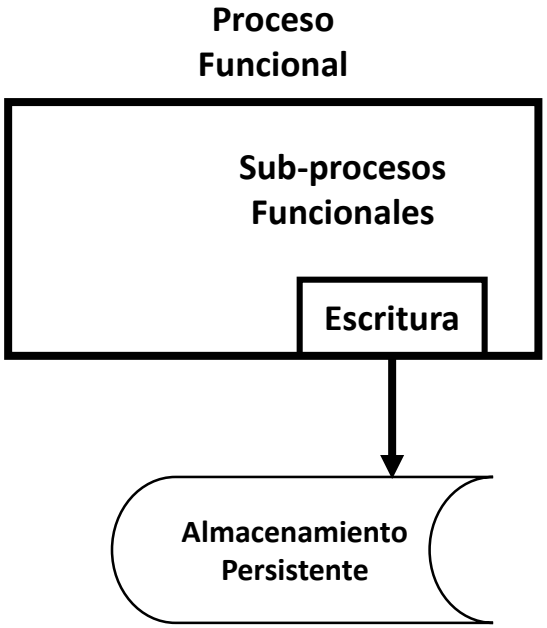
- ✓ Un movimiento de datos que mueve un grupo de datos desde un almacén persistente en el proceso funcional que los requiere.



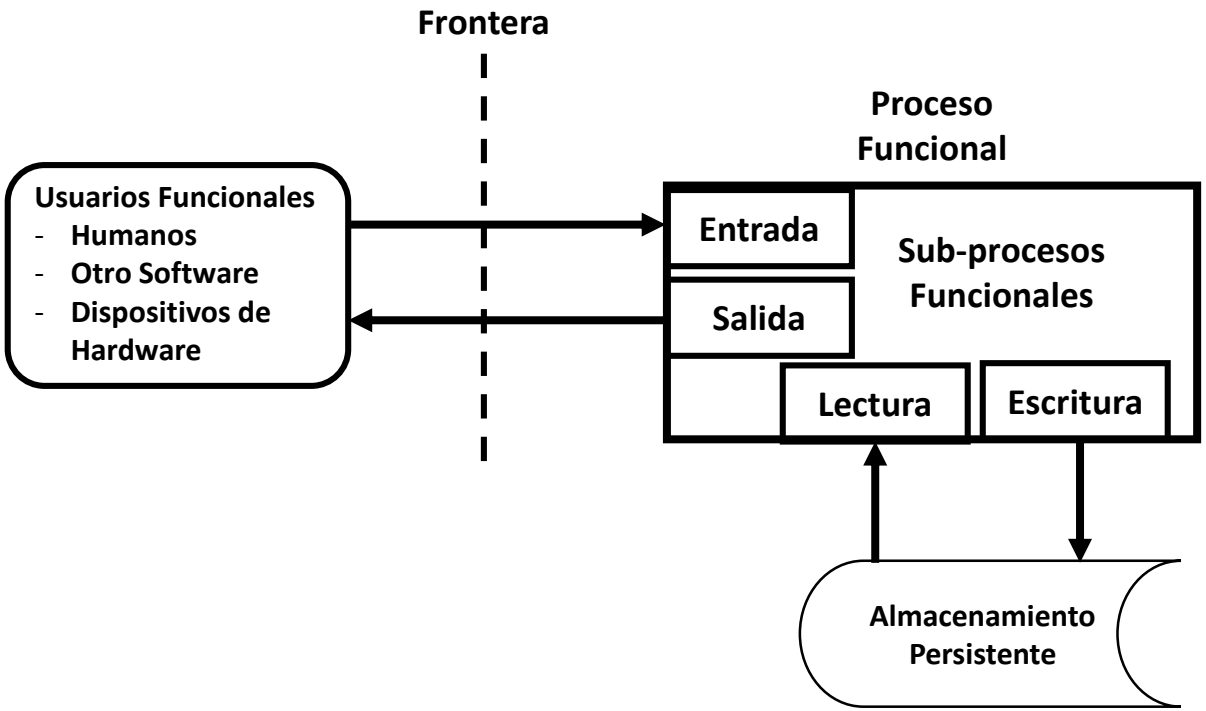
Escritura (Write, W)

DEFINICIÓN – Escritura (W)

- ✓ Un movimiento de datos que mueve un grupo de datos a un almacén persistente que se encuentra dentro de un proceso funcional.



Movimiento de datos



Los cuatro tipos de movimiento de datos, cada uno moviendo un grupo de datos, y su relación con un proceso funcional.

Principios de Entradas (E)

PRINCIPIOS – Entradas (E)

a) Una Entrada deberá mover un único grupo de datos describiendo un solo objeto de interés. Si la entrada a un proceso funcional comprende más de un grupo de datos, se identifica una entrada por cada grupo de datos que entra.

b) Una Entrada no deberá sacar datos a través de la frontera, ni leer ni escribir datos del/hacia el almacenamiento persistente.

Reglas de Entradas (E)

REGLAS – Entradas (E)

a) El grupo de datos de una Entrada desencadenante puede constar de sólo un atributo de datos.

b) Las marcas de tiempo (ticks) de reloj que están desencadenando eventos serán externos al software.

c) Salvo que sea necesario en un proceso funcional, obtener el tiempo del reloj del sistema no será considerado como causa de una Entrada.

Reglas de Entradas (E)

REGLAS – Entradas (E)

d) Si un evento desencadena la Entrada de un grupo de datos de “n” atributos de un objeto de interés y los FUR permiten que el mismo evento pueda desencadenar una Entrada de un subconjunto de ‘n’ atributos del objeto de interés, entonces identificar una entrada por los ‘n’ atributos.

e) En una pantalla que permite introducir datos, se debe analizar sólo las pantallas que están llenas de datos. Ignore cualquier pantalla ‘en blanco’, campos y otros encabezados que permiten comprender los datos de entrada requeridos.

Ejemplo

- *Aplicación de negocio: regla c)*
  - Cuando un proceso funcional añade una marca de tiempo a un registro que se hizo persistente, no se identifica una Entrada para la obtención del valor del reloj del sistema.

Principios de Salidas (X)

PRINCIPIOS – Salidas (X)

a) Una Salida deberá mover un único grupo de datos describiendo un solo objeto de interés. Si la salida de un proceso funcional comprende más de un grupo de datos, hay que identificar una salida para cada grupo de datos que sale.

b) Una Salida no introducirá datos a través de la frontera, ni leer ni escribir datos del/hacia el almacenamiento persistente.

Reglas de Salidas (X)

REGLAS – Salidas (X)

a) Una consulta que emite el texto fijo, se modela como que tiene una sola Salida por la salida de texto fijo.

b) Si la Salida de un proceso funcional mueve un grupo de datos de 'n' atributos de un objeto de interés y los FUR permiten que el proceso funcional pueda tener una Salida de un sub-conjunto de 'n' atributos del objeto de interés, entonces identificar una Salida por los 'n' atributos.

c) Al identificar Salidas, ignorar campos y encabezados que permiten a los usuarios comprender los datos de salida.

Principios de Lecturas (R)

PRINCIPIOS – Lecturas (R)

a) Una Lectura deberá mover un único grupo de datos describiendo un solo objeto de interés. Si el proceso funcional debe recuperar más de un grupo de datos del almacén, se debe identificar una Lectura para cada grupo de datos que es recuperado.

b) Una Lectura no recibirá ni sacará datos a través de la frontera ni escribirá datos al almacenamiento persistente.

Principios de Lecturas (R)

PRINCIPIOS – Lecturas (R)

c) No se considera como lectura el movimiento o manipulación de constantes o variables que son internas del proceso funcional y que sólo se pueden cambiar por un programador, o mediante los resultados de un cálculo, o de los datos almacenados en un proceso funcional.

d) Una Lectura siempre incluye cualquier funcionalidad de 'solicitud de lectura' (así, un movimiento de datos separado nunca será contado para cualquier funcionalidad de 'solicitud de lectura').

Reglas de Lecturas (R)

REGLAS – Lecturas (R)

a) Identifica una Lectura cuando el software debe recuperar un grupo de datos desde el almacenamiento persistente.

b) No identificar una Lectura cuando los FUR especifican algún usuario funcional de software o hardware como la fuente de un grupo de datos, o como los medios de recuperación de un grupo de datos almacenados.

Principios de Escrituras (W)

PRINCIPIOS – Escrituras (W)

a) Una Escritura deberá mover un único grupo de datos describiendo un solo objeto de interés. Si el proceso funcional debe pasar más de un grupo de datos al almacén persistente, identificar una Escritura por cada grupo de datos que se mueve al almacén persistente.

b) Una escritura no recibirá ni sacará datos de la frontera, ni leerá los datos.

c) Un requisito para borrar un grupo de datos de un almacén persistente se medirá como una sola Escritura.

Principios de Escrituras (W)

PRINCIPIOS – Escrituras (W)

d) Lo siguiente no podrá considerarse como movimientos de datos de Escritura:

- El movimiento o manipulación de los datos que no existía en el inicio de un proceso funcional y que no se ha hecho persistente cuando el proceso funcional es completado.
- Creación o actualización de variables o resultados intermedios que son internos al proceso funcional.
- Almacenamiento de los datos por un proceso funcional resultante sólo de la aplicación, en lugar de que estén establecidos en los FUR. (Un ejemplo es el procesamiento por lotes.)

Reglas de Escrituras (W)

REGLAS – Escrituras (W)

a) Identificar una Escritura en que, de acuerdo a los FUR, el software que se está midiendo debe mover un grupo de datos hacia el almacenamiento persistente.

b) No identificar una Escritura cuando los FUR del software que se está midiendo especifica algún usuario funcional de software o hardware como el destino del grupo de datos o como medio de almacenamiento del grupo de datos.

Manipulación de datos

DEFINICIÓN – Manipulación de datos
✓ Todo lo que le sucede a los datos, distinto de un movimiento de datos en o de un proceso funcional, o entre un proceso funcional y almacén persistente.

- Toda manipulación de datos se considera que se explica por los movimientos de datos asociados. Por lo tanto la manipulación de datos puede ser ignorada EXCEPTO si hay un FUR que debe ser medido para un *cambio* a la manipulación de datos.

Principios de Manipulación de datos

PRINCIPIOS – Manipulación de datos asociada con movimientos de datos

a) Todas las manipulaciones de datos en un proceso funcional serán asociadas con los cuatro tipos de movimiento datos (E, X, R, y W).

- Por convención, los movimientos de datos de un proceso funcional se supone que también representan la manipulación de los datos del proceso funcional.

Reglas de Manipulación de datos

REGLAS – Manipulación de datos asociada con movimientos de datos

a) Un movimiento de datos de Entrada representa toda manipulación de datos para permitir que un grupo de datos sea introducido por un usuario funcional y ser validado.

b) Un movimiento de datos de Salida representa toda manipulación de datos para crear los atributos de datos de un grupo de datos que va a ser la salida y/o para permitir que el grupo de datos sea la salida y para ser transferido al usuario funcional destinado.

Reglas de Manipulación de datos

REGLAS – Manipulación de datos asociada con movimientos de datos

c) Un movimiento de datos de Lectura representa todo cálculo y/o procesamiento lógico necesario para recuperar un grupo de datos desde el almacenamiento persistente.

d) Un movimiento de datos de Escritura representa todo cálculo y/o procesamiento lógico para crear o actualizar un grupo de datos, o eliminar un grupo de datos.

e) La manipulación de datos asociada con movimientos de datos no incluye manipulación de datos que se necesita después de que el movimiento se ha completado con éxito.

Ejemplos

- *Aplicación de negocio:*
  - Una Entrada incluye toda la manipulación necesaria para dar formato a una pantalla para permitir a un usuario humano introducir datos y validar los datos introducidos EXCEPTO cualquier Lectura(s) que pudiera requerirse para validar algún dato introducido.
- *Tiempo real:*
  - Una Salida incluye toda la manipulación para dar formato de salida y preparar algunos atributos de los datos para la impresión, incluyendo los encabezados de campo legibles EXCEPTO cualquier Lectura o Entrada que podrían ser necesarias para proporcionar los valores o las descripciones de algunos de los atributos de los datos impresos.

Movimiento de datos únicos y posibles excepciones

- Dos movimientos de datos no pueden ser considerados como diferentes a menos que tengan diferente FUR para su manipulación de datos asociada.

- Ejemplo:
  - Considere dos ocurrencias de un proceso funcional que requieren distintos subprocesos de manipulación de datos.
  - Ambas manipulaciones de datos de los subprocesos deben estar asociados con el mismo movimiento de datos y por lo tanto sólo un movimiento de datos debe ser identificado.

Reglas de Movimiento de datos únicos y posibles excepciones

REGLAS – Movimiento de datos únicos y posibles excepciones

a) Todos los datos que describen cualquier objeto de interés que se requiere para ser introducido en un proceso funcional deberá ser identificado como un grupo de datos movido por una Entrada.

b) Un FUR puede especificar diferentes grupos de datos que deben introducirse en un proceso funcional, donde cada grupo de datos describe el mismo objeto de interés. Una entrada se identificará para cada uno de estos diferentes grupos de datos.

Reglas de Movimiento de datos únicos y posibles excepciones

REGLAS – Movimiento de datos únicos y posibles excepciones

c) Un FUR puede especificar diferentes grupos de datos, que describen el mismo objeto de interés, a ser movidos desde el almacenamiento persistente a un proceso funcional. Una Lectura se identificará para cada uno de estos grupos de datos. La misma regla equivalente se aplica para Escrituras.

d) No se contarán ocurrencias repetidas de cualquier tipo de movimiento de datos cuando se está ejecutando. El número de ocurrencias es irrelevante para medir tamaño funcional.

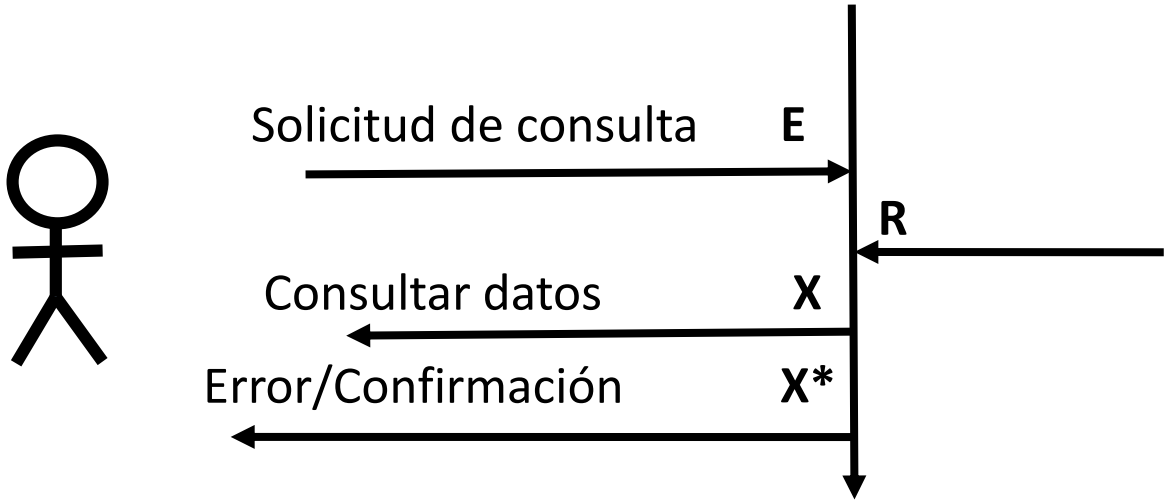
e) Esto se aplica incluso si múltiples ocurrencias del tipo de movimiento de datos difieren en su ejecución.

Ejemplos

- *Aplicación de negocio: regla d)*
  - Supongamos que una Lectura se requiere en el FUR y que en la práctica requiere de muchas ocurrencias de recuperación, como en una búsqueda a través de un archivo. Identificar una sola lectura.
- *Tiempo real: regla b)*
  - Se necesita un proceso funcional para aceptar diferentes grupos de datos a partir de dos sismógrafos diferentes (usuarios funcionales) que describen cada uno el mismo objeto de interés (el evento), por ejemplo, una explosión de ensayo. Identificar dos Entradas.

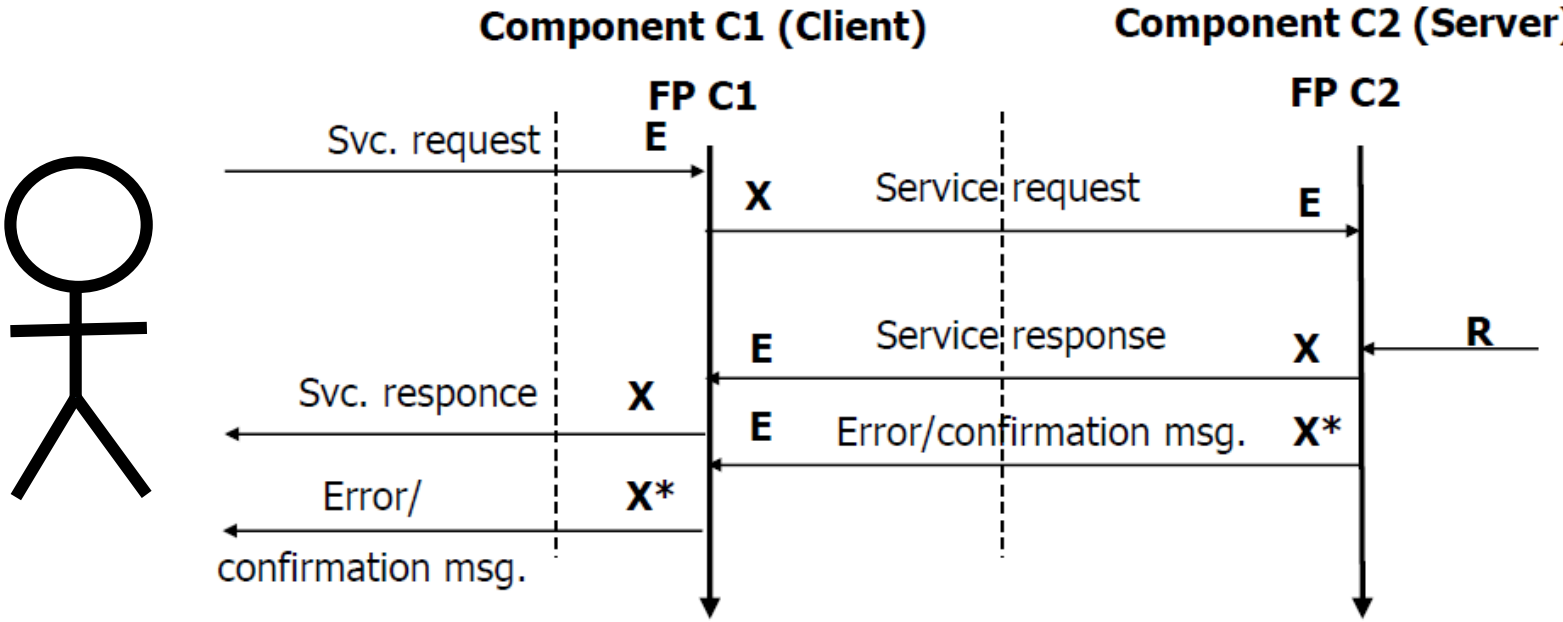
Moviendo datos desde el almacenamiento persistente

- Ejemplo 1:
  - Supongamos que una Lectura se requiere en el FUR y que en la práctica requiere de muchas ocurrencias de recuperación, como en una búsqueda a través de un archivo. Identificar una sola lectura.



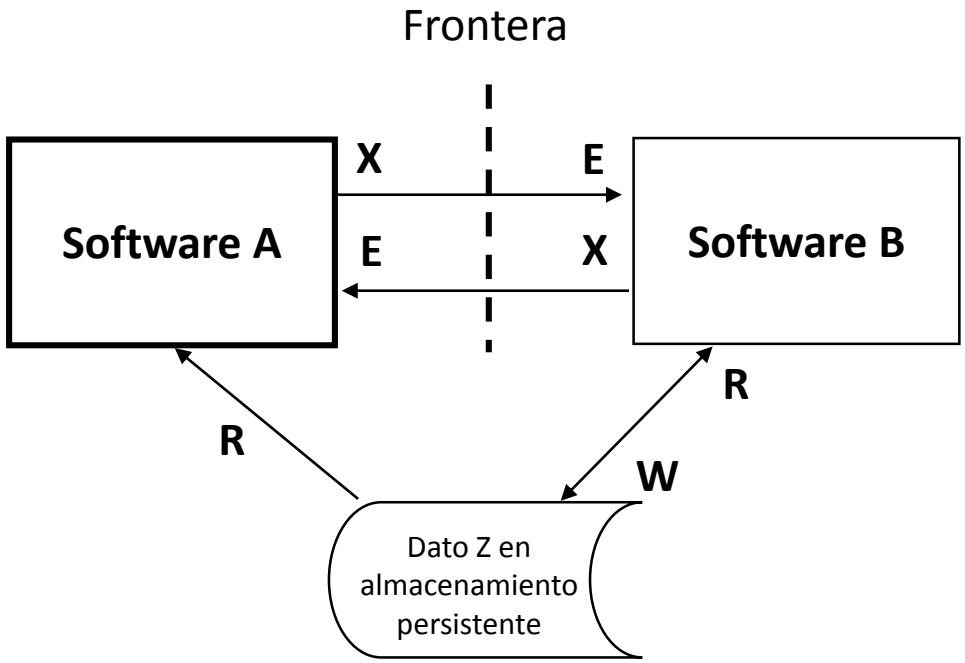
Moviendo datos desde el almacenamiento persistente

- Ejemplo 2:
  - Para un software con una arquitectura ‘cliente-servidor’, donde un proceso funcional de un cliente debe solicitar el servidor para algunos datos persistentes.



Moviendo datos desde el almacenamiento persistente

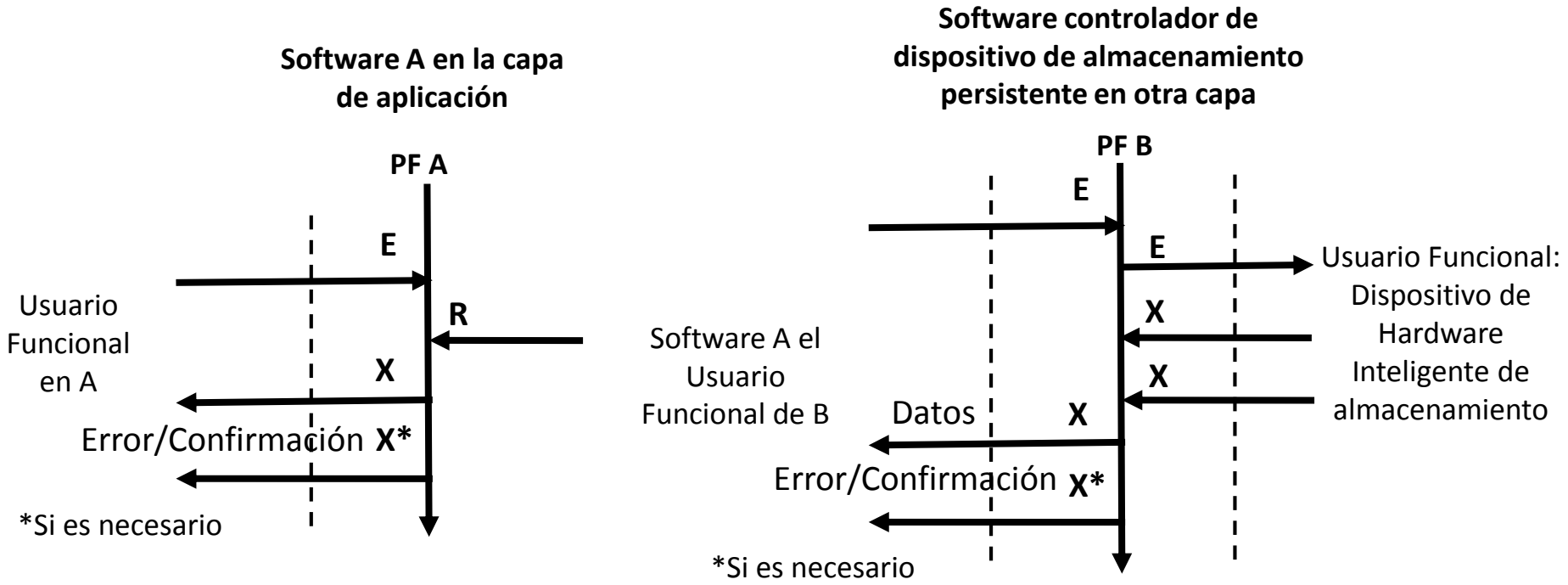
- Ejemplo 3:
  - El software cuenta con diferentes derechos de acceso a los datos almacenados para diferentes propósitos.



Datos persistentes Z dentro de la frontera de ambos programas A y B para una Lectura

Moviendo datos desde el almacenamiento persistente

- Ejemplo 4:
  - Se aplica cuando los datos deben ser obtenidos directamente de un almacenamiento de hardware físico tal vez por el software de controlador de dispositivo.



Solución para una Lectura del software de A en la capa de aplicación al software B en la capa de controlador de dispositivo

Reglas Cuando un proceso funcional requiere datos de un usuario funcional

REGLAS – Cuando un proceso funcional requiere datos de un usuario funcional

a) Cuando un proceso funcional deberá obtener un grupo de datos y no tiene que decirle al usuario funcional qué datos va a enviar, ocurre en cualquiera de los siguientes casos:

1. Cuando un usuario funcional envía un grupo de datos a través de una Entrada desencadenante que inicia del proceso funcional.
2. Cuando un proceso funcional, habiendo recibido un grupo de datos a través de una Entrada, espera el próximo grupo de datos.
3. Cuando un proceso funcional, habiéndose iniciado, pide al usuario funcional 'envíame tus datos ahora, si tienes alguno', y el usuario funcional envía sus datos, o inspecciona los datos de un usuario funcional y recupera los datos que necesita.

Reglas Cuando un proceso funcional requiere datos de un usuario funcional

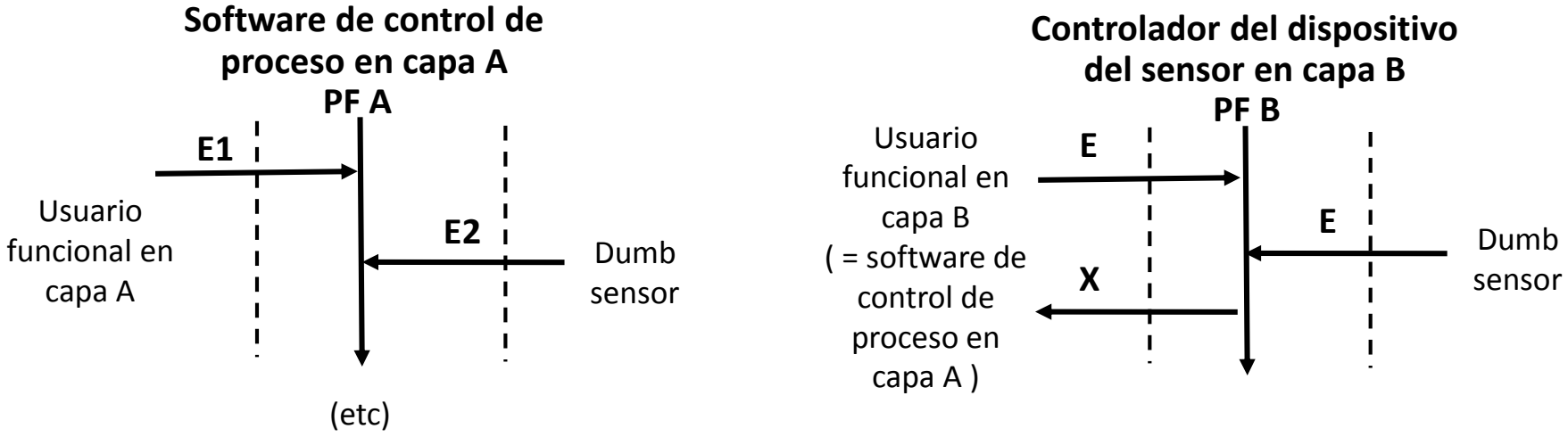
REGLAS – Cuando un proceso funcional requiere datos de un usuario funcional

b) Cuando un proceso funcional debe obtener los servicios de un usuario funcional y las necesidades de los usuarios le digan lo que debe enviar.

- Un movimiento de datos de Salida seguida de uno de Entrada serán identificados. La Salida contiene la solicitud específica de los datos; la Entrada contiene los datos.

Ejemplo

- *Tiempo real*:
  - Supongamos que un proceso funcional de una aplicación de software de control en tiempo real es requerido para comprobar un arreglo de sensores idénticos.



Solución para una entrada emitida por el software 'A' en la capa de la aplicación de control de proceso recibida por el software 'B' en la capa del dispositivo de control del sensor

Comandos de control

DEFINICIÓN – Comando de control
✓ Un comando que le permite al usuario funcional humano controlar el uso del software pero que no involucra ningún movimiento de datos sobre el objeto de interés del software que se está midiendo.

- NOTA: Un comando de control no es un movimiento de datos porque el comando no mueve datos acerca de un objeto de interés. Ejemplos de ello son comandos de ‘página arriba / abajo; presionar la tecla Tab o tecla Enter, hacer clic 'OK' para confirmar una acción anterior, al pulsar un botón para continuar, etc.

Reglas de Comando de control

REGLAS – Comando de control

a) En una aplicación con interfaz para humanos los ‘comandos de control' serán ignorados, ya que no implican ningún movimiento de datos sobre un objeto de interés.

b) Un ‘comando de control’ es un comando que es reconocida en cualquier aplicación que puede ser utilizada por los usuarios funcionales humanos y que deben ser ignorados cuando se mide un tamaño funcional.

Mensajes de error/confirmación

DEFINICIÓN – Mensajes de error/confirmación
✓ Una Salida emitida por un proceso funcional a un usuario humano funcional que, o bien confirma solamente que los datos han sido aceptados, o solamente que hay un error en los datos introducidos.

- NOTA: Cualquier Salida que puede incluir indicaciones de fallo, pero que no está destinado a un usuario funcional humano, no es un mensaje de error/confirmación.

Reglas de Mensajes de error/confirmación

REGLAS – Mensajes de error/confirmación y otras condiciones de error

a) Una salida se identificará para contabilizar todos los tipos de mensajes de error/confirmación emitidos por una proceso funcional

- Por ejemplo, éxitos o fracasos de: validación de los datos, recuperación datos, para hacerlos persistentes, o para la respuesta de un servicio solicitado a otra aplicación.
- NOTA: Si los FUR del proceso funcional no requieren ningún tipo de mensaje de error/confirmación a emitir, no identificar ninguna Salida correspondiente. casos:

Reglas de Mensajes de error/confirmación

REGLAS – Mensajes de error/confirmación y otras condiciones de error

b) Si un mensaje a un usuario funcional humano proporciona datos además del error/confirmación, entonces estos datos adicionales deben ser identificados como una Salida, además de la Salida de error/confirmación.

c) Todos los demás datos, emitidos o recibidos por el software deben ser analizados de acuerdo con los FUR como Salidas o Entradas, respectivamente, independientemente de si o no los valores de los datos indican una condición de error.

Reglas de Mensajes de error/confirmación

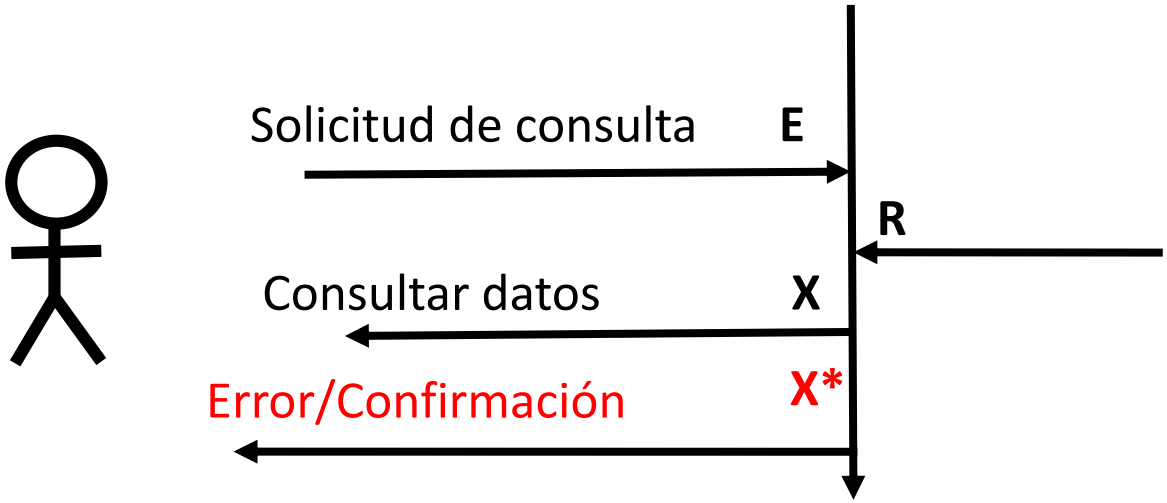
REGLAS – Mensajes de error/confirmación y otras condiciones de error

d) Las Lecturas y Escrituras se consideran que incluyen cualquier informe de condiciones de error. Por lo tanto ninguna Entrada al proceso funcional se debe identificar para cualquier indicación de error.

e) Ninguna Entrada o Salida se identificarán para cualquier mensaje que indica una condición de error que pueda ser emitido pero que no se requiere que sea procesado en modo alguno por el FUR de ese software.

Ejemplo 1

- *Aplicación de negocios: regla a)*
  - En un diálogo humano-computadora, ejemplos de mensajes de error que se producen durante la validación de datos que se introducen podría ser ‘error de formato’, ‘cliente no encontrado', ’límite de crédito superado', etc.



Ejemplo 2

- *Aplicación de negocios: regla a)*
  - Un proceso funcional 'A' pueden emitir 2 mensajes de confirmación distintos y 5 mensajes de error a sus usuarios funcionales.
  - ➢ Identifique una Salida para dar cuenta de todos estos mensajes de error/confirmación.

  - Un proceso funcional 'B' puede emitir 8 mensajes de error a sus usuarios funcionales.
  - ➢ Identifique una Salida para dar cuenta de estos 8 mensajes de error. .

Ejemplo 3

- *Aplicación de negocios: regla b)*
  - En un diálogo humano-computadora, si un mensaje se emite en las situaciones de error pero contiene datos de usuario funcionales.
  - Entonces debería ser considerado como una Salida en el proceso funcional donde se produce, además de la Salida de error/confirmación.
  - Un ejemplo de un mensaje de este tipo podría ser ‘Advertencia: la cantidad que desea retirar excede su límite de crédito en $ 100.

Ejemplo 4

- *Aplicación de negocios: regla e)*
  - Emisión de mensajes de error a los usuarios humanos, pero que no son generados o tratados por el software deben ser completamente ignorado en la medición.
  - Un ejemplo de tal mensaje transmitido desde el sistema operativo podría ser "impresora X no está respondiendo'. Entonces debería ser considerado como una Salida en el proceso funcional donde se produce, además de la Salida de error/confirmación.

Facultad de Ciencias

Contacto

- Jesús Iván Saavedra Martínez
- jism@ciencias.unam.mx

Métricas de Software

Fase de Medición

Jesús Iván Saavedra Martínez

Octubre 2017

COSMIC

- Introducción
- Fase de estrategia de medición
- Fase de representación
- **Fase de medición**
- Informe de medidas

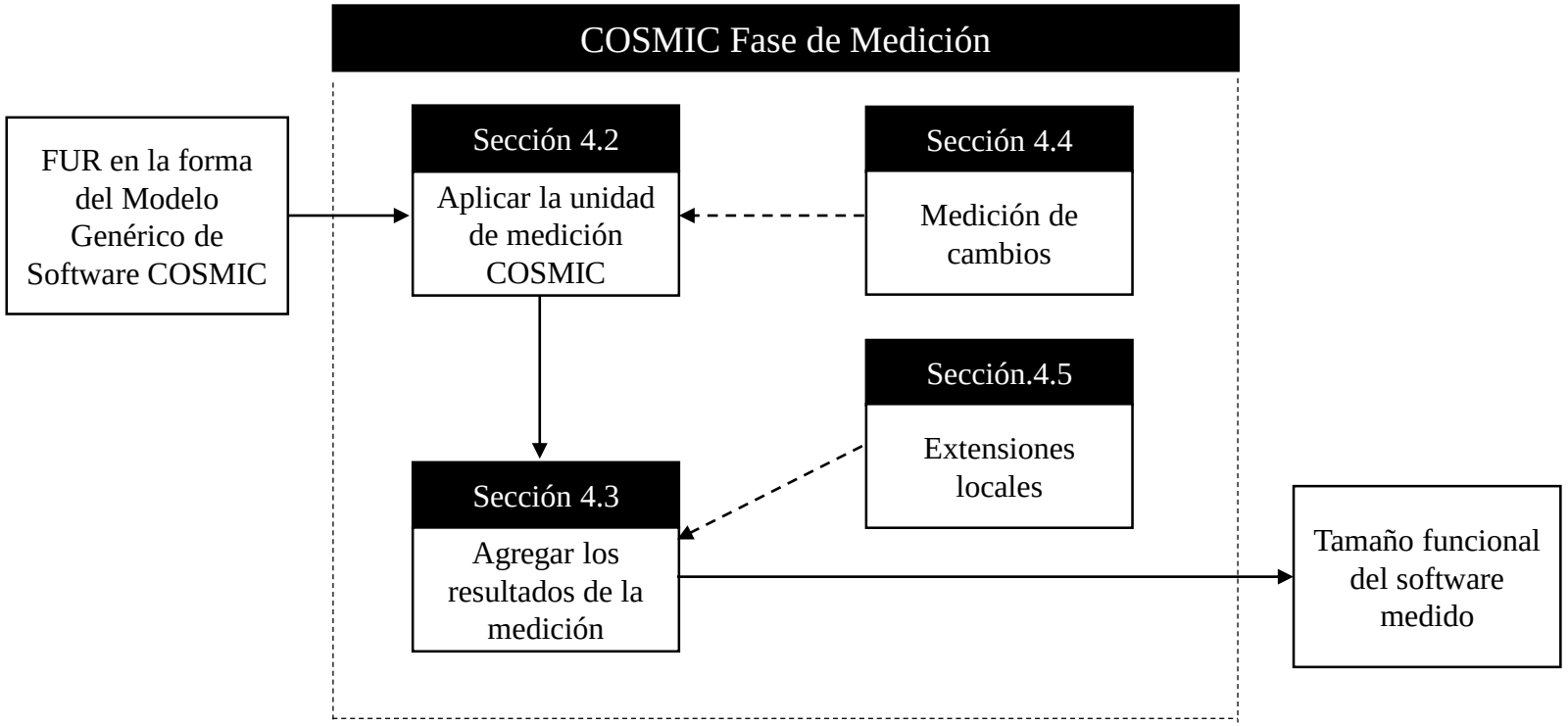
Fase de Medición

- Fase de Medición
  - Introducción
  - Aplicación de la unidad de medida COSMIC
  - Agregación de resultados de medición
  - Más en la medición del tamaño de los cambios de software
  - Extendiendo el método de medición COSMIC

Fase de Medición

- En esta fase se define la unidad de medida COSMIC.
- Se dan las reglas para asignar un tamaño a los FUR del software que se está midiendo. Las reglas se definen para saber cómo agregar tamaños de diferentes piezas de software.
- Las normas se definen para saber cómo medir cambios al software que son necesarios.
- Discute la posibilidad de 'extensiones locales' para el método COSMIC estándar que pueden utilizarse.

Fase de Medición



Proceso general de la Fase de Medición COSMIC

Fase de Medición

- Fase de Medición
  - Introducción
  - Aplicación de la unidad de medida COSMIC
  - Agregación de resultados de medición
  - Más en la medición del tamaño de los cambios de software
  - Extendiendo el método de medición COSMIC

Unidad de medida COSMIC

DEFINICIÓN – Unidad de medida COSMIC
✓ 1 CFP (Punto de Función COSMIC) que se define como el tamaño de un movimiento de datos.

- Un movimiento de datos se mide como un Punto de Función COSMIC, o ‘CFP’.
- NOTA: La unidad de medida se conoce como un 'Cfsu', antes de la versión 3.0 del método (unidad de tamaño funcional COSMIC).

Fase de Medición

- Fase de Medición
  - Introducción
  - Aplicación de la unidad de medida COSMIC
  - Agregación de resultados de medición
  - Más en la medición del tamaño de los cambios de software
  - Extendiendo el método de medición COSMIC

Agregación de resultados de medición

REGLAS – Agregando los resultados de la medición

a) Para cualquier proceso funcional, el tamaño funcional de cada movimiento de datos individual debe ser agregado en un único valor de tamaño funcional en unidades de CFP.

Tamaño (proceso funcional*i*) = Σ tamaño(Entradas*i*) +

Σ tamaño(Salidas*i*) +

Σ tamaño(Lecturas*i*) +

Σ tamaño(Escrituras*i*)

Agregación de resultados de medición

REGLAS – Agregando los resultados de la medición

b) Para cualquier proceso funcional, el tamaño funcional de los cambios se sumarán al tamaño de movimientos de datos que han sido añadidos, modificados o eliminados para dar un tamaño del cambio en unidades de CFP.

Tamaño (Cambio proceso funcional*i*) =

Σ tamaño(movimientos de datos añadidos*i*) +

Σ tamaño(movimiento de datos modificados*i*) +

Σ tamaño(movimiento de datos eliminados*i*)

Agregación de resultados de medición

REGLAS – Agregando los resultados de la medición

c) El tamaño de una pieza de software se obtendrá sumando los tamaños de sus procesos funcionales, sujeto a las normas e) y f) a continuación.

d) El tamaño de cualquier cambio se obtendrá sumando los tamaños de todos los cambios de todos los procesos funcionales, sujeto a las normas e) y f) a continuación.

e) Los Tamaños de piezas de software o de cambios de piezas de software podrán sumarse sólo si se mide al mismo nivel de granularidad de los FUR el proceso funcional.

f) Los Tamaños de piezas de software y/o cambios en los tamaños de piezas de software dentro de una capa o de diferentes capas serán sumados sólo si tiene sentido hacerlo, a efectos de la medición.

Agregación de resultados de medición

REGLAS – Agregando los resultados de la medición

g) El tamaño de una pieza de software se obtiene sumando los tamaños de sus componentes (independientemente de cómo se descompone) proporcionados

- El tamaño de las contribuciones de los movimientos de datos inter-componentes son eliminadas y
- Sólo una Salida se identifica para todos los mensajes de error/confirmación emitidos por un proceso funcional a un usuario funcional humano.

h) Si el método COSMIC se extiende localmente, entonces el tamaño medido por la extensión local debe ser informado por separado como se describe en la sección y no puede ser añadido al tamaño obtenido por el método estándar, medido en CFP.

Ejemplo

- *Para las reglas b) y c): Un cambio solicitado para una pieza de software podría ser:*
  - Añadir un nuevo proceso funcional de tamaño 6 CFP.
  - En otro proceso funcional agregar un movimiento de datos.
  - Hacer modificaciones a otros tres movimientos de datos.
  - Eliminar dos movimientos de datos.

➢ El tamaño total del cambio solicitado es de:

6 + 1 + 3 + 2 = 12 CFP.

Agregación del tamaño funcional

- En un contexto donde el tamaño funcional deba utilizarse como una variable en un modelo para estimar el esfuerzo, la agregación normalmente se realizará por capas debido a que el software en diferentes capas a menudo no se implementa con la misma tecnología.
- El software cada capa es un usuario funcional en otras capas que usa y cualquier pieza de software de una capa puede ser un usuario funcional de cualquier otra pieza de software semejante en la misma capa.
- Es por lo tanto lógico solamente que los tamaños de las distintas piezas puedan sumarse, siempre sujeto a las normas d), e) y f).

Fase de Medición

- Fase de Medición
  - Introducción
  - Aplicación de la unidad de medida COSMIC
  - Agregación de resultados de medición
  - Más en la medición del tamaño de los cambios de software
  - Extendiendo el método de medición COSMIC

Más en la medición del tamaño de los cambios de software

- Un ‘cambio funcional’ de un software existente es interpretado en el método COSMIC como ‘cualquier combinación de sumas de nuevos movimientos de datos o de modificaciones o eliminaciones de los movimientos de datos existentes incluyendo la manipulación de datos asociada’.
- Los términos ‘Mejora’ y ‘Mantenimiento’ se utilizan a menudo para lo que aquí llamamos un ‘cambio funcional’.

Más en la medición del tamaño de los cambios de software

- La necesidad de un cambio de software puede surgir de cualquier:
  - Nuevo FUR (es decir, sólo adiciones a la funcionalidad existente)
  - Desde un cambio en el FUR (quizá agregaciones, modificaciones y eliminaciones)
  - Desde un 'mantenimiento', la necesidad de corregir un defecto.

- Cuando un software es reemplazado por completo, el tamaño funcional de este cambio es el tamaño del software de reemplazo
- Nota: Las reglas para la medición de cualquiera de estos cambios son las mismas.

Modificando Funcionalidades

DEFINICIÓN – Modificación (de la funcionalidad de un movimiento de datos)

a) Un movimiento de datos se considera que es modificado funcionalmente si al menos aplica una de las siguientes consideraciones:
   - El grupo de datos movido es modificado,
   - La manipulación de datos asociada es modificada.

Modificando Funcionalidades

DEFINICIÓN – Modificación (de la funcionalidad de un movimiento de datos)

b) Un grupo de datos es modificado si al menos aplica una de las siguientes consideraciones:
   - Uno o más atributos nuevos son agregados al grupo de datos.
   - Uno o más atributos existentes son removidos del grupo de datos.
   - Uno o más atributos existentes son modificados, por ejemplo, en el significado o en el formato (pero no en sus valores)

Modificando Funcionalidades

REGLAS – Modificando un movimiento de datos

a) Si un movimiento de datos debe ser modificado debido a un cambio de la manipulación de datos y/o debido a un cambio en el número o tipo de los atributos del grupo de datos, un cambio CFP se medirá, independientemente de la cantidad real de modificaciones en el movimiento de datos.

b) Si un grupo de datos debe ser modificado, los movimientos de datos moviendo el grupo de datos modificado, cuya funcionalidad no se ve afectada por la modificación del grupo de datos, no serán identificados como movimientos de datos modificados

Ejemplo

Una solicitud de cambio para un proceso funcional requiere:

- Tres cambios a la manipulación de datos asociada con su Entrada desencadenante
- Dos cambios a la manipulación asociada con una Salida
- Dos cambios en los atributos del grupo de datos movidos por esta Salida.

✓ Medir el tamaño del cambio como 2 CFP, es decir, contar el número total de movimientos de datos cuyos atributos y manipulación de datos asociada debe ser cambiada.

✓ NO contar el número de manipulaciones de datos o datos de atributos para ser cambiados.

Tamaño del software funcionalmente modificado

Después del cambio funcional de una pieza de software, su nuevo tamaño total es igual a:

- El tamaño original
- más el tamaño funcional de todos los movimientos de datos agregados
- menos el tamaño funcional de todos los movimientos de datos eliminados
- Los movimientos de datos modificados no tienen influencia sobre el tamaño de la pieza de software ya que estos existen tanto antes como después de que las modificaciones han sido realizadas.

Fase de Medición

- Fase de Medición
  - Introducción
  - Aplicación de la unidad de medida COSMIC
  - Agregación de resultados de medición
  - Más en la medición del tamaño de los cambios de software
  - Extendiendo el método de medición COSMIC

Extendiendo el método COSMIC

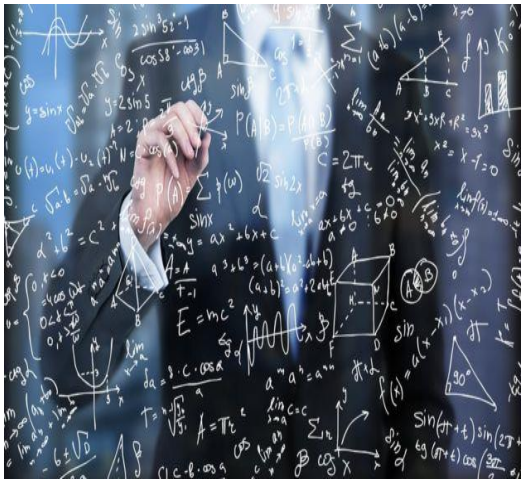
- COSMIC no pretende medir todos los aspectos del 'tamaño' del software
- El método no está actualmente diseñado para medir por separado y de forma explícita el tamaño de sub-procesos de manipulación de datos.
- El tamaño de manipulación de datos se toma en cuenta a través de un supuesto simplificador de que es válido para una amplia gama de ámbitos de software
- El número de atributos de datos en un movimiento de datos no se captura en tamaño del software.
- La medida de tamaño COSMIC se considera una buena aproximación para el estado del método propuesto y el ámbito de aplicabilidad.
- El método de medición COSMIC permite extensiones locales, donde una organización puede tener un estándar local donde tome en cuenta parámetros que el método COSMIC no tome en cuenta en la medición.

Manipulación de software rico en datos

- El método COSMIC fue diseñado para medir software 'rico en movimiento de datos’.
- Al igual que todos los demás FSM, no fue diseñado para medir de forma explícita la funcionalidad de manipulación de datos.
- Asume que los tipos de movimiento de datos contabilizan la funcionalidad de manipulación de datos asociados.
- La experiencia ha demostrado que puede también ser aplicado con éxito a tamaño software 'rico en manipulación de datos‘.
- Donde el método no puede contabilizar adecuadamente la manipulación de datos, puede ser posible desarrollar una extensión local del método para superar la limitación.

Extensiones locales con algoritmos complejos

- Si se considera necesario para tener en cuenta algoritmos complejos, puede organizarse un estándar local para esta funcionalidad excepcional.
- En cualquier proceso funcional donde existe una manipulación de datos anormalmente compleja de un sub-proceso funcional, el medidor está libre de asignar su propia determinación-local de puntos de función.



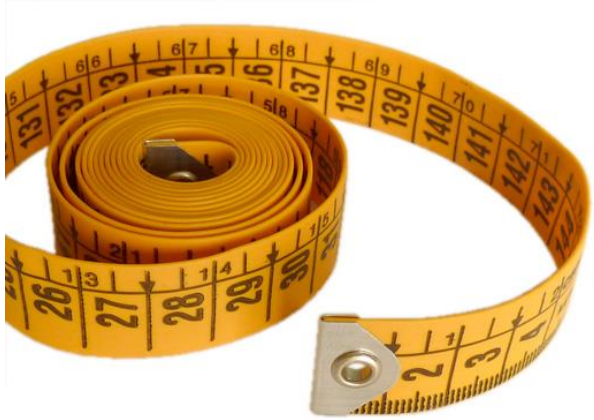
Una extensión local estándar puede ser “En nuestra organización”

- Un punto de función local FP se asigna a algoritmos matemáticos complejos.

Extensión local con sub-unidades de medida

Cuando se necesita más precisión en la medición de los movimientos de datos, entonces se puede definir una sub-unidad de medida.

- Por ejemplo, un metro puede ser sub-dividido en 100 centímetros o 1000 milímetros. Análogamente, el movimiento de un único atributo podría ser utilizado como subunidad de medida.



Facultad de Ciencias

Extensión local con sub-unidades de medida

- Las mediciones sobre una pequeña muestra de software indican que el número promedio de atributos de datos por movimiento de datos no varía mucho en los cuatro tipos de movimiento de datos.
  - Por esta razón y para facilitar las mediciones, la unidad de medida de COSMIC, 1 CFP, ha sido fijada en el nivel de un movimiento de datos.

- Cualquiera que desee refinar el método COSMIC con una sub-unidad de medida es libre de hacerlo, dejando en claro que el tamaño resultante no se expresa en el estándar de puntos de función COSMIC.

COSMIC

- Introducción
- Fase de estrategia de medición
- Fase de representación
- Fase de medición
- Informe de medidas

Etiquetado de las mediciones COSMIC

Los resultados de una medición COSMIC se deberían anotar como ‘x CFP(v.y)’, donde:

- ‘x’ representa el valor numérico del tamaño funcional.
- ‘v.y’ representa la identificación de la versión estándar del método COSMIC usado para obtener el valor numérico del tamaño funcional “x”.

EJEMPLO: Un resultado obtenido se debería anotar como ‘100 CFP (v4.0.1)’

Etiquetado de extensiones locales COSMIC

Un resultado de una medición COSMIC utilizando extensiones locales debería ser anotado como:

x CFP (v. y) + z Local FP', donde:

- 'x' representa el valor numérico obtenido al agregar todos los resultados de medidas individuales de acuerdo al método estándar COSMIC, versión v.y.
- 'v.y' representa la identificación de la versión estándar del método COSMIC usado para obtener el valor numérico de tamaño funcional "x".
- 'z' representa el valor numérico obtenido al agregar todos los resultados individuales de medida obtenidos de las extensiones locales al método COSMIC..

EJEMPLO: '100 CFP (v4.0.1) + 20 Local FP'

Archivando los resultados de medición COSMIC

Además de las medidas reales, descritas en el punto 5.1, todos o algunos de los siguientes atributos de cada medición se debe registrar:

- Identificación del componente de software medido (nombre, versión ID)
- Las fuentes de información usadas para identificar los FUR utilizados.
- El ámbito del software.
- Una descripción de la arquitectura en capas en las que la medición es realizada, si aplica.
- Una especificación del propósito de la medición.
- Una descripción del alcance de la medición.
- El patrón de medida utilizado (COSMIC o local).
- Los usuarios funcionales del software
- El nivel de granularidad de los artefactos de software disponibles y el nivel de descomposición del software.
- El punto en el ciclo de vida del proyecto cuando se hizo la medición.
- El objetivo o margen de error que se considera para la medida

Archivando los resultados de medición COSMIC

- Indicaciones de si el método de medición estándar COSMIC fue utilizado, y/o se usó una aproximación local al método estándar y/o se utilizaron extensiones locales
- Una indicación de si la medida es de funcionalidad desarrollada o entregada (funcionalidad 'desarrollada' se obtiene creando nuevo software; funcionalidad 'entregada' incluye funcionalidad desarrollada y también incluye funcionalidad obtenida por otros medios diferentes a crear nuevo software).
- Una indicación de si la medida es de funcionalidad recientemente proporcionada o si es el resultado de una actividad de 'mejora'.
- El número de componentes importantes, si es aplicable, cuyos tamaños han sido añadidos para el tamaño total registrado.
- El porcentaje de funcionalidad implementada por el software de re-utilización.
- Alcance dentro del alcance global de la medición, si aplica.
- El nombre del medidor y cualquier certificación de COSMIC, la fecha de la medición.

Facultad de Ciencias

Contacto

- Jesús Iván Saavedra Martínez
- jism@ciencias.unam.mx

Universidad Nacional Autónoma de México

UNAM

Facultad de Ciencias

Métricas de Software

Ejemplos de Medición COSMIC

Jesús Iván Saavedra Martínez

Octubre 2017

Agenda

I. Caso de uso: Iniciar Sesión

II. Medición Caso de uso: Iniciar Sesión

III. Caso de uso: “Administración de avisos de venta de vehículos”

IV. Medición Caso de uso: Administración de avisos de venta de vehículos

V. Medición Caso de estudio: Compra-Venta de Autos

Caso de estudio “Administración de avisos de venta de vehículos”

- Flujo básico

Acción que realiza el usuario	Acción que realiza el sistema
1. Ingresa al módulo de inicio de sesión.	2. Muestra en la pantalla de inicio de sesión, un formulario para capturar la siguiente información: • Nombre de usuario • Contraseña
3. Captura el nombre de usuario y contraseña.	4. Valida el nombre de usuario y contraseña. En caso de que los datos sean incorrectos, muestra mensaje de error.
	5. Muestra la pantalla de inicio del usuario.
Fin del flujo básico	

Facultad de Ciencias

Agenda

I. Caso de uso: Iniciar Sesión

II. Medición Caso de uso: Iniciar Sesión

III. Caso de estudio “Administración de avisos de venta de vehículos”

IV. Medición Caso de uso: Administración de avisos de venta de vehículos

V. Medición Caso de estudio: Compra-Venta de Autos

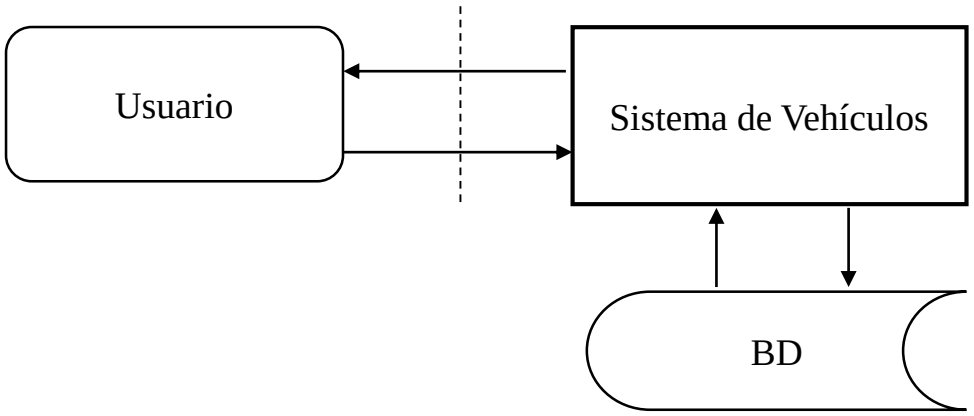
Medición Iniciar Sesión

Fase de Estrategia de Medición

- Propósito: determinar el tamaño funcional del caso de uso Iniciar Sesión como propósito de ejemplo.
- Alcance: global, ya que la medición considera toda la funcionalidad del caso de uso.
- Identificación de capas: se considera una sola capa global de aplicación para la medición del caso de uso.
- Identificación de usuarios funcionales: Se identifica un solo usuario funcional genérico, el cual interactúa con el sistema.

Medición Iniciar Sesión

- Nivel de granularidad: de las descripciones textuales del caso de uso, se identifica un nivel de granularidad adecuado para realizar la medición.
- Diagrama de contexto:



Medición Iniciar Sesión

Fase de Representación

- Identificación de procesos funcionales: se identifica un solo evento desencadenante donde un usuario desea ingresar al sistema, identificando un solo proceso funcional que atiende a este evento.

Evento desencadenante	Proceso funcional
Usuario desea ingresar al sistema	Iniciar sesión

- Identificación de objetos de interés y grupos de datos: se identifica un solo objeto de interés llamado “Usuario” el cual tiene un grupo de datos llamado “Datos de usuario”.

Objeto de interés	Grupos de Datos
Usuario	Datos de Usuario

Medición Iniciar Sesión

Identificación de Movimientos de Datos y Fase de Medición

Proceso Funcional	Evento Desencadenante	Descripción de Subprocesos	Grupo de Datos	Mov. de datos	CFP
Iniciar sesión	Usuario desea ingresar al sistema	Usuario ingresa al módulo de inicio de sesión.	N/A	N/A	0
		Sistema muestra en la pantalla de inicio de sesión, un formulario para capturar la siguiente información: •Nombre de usuario •Contraseña	N/A	N/A	0
		Usuario captura nombre y contraseña.	Datos de Usuario	E	1
		Sistema valida nombre y contraseña.	Datos de Usuario	R	1
		Sistema muestra la pantalla de inicio.	N/A	N/A	0
		Sistema muestra mensaje de error en caso de que los datos sean incorrectos.	Mensajes de error/ confirmación	X	1
				Total:	3 CFP v4.0.1

Agenda

I. Caso de uso: Iniciar Sesión

II. Medición Caso de uso: Iniciar Sesión

III. Caso de estudio “Administración de avisos de venta de vehículos”

IV. Medición Caso de uso: Administración de avisos de venta de vehículos

V. Medición Caso de estudio: Compra-Venta de Autos

Caso de estudio “Administración de avisos de venta de vehículos”

Flujo básico

Acción que realiza el usuario	Acción que realiza el sistema
1. Ingresa al menú principal y solicita la opción “Venta de vehículos”.	2. Muestra pantalla con los siguientes campos de captura: • NIV • NRPV Muestra la opción • Buscar • Venta de Vehículo (Continúa con el flujo AO01)
3. Ingresa alguno de los siguientes campos: • NIV • NRPV Y selecciona la opción Buscar	4. Valida la información capturada y muestra: Detalle del aviso de venta: • Fecha de movimiento • Tipo de movimiento (1 = venta) • Fecha de entrega o recepción • Factura (archivo adjunto) • NIV • NRPV • Número de Placa • Modelo • Marca • Nombre completo del dueño • CURP • RFC • Teléfono Muestra las siguientes opciones: • Actualización (Continúa con el flujo AO02) • Cancelación de venta (Continúa con el flujo AO03) Si el NIV o NRPV son inválidos muestra mensaje de error.
Fin del flujo básico	

Caso de estudio “Administración de avisos de venta de vehículos”

Flujo alterno AO01: Registrar aviso de venta

Acción que realiza el usuario	Acción que realiza el sistema
	1. Valida la información capturada y muestra la siguiente: Detalle del vehículo: • NIV • NRPV • Número de Placa • Modelo • Marca • Nombre completo del dueño • CURP • RFC • Teléfono Muestra los siguientes campos: • Fecha de movimiento (otorgado por el sistema) • Tipo de movimiento (otorgado por el sistema 1 = venta) • Fecha de entrega o recepción(campo de captura) • Factura (campo de selección de archivo) Muestra las siguientes opciones: • Registrar • Cancelar • Si el NIV o NRPV son inválidos muestra mensaje de error.
2. Ingresa la fecha y selecciona la factura para adjuntar. Selecciona la opción Registrar.	3. Valida que la fecha de entrega sea posterior a la fecha actual y que la factura esté en formato PDF. Si cumple con las validaciones guarda la información del aviso y muestra mensaje de confirmación, en caso contrario muestra un mensaje de error.
Fin del flujo alterno AO01	

Caso de estudio “Administración de avisos de venta de vehículos”

Flujo alterno AO02: Actualización de avisos de venta de vehículos

Acción que realiza el usuario	Acción que realiza el sistema
	1. Muestra pantalla con los siguientes campos: • Fecha de movimiento (otorgado por el sistema = fecha actual) • Tipo de movimiento (otorgado por el sistema 1 = venta) • Fecha de entrega o recepción (campo habilitado) • Factura (campo de selección de archivo) Muestra las siguientes opciones: • Actualizar • Cancelar
2. Modifica la fecha de entrega y/o selecciona la factura para adjuntar. Selecciona la opción Actualizar.	3. Valida que la fecha de entrega sea posterior a la fecha actual y que la factura esté en formato PDF. Si cumple con las validaciones actualiza la información del aviso y muestra mensaje de confirmación, en caso contrario muestra un mensaje de error.
Fin del flujo alterno AO02	

Caso de estudio “Administración de avisos de venta de vehículos”

Flujo alterno AO03: Cancelación de avisos de venta de vehículos

Acción que realiza el usuario	Acción que realiza el sistema
	1. Muestra el siguiente mensaje: “¿Deseas cancelar el aviso de venta del vehículo con NIV: NIV del vehículo?” Muestra las siguientes opciones: • Aceptar • Cancelar
2. Selecciona la opción Aceptar	3. Elimina el aviso de venta del vehículo y muestra el siguiente mensaje: “aviso de venta del vehículo con NIV: NIV del vehículo ha sido cancelado”
Fin del flujo alterno AO03	

Agenda

I. Caso de uso: Iniciar Sesión

II. Medición Caso de uso: Iniciar Sesión

III. Caso de estudio “Administración de avisos de venta de vehículos”

IV. Medición Caso de uso: Administración de avisos de venta de vehículos

V. Medición Caso de estudio: Compra-Venta de Autos

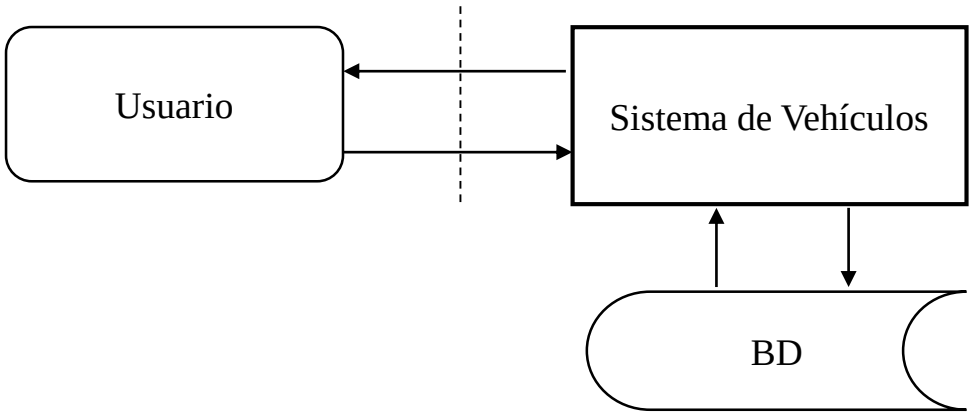
Medición Administración de avisos de venta de vehículos

Fase de Estrategia de Medición

- Propósito: determinar el tamaño funcional del caso de uso Administrar avisos de venta de vehículos como propósito de ejemplo.
- Alcance: global ya que la medición considera toda la funcionalidad del caso de uso.
- Identificación de capas: se considera una sola capa global de aplicación para la medición del caso de uso.
- Identificación de usuarios funcionales: se identifica un solo usuario funcional genérico, el cual interactúa con el sistema.

Medición Administración de avisos de venta de vehículos

- Nivel de granularidad: de las descripciones textuales del caso de uso, se identifica un nivel de granularidad adecuado para realizar la medición.
- Diagrama de contexto:



Medición Administración de avisos de venta de vehículos

Fase de Representación

- Identificación de procesos funcionales: se identifican cuatro eventos desencadenantes donde un usuario desea buscar, registrar, modificar y cancelar un aviso de venta de vehículo, identificando cuatro procesos funcionales que atienden a cada evento desencadenante.

Evento desencadenante	Proceso funcional
Usuario desea consultar la información de un aviso de venta	Buscar aviso de venta
Usuario desea registrar un aviso de venta	Registrar aviso de venta
Usuario desea actualizar la información de un aviso de venta	Actualizar aviso de venta
Usuario desea cancelar un aviso de venta	Cancelar aviso de venta

Medición Administración de avisos de venta de vehículos

Fase de Representación

- Identificación de objetos de interés y grupos de datos: De las descripciones textuales del caso de uso: Administrar avisos de venta de vehículos, se identifican los siguientes objetos de interés y grupos de datos.

Objeto de Interés	Grupo de Datos
Vehículo	Datos de Vehículo
Aviso de Venta	Datos de Aviso de Venta
Factura	Datos de Factura

Medición Administración de avisos de venta de vehículos

Identificación de Movimientos de Datos y Fase de Medición

Proceso Funcional	Evento Desencadenante	Descripción de Subprocesos	Grupo de Datos	Mov. de datos	CFP
Buscar aviso de venta	Usuario desea consultar la información de un aviso de venta	Usuario ingresa al menú principal y solicita la opción “Venta de vehículos”	N/A	N/A	0
		Sistema muestra pantalla con los siguientes campos de captura: •NIV •NRPV	N/A	N/A	0
		Usuario captura NIV o NRPV y selecciona Buscar	Vehículo	E	1
		Sistema valida la información capturada y muestra el detalle del aviso de venta, del vehículo y del dueño del vehículo.	Aviso de Venta, Factura	R	2
			Aviso de Venta, Factura	X	2
		Si el NIV o NRPV son inválidos el sistema muestra mensaje de error.	Mensajes de error	X	1
			Total:	6 CFP v4.0.1	

Medición Administración de avisos de venta de vehículos

Identificación de Movimientos de Datos y Fase de Medición

Proceso Funcional	Evento Desencadenante	Descripción de Subprocesos	Grupo de Datos	Mov. de datos	CFP
Registrar aviso de venta	Usuario desea registrar un aviso de venta	Usuario ingresa al menú principal	N/A	N/A	0
		Sistema muestra pantalla con los siguientes campos de captura: •NIV •NRPV	N/A	N/A	0
		Usuario captura NIV o NRPV y selecciona Venta de Vehículos	Vehículo	E	1
		Sistema valida la información capturada y muestra el detalle del vehículo y del dueño del vehículo.	Vehículo	R	1
			Vehículo	X	1

Facultad de Ciencias

Medición Administración de avisos de venta de vehículos

Proceso Funcional	Evento Desencadenante	Descripción de Subprocesos	Grupo de Datos	Mov. de datos	CFP
		Sistema Muestra los campos: •Fecha de movimiento (otorgado por el sistema) •Tipo de movimiento (otorgado por el sistema) •Fecha de entrega o recepción (campo de captura) •Factura (campo de selección de archivo)	Datos de Aviso de Venta	X	1
		Usuario ingresa la fecha y selecciona la factura para adjuntar. Selecciona la opción Registrar.	Aviso de Venta, Factura	E	2
		Sistema valida que la fecha de entrega sea posterior a la fecha actual y que la factura esté en formato PDF. Si cumple guarda la información del aviso.	Aviso de Venta, Factura	W	2
		Si el NIV, NRPV, fecha o el PDF son inválidos, muestra mensaje de error.	Mensajes de error	X	1
				Total:	9 CFP v4.0.1

Medición Administración de avisos de venta de vehículos

Identificación de Movimientos de Datos y Fase de Medición

Proceso Funcional	Evento Desencadenante	Descripción de Subprocesos	Grupo de Datos	Mov. de datos	CFP
Actualizar aviso de venta	Usuario desea actualizar un aviso de venta	Usuario selecciona la opción Actualización	Comando	E	1
		Sistema Muestra los campos: •Fecha de movimiento (otorgado por el sistema) •Tipo de movimiento (otorgado por el sistema) •Fecha de entrega o recepción (campo habilitado) •Factura (campo de selección de archivo)	Aviso de Venta	X	1
		Usuario ingresa la fecha y/o selecciona la factura para adjuntar. Selecciona la opción Actualizar.	Aviso de Venta, Factura	E	2

Medición Administración de avisos de venta de vehículos

Proceso Funcional	Evento Desencadenante	Descripción de Subprocesos	Grupo de Datos	Mov. de datos	CFP
		Sistema valida que la fecha de entrega sea posterior a la fecha actual y que la factura esté en formato PDF. Si cumple guarda la información del aviso.	Aviso de Venta, Factura	W	2
		Si el NIV, NRPV, fecha o el PDF son inválidos, muestra mensaje de error.	Mensajes de error	X	1
			Total:	7 CFP v4.0.1	

Medición Administración de avisos de venta de vehículos

Identificación de Movimientos de Datos y Fase de Medición

Proceso Funcional	Evento Desencadenante	Descripción de Subprocesos	Grupo de Datos	Mov. de datos	CFP
Cancelar aviso de venta	Usuario desea cancelar un aviso de venta	Usuario selecciona la opción Cancelar aviso de venta	Comando	E	1
		Sistema muestra el siguiente mensaje: "¿Deseas cancelar el aviso de venta del vehículo con NIV: NIV del vehículo?"	Datos de Vehículo	X	1
		Usuario selecciona aceptar	N/A	N/A	0
		Sistema elimina el aviso de venta del vehículo	Datos de Aviso de Venta	W	1
		Sistema muestra el siguiente mensaje: "aviso de venta del vehículo con NIV: NIV del vehículo ha sido cancelado"	Datos de Vehículo	X	1
				Total:	4 CFP v4.0.1

Resumen de medición

Total de movimientos de datos, total por procesos funcionales y total por casos de uso.

Caso de Uso	Proceso Funcional	Entradas	Salidas	Lecturas	Escrituras	Total por PF	Total por CU
Iniciar Sesión	Iniciar Sesión	1	1	1	0	3	3
Administrar avisos de ventas	Buscar aviso de venta	1	3	2	0	6	26
	Registrar aviso de venta	3	3	1	2	9	
	Actualizar aviso de venta	3	2	0	2	7	
	Cancelar aviso de venta	1	2	0	1	4	
Total		9	11	3	5	29	29

Facultad de Ciencias

Agenda

I. Caso de uso: Iniciar Sesión
II. Medición Caso de uso: Iniciar Sesión
III. Caso de estudio “Administración de avisos de venta de vehículos”
IV. Medición Caso de uso: Administración de avisos de venta de vehículos
V. Medición Caso de estudio: Compra-Venta de Autos

Medición Compra-Venta de Autos

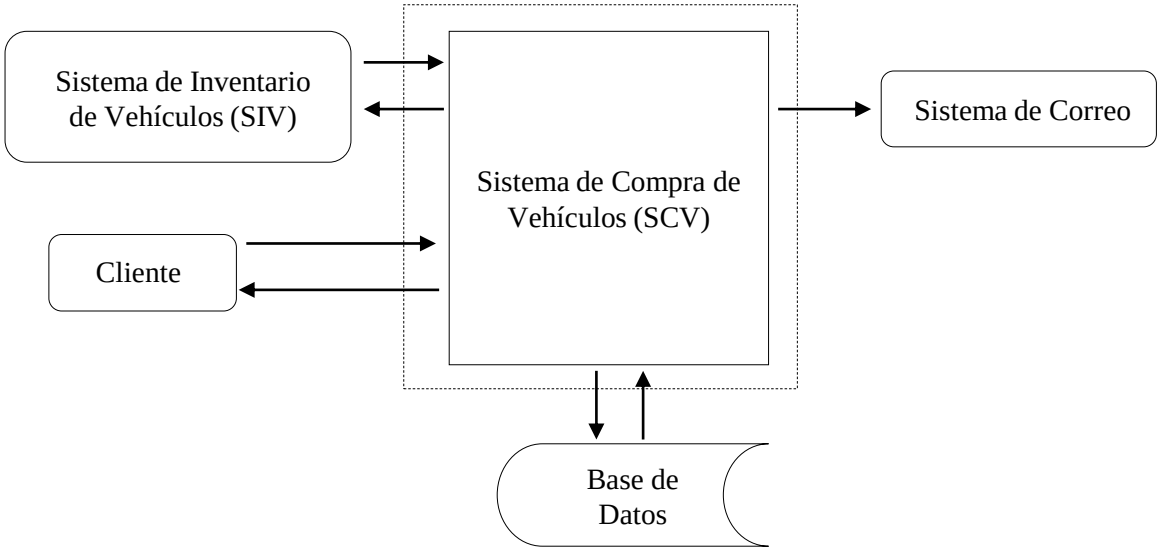
Fase de Estrategia de Medición

- Propósito: determinar el tamaño funcional para posteriormente calcular el esfuerzo y costo del proyecto.
- Alcance: global, ya que la medición considera toda la funcionalidad descrita en los requerimientos.
- Identificación de capas: se considera una sola capa global de aplicación.
- Identificación de usuarios funcionales:
  - Cliente (envía y recibe información)
  - SIV (envía y recibe información)
  - Sistema de correo (recibe información)

Nota: Los requerimientos del caso de estudio de Compra-Venta de Autos se detallaron en el ejemplo de medición IFPUG.

Medición Compra-Venta de Autos

- Nivel de granularidad: de las descripciones textuales del caso de uso, se identifica un nivel de granularidad adecuado para realizar la medición.
- Diagrama de contexto:



Medición Compra-Venta de Autos

Fase de Representación

- Identificación de procesos funcionales: se identifican los siguientes eventos desencadenantes y procesos funcionales:

Evento desencadenante	Proceso funcional
Cliente desea consultar listado vehículos	Consultar lista de vehículos
Cliente desea ver el detalle de un vehículo	Ver detalle de vehículo
Cliente desea comprar un vehículo	Comprar vehículo

- Identificación de objetos de interés y grupos de datos: se identifican los siguientes objetos de interés y grupos de datos:

Objeto de Interés	Grupo de Datos
Vehículo	Datos de Vehículo
Aviso de Venta	Datos de Aviso de Venta
Factura	Datos de Factura

Medición Compra-Venta de Autos

Identificación de Movimientos de Datos y Fase de Medición

Proceso Funcional	Evento Desencadenante	Descripción de Subprocesos	Grupo de Datos	Mov. de datos	CFP
Consultar lista de vehículos	Cliente desea consultar listado vehículos	Cliente accede a la página y la opción de búsqueda	Comando	E	1
		Sistema solicita información a SIV para llenar filtros de búsqueda (drop-downs)	Vehículo	X	1
		Sistema recibe la información de SIV para llenar filtros de búsqueda (drop-downs)	Año/Modelo, Marca/Modelo, Accesorio, Rango precio	E	4
		Sistema muestra los filtros de búsqueda (drop-downs)	Año/Modelo, Marca/Modelo, Accesorio, Rango precio	X	4

Medición Compra-Venta de Autos

Proceso Funcional	Evento Desencadenante	Descripción de Subprocesos	Grupo de Datos	Mov. de datos	CFP
		Cliente selecciona los filtros para buscar un vehículo, sino selecciona nada se buscan todos los vehículos, presiona el botón “buscar”	Parámetros de Vehículo	E	1
		Sistema envía los filtros seleccionados a SIV.	Parámetros de Vehículo	X	1
		Sistema recibe de SIV la lista de vehículos que cumplen con los filtros seleccionados.	Vehículo	E	1
		Sistema muestra lista de vehículos que cumplen con los filtros seleccionados.	Vehículo	X	1
		Si no se encuentran resultados, el sistema muestra “No hay vehículos con esas características”.	Mensaje de error	X	1
				Total:	15 CFP v4.0.1

Medición Compra-Venta de Autos

Identificación de Movimientos de Datos y Fase de Medición

Proceso Funcional	Evento Desencadenante	Descripción de Subprocesos	Grupo de Datos	Mov. de datos	CFP
Ver detalle de vehículo	Cliente desea ver el detalle de un vehículo	Cliente selecciona vehículo y presiona “Ver Detalle”	Vehículo	E	1
		Sistema solicita a SIV el detalle del vehículo seleccionado	Vehículo	X	1
		Sistema recibe la información de SIV del vehículo	Vehículo, Accesorio	E	2
		Sistema muestra el detalle del vehículo seleccionado	Vehículo, Accesorio	X	2
				Total:	6 CFP v4.0.1

Medición Compra-Venta de Autos

Identificación de Movimientos de Datos y Fase de Medición

Proceso Funcional	Evento Desencadenante	Descripción de Subprocesos	Grupo de Datos	Mov. de datos	CFP
Comprar Vehículo	Cliente desea comprar un vehículo	Cliente selecciona la opción “Comprar”	N/A	N/A	0
		Sistema muestra el formulario de captura de compra	N/A	N/A	0
		Cliente captura los detalles de la compra y selecciona “Comprar”	Compra	E	1
		Sistema envía información a SIV para que el auto quede como “comprado”	Compra	X	1
		Sistema guarda la información de compra en el almacenamiento de compras.	Compra	W	1
		Sistema envía correo de confirmación al comprador.	Correo de Compra	X	1
				Total:	6 CFP v4.0.1

Resumen de medición

Total de movimientos de datos, total por procesos funcionales.

Caso de Estudio	Proceso Funcional	Entradas	Salidas	Lecturas	Escrituras	Total por PF	Total por CU
Compra-Venta de Autos	Consultar lista de vehículos	7	8	0	0	15	25
	Ver detalle de vehículo	3	3	0	0	6	
	Comprar Vehículo	1	2	0	1	4	
Total		11	13	0	1	25	25

Facultad de Ciencias

Contacto

- Jesús Iván Saavedra Martínez
- jism@ciencias.unam.mx

Métricas de Software

Aproximación de Tamaño Funcional

Jesús Iván Saavedra Martínez

Octubre 2017

Facultad de Ciencias

Agenda

I. Introducción a la Aproximación de Tamaño Funcional

II. Escalamiento y Clasificación

III. Métodos de Aproximación

IV. Ejemplos de Casos de Estudio

V. ISBSG

Aproximación de Tamaño Funcional

Reglas – Nivel de Granularidad de un Proceso Funcional

Una medición precisa de tamaño funcional de una pieza de software requiere que sus FUR sean conocidos a nivel de granularidad en la que se pueden identificar sus procesos funcionales y sub-procesos de movimiento de datos.

- Si algunos requisitos deben medirse antes de que se hayan definido con suficiente detalle para una medición precisa, los requisitos se pueden medir utilizando un enfoque de aproximación.
- Estos enfoques definen cómo los requisitos pueden ser medidos en niveles de granularidad superiores.
- Los factores de escala se aplican entonces a las mediciones realizadas a niveles de granularidad más altos para producir un tamaño aproximado al nivel de granularidad de procesos funcionales y sus subprocesos de movimientos de datos

Aproximación de Tamaño Funcional

- Razones para aproximar el tamaño funcional:
  - Velocidad
  - Tiempo en el proyecto

- Un método de medición lo que hace es contar, mientras que un enfoque de aproximación calcula o una combinación entre calcular y juzgar.
- En la siguiente imagen, ¿cuántas personas hay en el estadio?


TODO CON MEDIDA
Corona
www.estadiosaztecа.com.mx
SOPORTES PARA MOTOR
Office Depot

Facultad de Ciencias

¿Cuantas personas hay?

Estadio Azteca

Contar	Calcular	Juzgar
Tickets Escaneados: 41,392 Palcos 24,802 Staff 1,593 **67,787 personas**	Anillo arriba 5 secciones x 1,500 asientos Anillo medio 16 secciones x 2,000 asientos Palcos 25,000 asientos **64,500 personas**	Capacidad: 114,500 45% lleno **51,500 personas**

Contar, Calcular, Juzgar

- COSMIC Standard — Cuenta
- Enfoques de aproximación comunes — Calcula
- Nuevos enfoques — Juzga y Calcula

Facultad de Ciencias

Agenda

I. Introducción a la Aproximación de Tamaño Funcional

II. Escalamiento y Clasificación

III. Métodos de Aproximación

IV. Ejemplos de Casos de Estudio

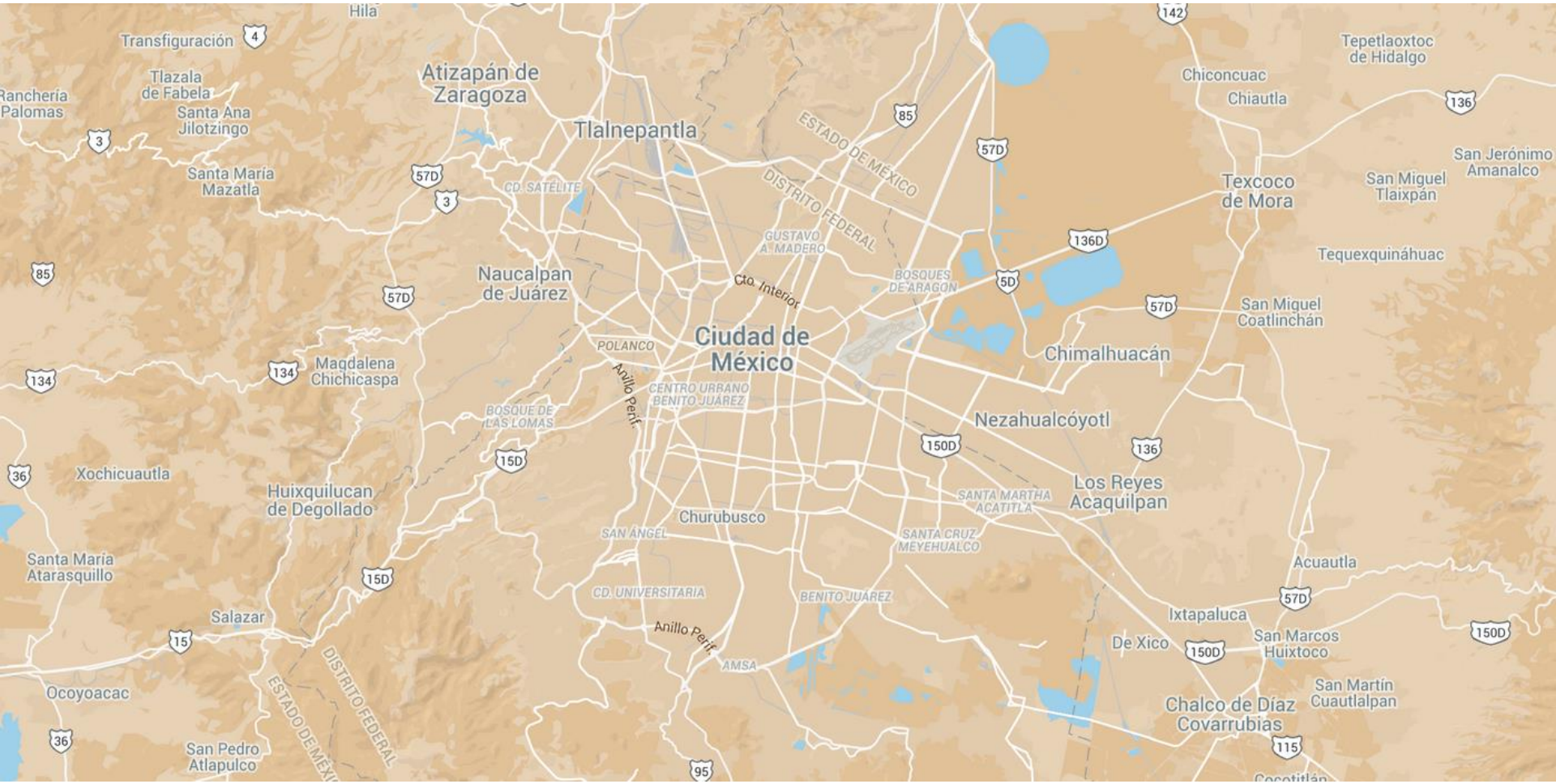
V. ISBSG

Facultad de Ciencias

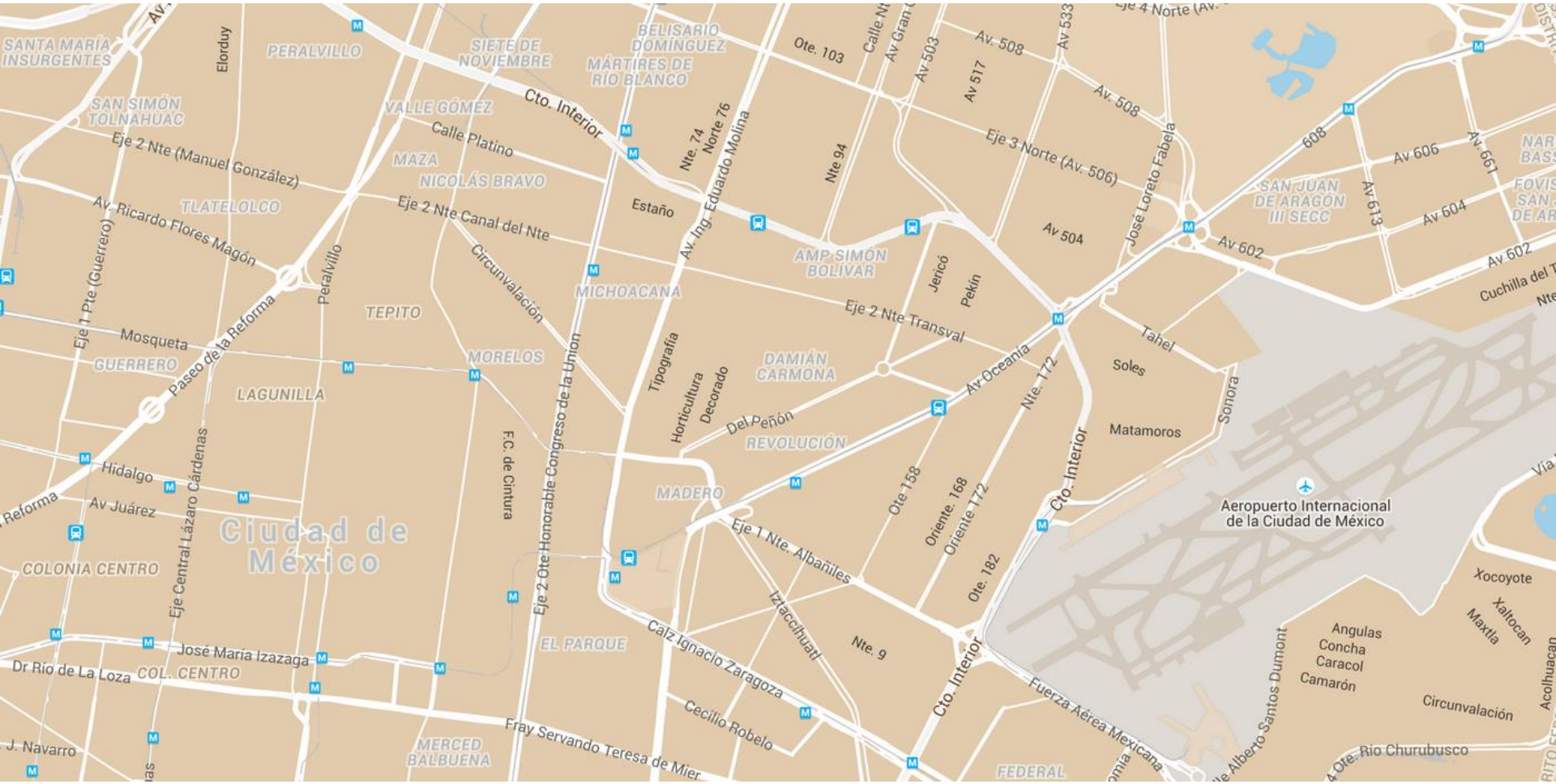
Escalamiento – overview

Facultad de Ciencias

Escalamiento – Alto Nivel



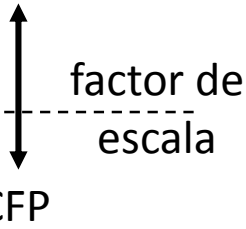
Escalamiento – zooming-in



Escalamiento – detalles

Escalamiento – Nivel de Granularidad

Nivel de granularidad de los Requerimientos Funcionales de Usuario	Método de Medición	Estándar de Medición
Requisitos en un alto nivel de granularidad derivados de por ejemplo: • alto nivel en la descripción de los requisitos de software • artefactos de arquitectura • vista de alto nivel del software existente Expresados en unidades locales definidas, como casos de uso o en CFP	'Enfoque de aproximación' al método de medición COSMIC. Calibrado localmente	El tamaño promedio de la unidad contable definida localmente, expresada en unidades locales o en CFP
El nivel de granularidad del proceso funcional	Método de Medición COSMIC	CFP



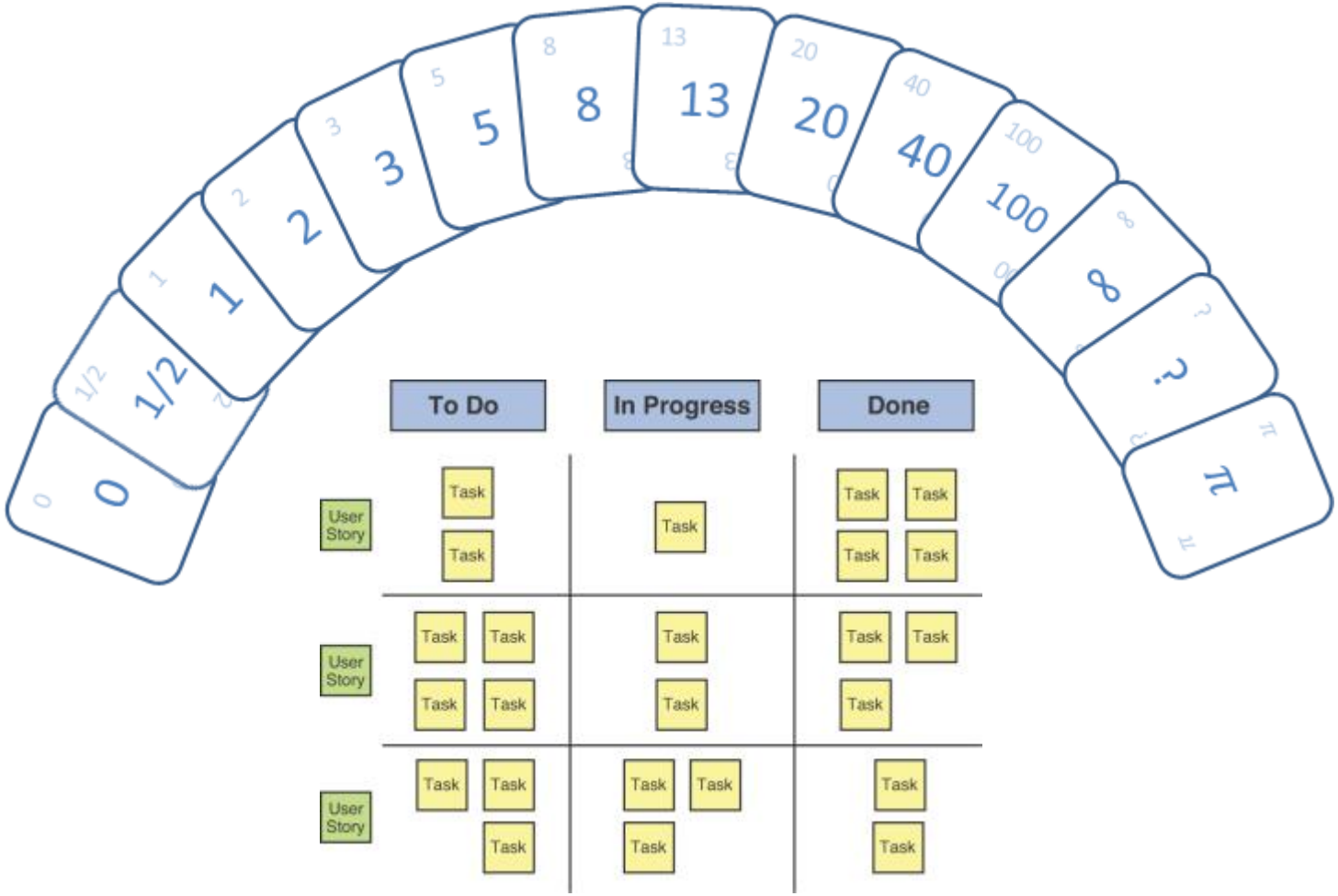
Clasificación

M

L

S

Clasificación



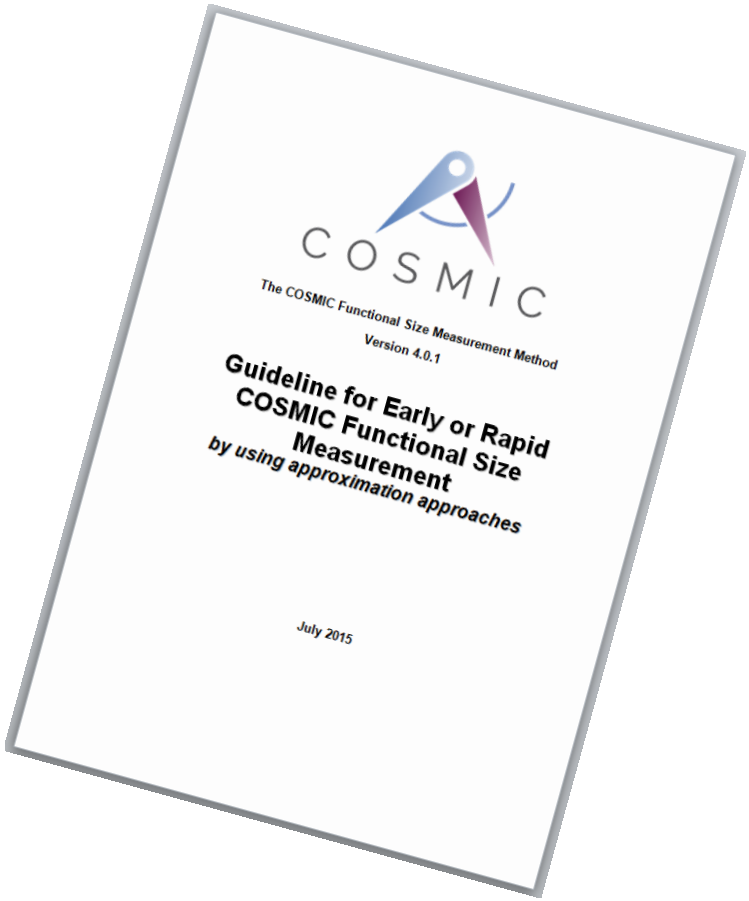
Facultad de Ciencias

Agenda

I. Introducción a la Aproximación de Tamaño Funcional
II. Escalamiento y Clasificación
III. Métodos de Aproximación
IV. Ejemplos de Casos de Estudio
V. ISBSG

Métodos de Aproximación

- Average functional process
- Fixed size classification
- Equal size bands
- Average use case
- Early & Quick
- EASY
- Textual requirements
- EPCU



Average functional process

Escala de Proceso Funcional a CFP

4,880 CFP

610 Procesos Funcionales

8 CFP/Proceso Funcional

Fixed size classification

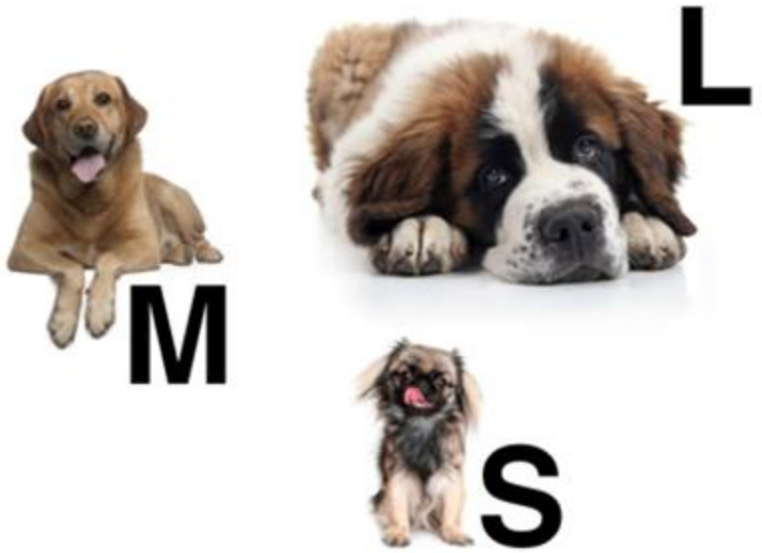
Asignar una categoría a un proceso funcional:

Small

Medium

Large

…



- Para estimar: Clasificar a cual banda pertenece un Proceso Funcional

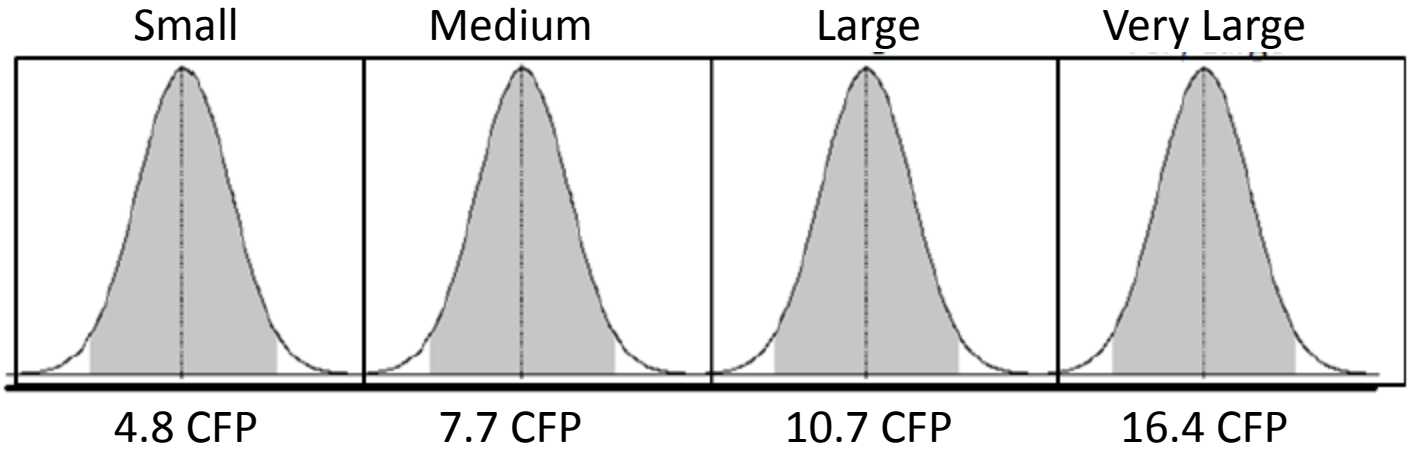
Equal size bands

1. Medir una muestra de software
2. Ordenar el tamaño de los procesos funcionales
3. Dividir el total en bandas
4. Calcular el tamaño promedio de un proceso funcional en cada banda

- Para estimar:
  Clasificar a cual banda pertenece un proceso funcional

Equal size bands

- (Vogelezang , 2007) utilize mediciones en 37 proyectos de desarrollo de aplicaciones negocios, cada uno con un tamaño total mayor a 100 CFP. Los valores de **los cuartiles** de este conjunto de datos es el siguiente: **Small** = 4.8 CFP, **Medium** =7.7 CFP, **Large** = 10.7, y **Very Large** = 16.4 CFP.



Average functional process

Escala de Caso de Uso a CFP

8 CFP/Proceso Funcional

3.5 Procesos Funcionales/Caso de Uso

28 CFP/Caso de Uso

Early & Quick

Dos niveles de clasificación

Tipo	Nivel	Rangos / Equivalencia con COSMIC	Mín CFP	Más probable	Máx CFP
Proceso Funcional	Pequeño	1-5 movimientos de datos	2	3.9	5
	Mediano	5.8 movimientos de datos	5	6.9	8
	Grande	8-14 movimientos de datos	8	10.5	14
	Muy Grande	14+ movimientos de datos	14	23.7	30
Proceso Típico	Pequeño	CRUD (procesos pequeños/medianos) CRUD + lista (procesos pequeños)	15.6	20.4	27.6
	Mediano	CRUD (procesos medianos/grandes) CRUD + lista (procesos medianos) CRUD + lista + reporte (procesos pequeños)	27.6	32.3	42
	Grande	CRUD (procesos grandes) CRUD + lista (procesos medianos/grandes) CRUD + lista + reporte (procesos medianos)	42	48.5	63
Proceso General	Pequeño	6-10 PF´s genericos	20	60	110
	Mediano	10-15 PF´s genéricos	40	95	160
	Grande	15-20 PF´s genéricos	60	130	220
Macro Proceso	Pequeño	2-4 PG´s genéricos	120	285	520
	Mediano	4-6 PG´s genéricos	240	475	780
	Grande	6-10 PG´s genéricos	360	760	1,300

EASY

Dos niveles de clasificación

Clasificación de PF	Nivel de especificación	CFP (min)	CFP	CFP (max)	CFP Aproximado	Probabilidad
PF pequeño	Poco desconocido	2 (10%)	3 (75%)	5 (15%)	3.2	>80%
PF pequeño	Desconocido (No FUR)	2 (15%)	4 (50%)	8 (35%)	5.1	<50%
PF mediano	Poco desconocido	5 (10%)	7 (75%)	10 (15%)	7.25	>80%
PF mediano	Desconocido (No FUR)	5 (15%)	8 (50%)	12 (35%)	8.95	<50%
PF grande	Poco desconocido	8 (10%)	10 (75%)	12 (15%)	10.1	>80%
PF grande	Desconocido (No FUR)	8 (15%)	10 (50%)	15 (35%)	11.45	<50%
PF complejo	Poco desconocido	10 (10%)	15 (75%)	20 (15%)	15.25	>80%
PF complejo	Desconocido (No FUR)	10 (15%)	18 (50%)	30 (35%)	21	<50%

Supuestos de los Métodos de Aproximación de Tamaño Funcional

- Cada ejemplo acerca de Enfoques de Aproximación de Tamaño está basado en dos principales supuestos:
  1. Existen datos históricos para calcular el factor de escala (promedio, o bandas de tamaño).
  2. Se describe todo el conjunto de requisitos, o al menos hay un compromiso, definido por los requisitos, sobre el alcance del software que se va a desarrollar.

Calibración Local

	Necesita calibración local	Nivel de granularidad de los requerimientos	Consideración
Average Functional Process	X	Proceso Funcional	Esta aproximación es válida mientras existan razones suficientes para suponer que la muestra en la que se calcula el tamaño del proceso funcional promedio es representativo del software cuyo tamaño funcional se aproxima
Fixed Size Classification	X	Proceso Funcional	Esta aproximación es válida mientras existan razones suficientes para suponer que la clasificación de tamaño asignada es representativa del software cuyo tamaño funcional se aproxima
Equal Size Bands	X	Proceso Funcional	Este método se recomienda para el dimensionamiento aproximado de software en el que la distribución de los tamaños de proceso funcionales es sesgada. Para aplicaciones de negocio este método tiene poco valor añadido sobre el método de Average Functional Process o Fixed Size Classification
Average Use Case	X	Caso de Uso	Esta aproximación es válida mientras existan razones suficientes para suponer que la muestra en la que se calcula el tamaño del caso de uso promedio es representativo del software cuyo tamaño funcional se aproxima
Early & Quick	X	Enfoque Multinivel	La precisión del método depende en gran medida del entrenamiento y la capacidad de los profesionales que lo utilizaron para entender las categorías en los niveles más altos de granularidad. Este enfoque de aproximación combina enfoques de escala y clasificación
Quick/Early	X	Casos de Uso	La precisión es directamente proporcional al nivel de granularidad del modelo de casos de uso analizado
EPCU		Proceso Funcional y Casos de Uso	No requiere calibración local (menos costoso) y es útil cuando no hay datos históricos disponibles

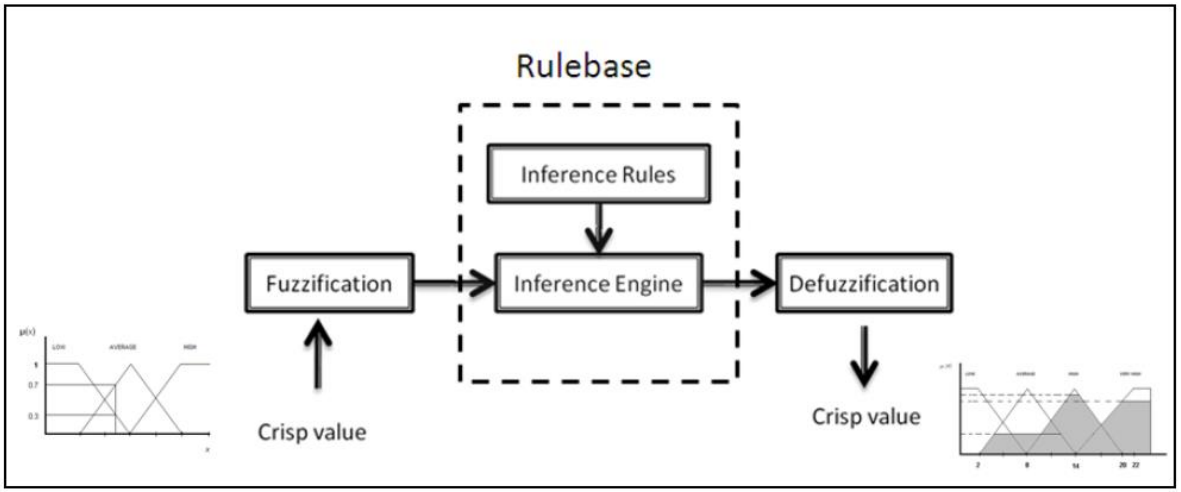
Requerimientos textuales

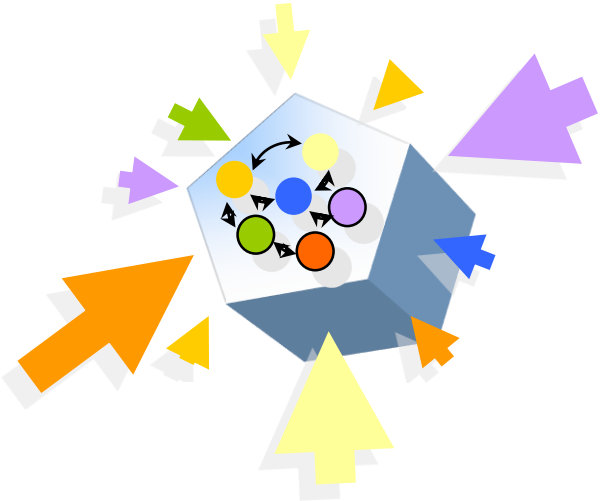
- Cuenta los requisitos informalmente escritos por proceso funcional
- Requisitos y tamaño almacenado como referencia
- Divide los procesos funcionales en conjuntos de clases de tamaños difusos

- Forma un algoritmo de clasificación de texto a las características lingüísticas del conjunto de referencia

EPCU

"La Incertidumbre: no es posible medirla, sin embargo es posible contextualizarla"





EPCU es el acrónimo de Estimation of Projects in a Context of Uncertainty

Facultad de Ciencias

Agenda

I. Introducción a la Aproximación de Tamaño Funcional
II. Escalamiento y Clasificación
III. Métodos de Aproximación
IV. Ejemplos de Casos de Estudio
V. ISBSG

Case Study: COSMIC Approximate Sizing Approach Without Using Historical Data

Francisco Valdés Souto
École de Technologie Supérieure,
francisco.valdes.1@ens.etsmtl.ca

IWSM MENSURA 2012

Alain Abran
École de Technologie Supérieure
alain.abran@etsmtl.ca

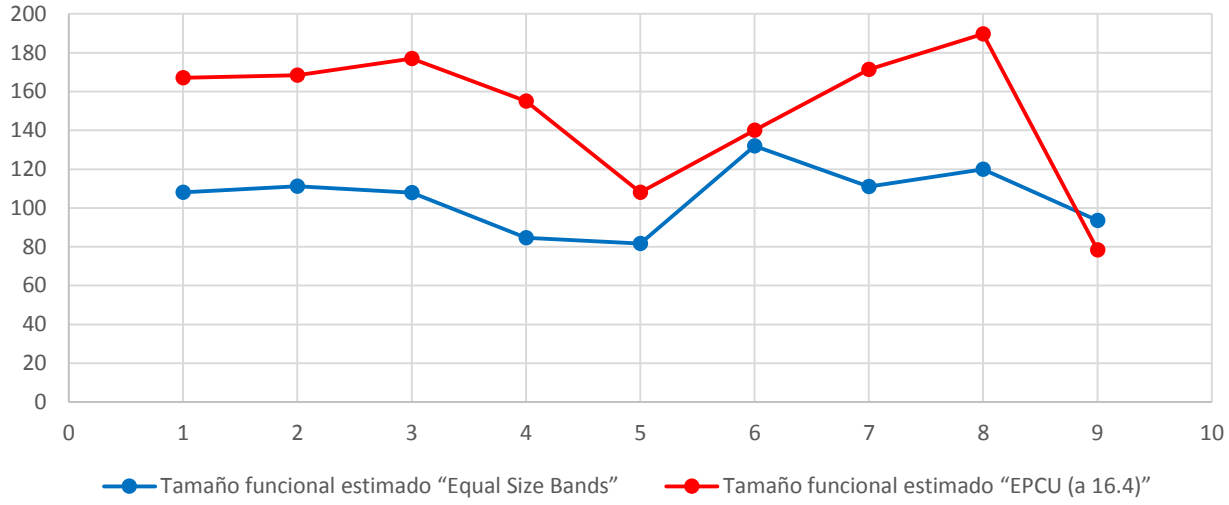
Abstract—In mature engineering disciplines, international consensus can be reached on measurement, as evidenced through established measurement standards. In software engineering, there are 5 functional size measurement standards. These standards work best when the functionality to be measured is fully known, although this usually doesn't happen in the early phases of software development.

The techniques most often used to approximate the sizing of the software to be developed in the early phases involve historical data. However, gathering historical data is a challenge in itself. This paper proposes the use of a fuzzy logic model to approximate the functional size of a piece of software.

Análisis de información

Profesional	Valor real	Tamaño funcional estimado "Equal Size Bands"	MRE	Tamaño funcional estimado "EPCU (a 16.4)"	MRE	Diferencia
1	107	108.1	1%	167.1	56%	55%
2	107	111.2	4%	168.4	57%	53%
3	107	107.9	1%	176.9	65%	64%
4	107	84.6	21%	155.1	45%	24%
5	107	81.7	24%	108.1	1%	23%
6	107	131.9	23%	140.1	31%	8%
7	107	111	4%	171.3	60%	56%
8	107	119.9	12%	189.6	77%	65%
9	107	93.5	13%	78.3	27%	14%
PROMEDIO MRE			**11%**		**47%**	

Análisis de información



- La forma de las curvas muestra un comportamiento similar.
- SDMRE con el modelo EPCU del 23% y un MMRE del 47%. Este MMRE es mayor que el del valor aproximado de las Bandas de Tamaño Igual de 36%, y el valor SDMRE también es más alto, al 14%.
- La diferencia máxima de MMRE es 65%, y la diferencia mínima es 8%.
- El contexto EPCU no tiene que ser calibrado: no utiliza bandas, sino un rango continuo en ε R.

Experimento con datos de la industria (2014)

IWSM MENSURA
WHERE ACADEMIC IDEAS MEET INDUSTRY PRACTICE ON SOFTWARE MEASUREMENT TOPICS

COSMIC Approximate Sizing Using a Fuzzy Logic Approach: An Experiment with Industry Data

Francisco Valdés Souto
École de Technologie Supérieure, University of Québec
Dept. of Software Engineering
Montréal, Canada
francisco.valdes.1@ens.etsmtl.ca

Alain Abran
École de Technologie Supérieure, University of Québec
Dept. of Software Engineering
Montréal, Canada
alain.abran@ens.etsmtl.ca

Aproximación de tamaño sin utilizar datos históricos en un proyecto real

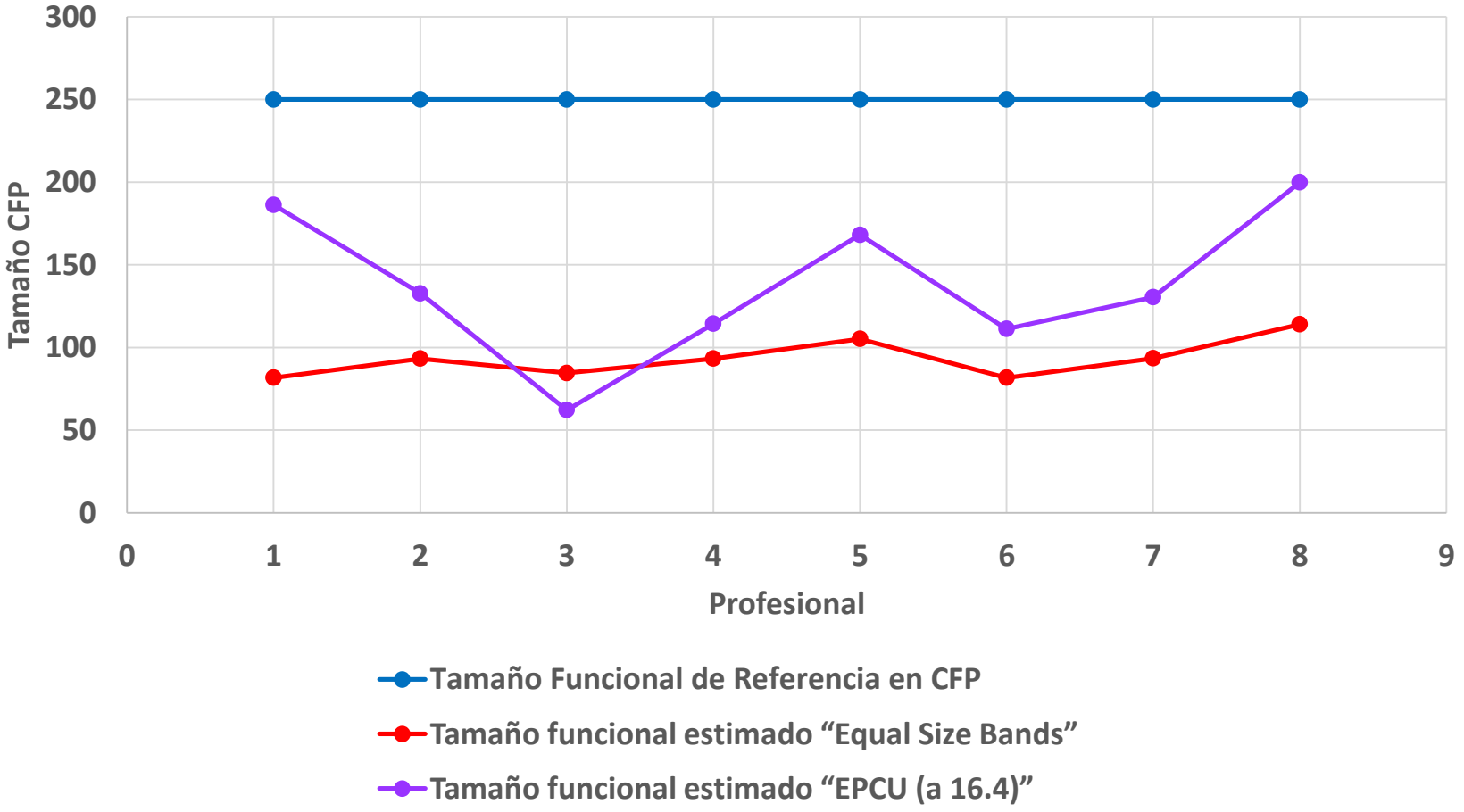
Caso de uso ID	Entradas en CFP	Salidas en CFP	Lecturas en CFP	Escrituras en CFP	Tamaño en CFP
Caso de uso 1	1	7	6	2	16
Caso de uso 2	5	18	23	9	55
Caso de uso 3	1	12	12	2	27
Caso de uso 4	1	4	2	1	8
Caso de uso 5	1	1	7	0	9
Caso de uso 6	1	2	3	0	6
Caso de uso 7	1	11	11	3	26
Caso de uso 8	1	4	3	0	8
Caso de uso 9	4	3	6	3	16
Caso de uso 10	1	1	1	1	4
Caso de uso 11	5	4	9	5	23
Caso de uso 12	3	3	7	3	16
Caso de uso 13	3	2	5	4	14
Caso de uso 14	1	7	14	0	22
Tamaño total					**250**

Aproximación de tamaño sin utilizar datos históricos en un proyecto real

Profesional	Tamaño Funcional de Referencia en CFP	Tamaño funcional estimado "Equal Size Bands"	MRE	Tamaño funcional estimado "EPCU (a 16.4)"	MRE
Profesional 1	250	81.7	67%	186.32	25%
Profesional 2	250	93.3	63%	132.76	47%
Profesional 3	250	84.6	66%	62.19	75%
Profesional 4	250	93.3	63%	114.34	54%
Profesional 5	250	105.2	58%	168.13	33%
Profesional 6	250	81.7	67%	111.26	55%
Profesional 7	250	93.5	63%	130.43	48%
Profesional 8	250	114	54%	199.82	20%
MMRE			**63%**		**45%**
SDMRE			**5%**		**18%**

Facultad de Ciencias

Aproximación de tamaño sin utilizar datos históricos en un proyecto real



Improving the COSMIC Approximate Sizing Using the Fuzzy Logic EPCU Model

Francisco Valdés Souto[1], Alain Abran[2]

[1] École de Technologie Supérieure – Université du Québec, francisco.valdes.1@ens.etsmtl.ca

[2] École de Technologie Supérieure – Université du Québec, alain.abran@ens.etsmtl.ca

Resumen. En la Ingeniería de Software, los estándares de medición de tamaño funcional requieren, para resultados de medición precisos, que la funcionalidad a ser medida sea completamente conocida. Por lo tanto, en etapas tempranas del desarrollo de software cuando hay falta de detalle, un enfoque de aproximación de tamaño es adecuado.

Mejorando la Aproximación

# de Proyectos	Sector	Rango del Proyecto	Tamaño Funcional Promedio del Proyecto	# FP	Tamaño
26	Bancario	11 – 2743	476	1345	12375
8	Gobierno	64 – 2364	481	838	3845
6	Industria	84 – 1311	551	342	3305
7	Logística	193 – 1164	538	321	3766

Caracterización del Conjunto de Datos

Quartile		% FP Incluido	Descripción	Valor Promedio
Q1	Small FP's	55%	contiene FP's de hasta 6 CFP	3.7
Q2	Medium FP's	26%	contiene FP's entre 6 - 10 CFP	7.7
Q3	Large FP's	14%	contiene FP's entre 10 - 25 CFP	14.6
Q4	Very Large FP's	5%	contiene FP's de 25 CFP y más	44.1

Q-Size considetando cuatro sectores

Facultad de Ciencias

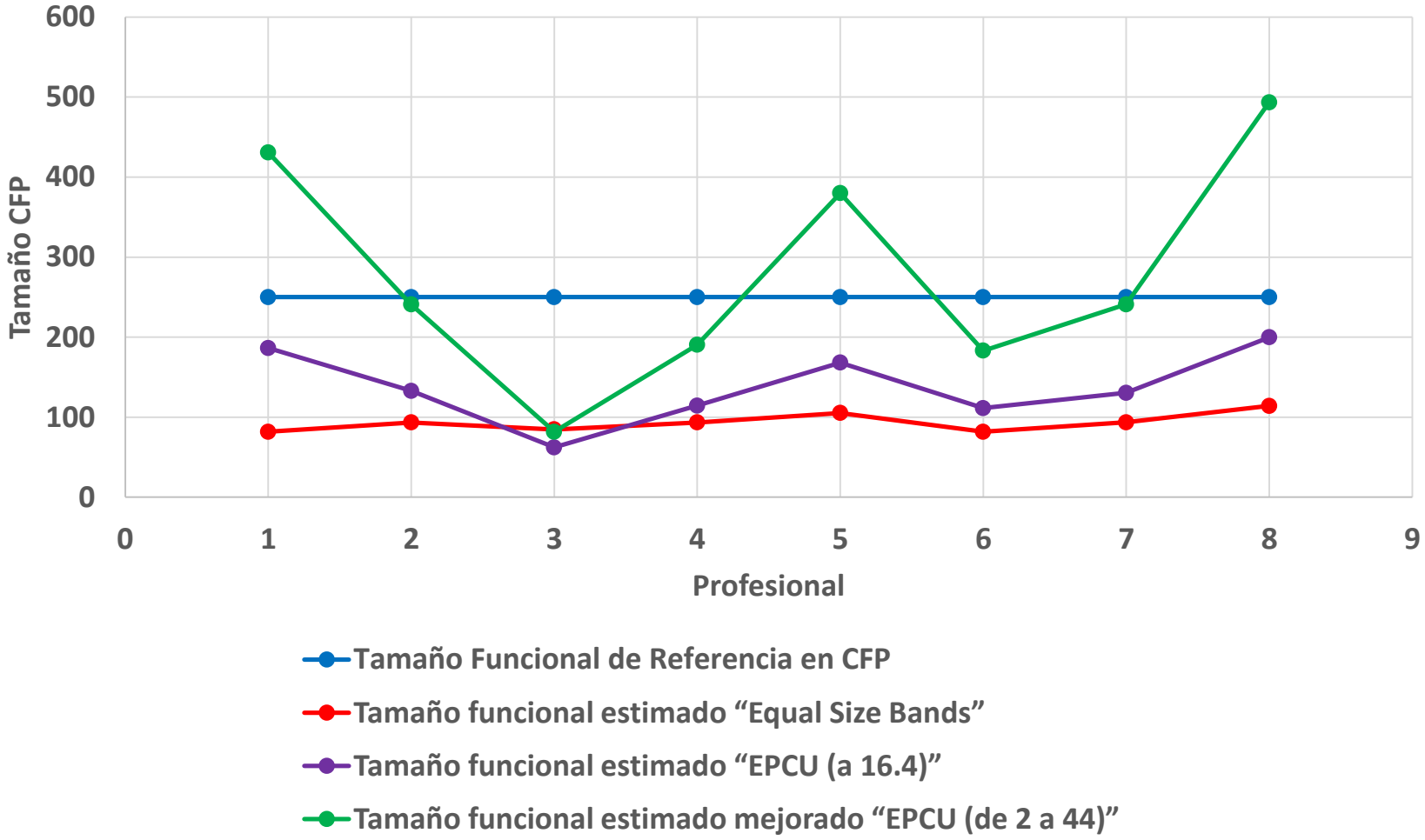
Datos y Análisis del Caso de Estudio 2014-2015

Profesional	Tamaño Funcional de Referencia en CFP	Tamaño funcional estimado "Equal Size Bands"	MRE	Tamaño funcional estimado "EPCU (de 2 a 16.4)"	MRE	Tamaño funcional estimado mejorado "EPCU (de 2 a 44)"	MRE
Profesional 1	250	81.7	67%	186.32	25%	430.76	72%
Profesional 2	250	93.3	63%	132.76	47%	240.74	4%
Profesional 3	250	84.6	66%	62.19	75%	81.65	67%
Profesional 4	250	93.3	63%	114.34	54%	190.33	24%
Profesional 5	250	105.2	58%	168.13	33%	379.86	52%
Profesional 6	250	81.7	67%	111.26	55%	183.14	27%
Profesional 7	250	93.5	63%	130.43	48%	240.91	4%
Profesional 8	250	114	54%	199.82	20%	493.47	97%
MMRE			**63%**		**45%**		**43%**
SDMRE			**5%**		**18%**		**34%**

2014 2015

Facultad de Ciencias

Datos y Análisis del Caso de Estudio 2014-2015

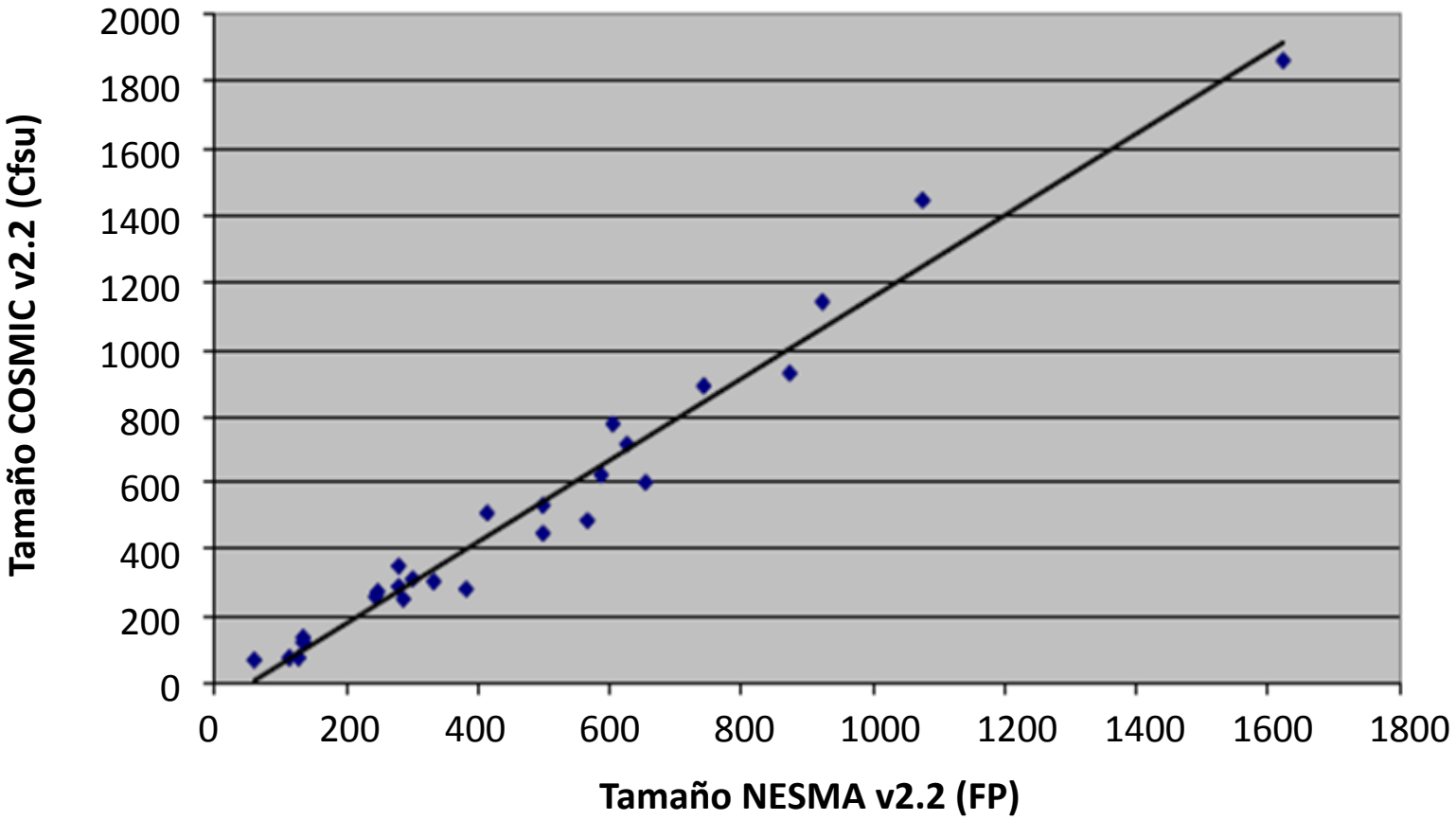


Comparabilidad de Tamaño de Mediciones

- Los métodos existentes no miden lo mismo, puede ser que matemáticamente existan fórmulas que lo relacionen, sin embargo, teóricamente hay razones para que esto no sea posible.

- Hay tres enfoques posibles de conversión:
  - Conversión Teórica, con un rango empírico de tamaño.
  - Conversión Manual
  - Conversión Estadística

Comparabilidad de Tamaño de Mediciones



Comparabilidad de Tamaño de Mediciones

Autor	No. de puntos de datos	Rango de Tamaños (FP)	Fórmula de conversión obtenida por análisis de regresión	R2
Fetke (1999)	4	40 – 77	CFP = 1.1 x FP (IFPUG) – 7.6	0.97
Vogelezang & Lesterhuis (2003)	11	39 – 1424	CFP = 1.2 x FP (NESMA) – 87	0.99
Abran, Desharnais, Azziz (2005)	6	103 – 1146	CFP = 0.84 x FP (IFPUG) +18	0.91
Desharnais & Abran (2006)	14	111 – 647	CFP = 1.0 x FP (IFPUG) – 3	0.93
Van Heeringen (2007)	26	61 – 1422	CFP = 1.22 x FP (NESMA) – 64	0.97

Facultad de Ciencias

Agenda

I. Introducción a la Aproximación de Tamaño Funcional
II. Escalamiento y Clasificación
III. Métodos de Aproximación
IV. Ejemplos de Casos de Estudio
V. ISBSG

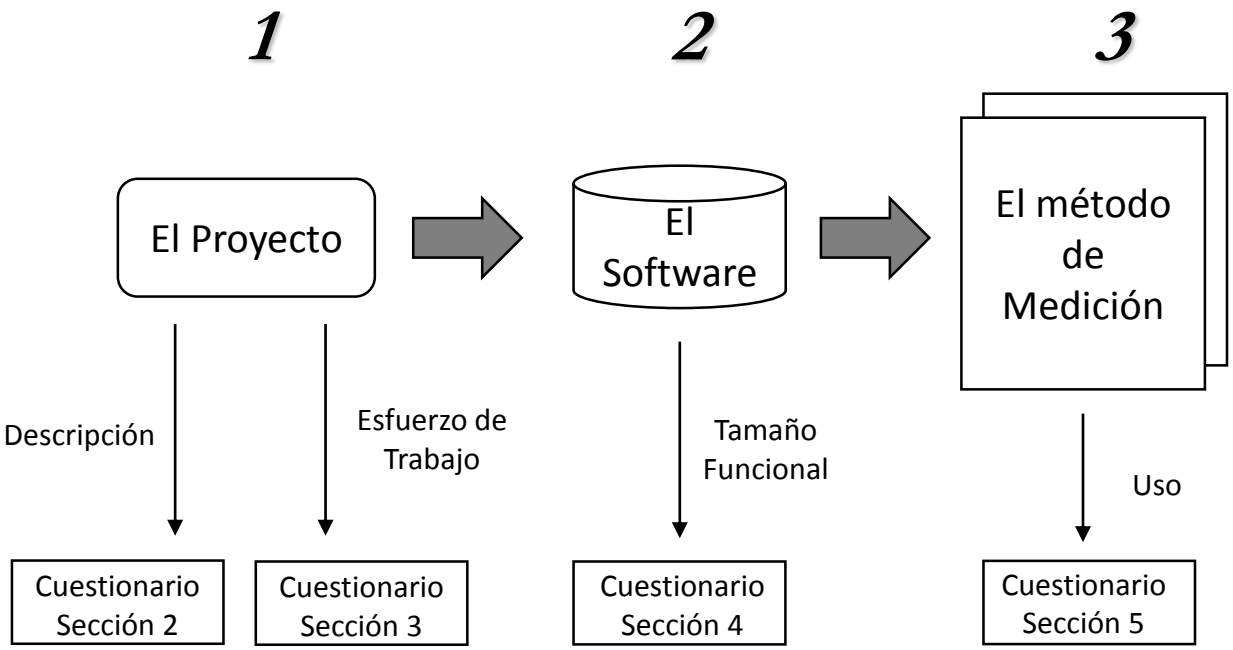
ISBSG

- **ISBSG**: International Software Benchmarking Standards Group (Non for profit)

- Se recomienda encarecidamente a las organizaciones enviar datos al Repositorio Internacional de ISBSG:
  - Directamente: www.isbsg.org
  - O a través de un miembro internacional de COSMIC para agregar otra capa para el anonimato

ISBSG Overview

Recopilación de datos estandarizada para evaluación comparativa y estimación



Cuestionario COSMIC de recolección de datos

ISBSG Datos del Proyecto

- Proyecto
  - La organización (tipo negocios),
  - Tipo de software,
  - Tipo de proyecto (desarrollo, mantenimiento, ...),
  - Desarrollo y plataforma de destino,
  - Duración
  - Fecha de puesta en marcha
- Esfuerzo
  - Método de registro de esfuerzo
  - Completitud de los datos de esfuerzo
  - Confianza en los datos de esfuerzo
  - Nivel de esfuerzo
  - Alto nivel de desglose del esfuerzo

Proceso de recopilación de datos

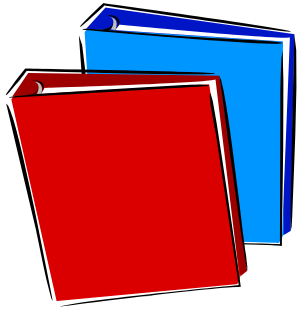
Planificación
- Adquirir compromiso
- Seleccionar proyectos

Preparación
- Preparación
- Ejercicio de repetibilidad

Recopilación de datos
- Estándares COSMIC
- Estándares ISBSG

Análisis
- Controles de calidad de medición
- Validación
- Registro de datos
- Análisis de datos
- Análisis comparativo

Reporte de análisis



Realimentación organizacional

Background

- Datos de la industria ahora disponibles en ISBSG Versión 2009:

- ~ 350 proyectos medidos por COSMIC, principalmente:
  - Aplicaciones de negocio
  - Aplicaciones de Tiempo-real
  - Componentes de Software re-utilizables

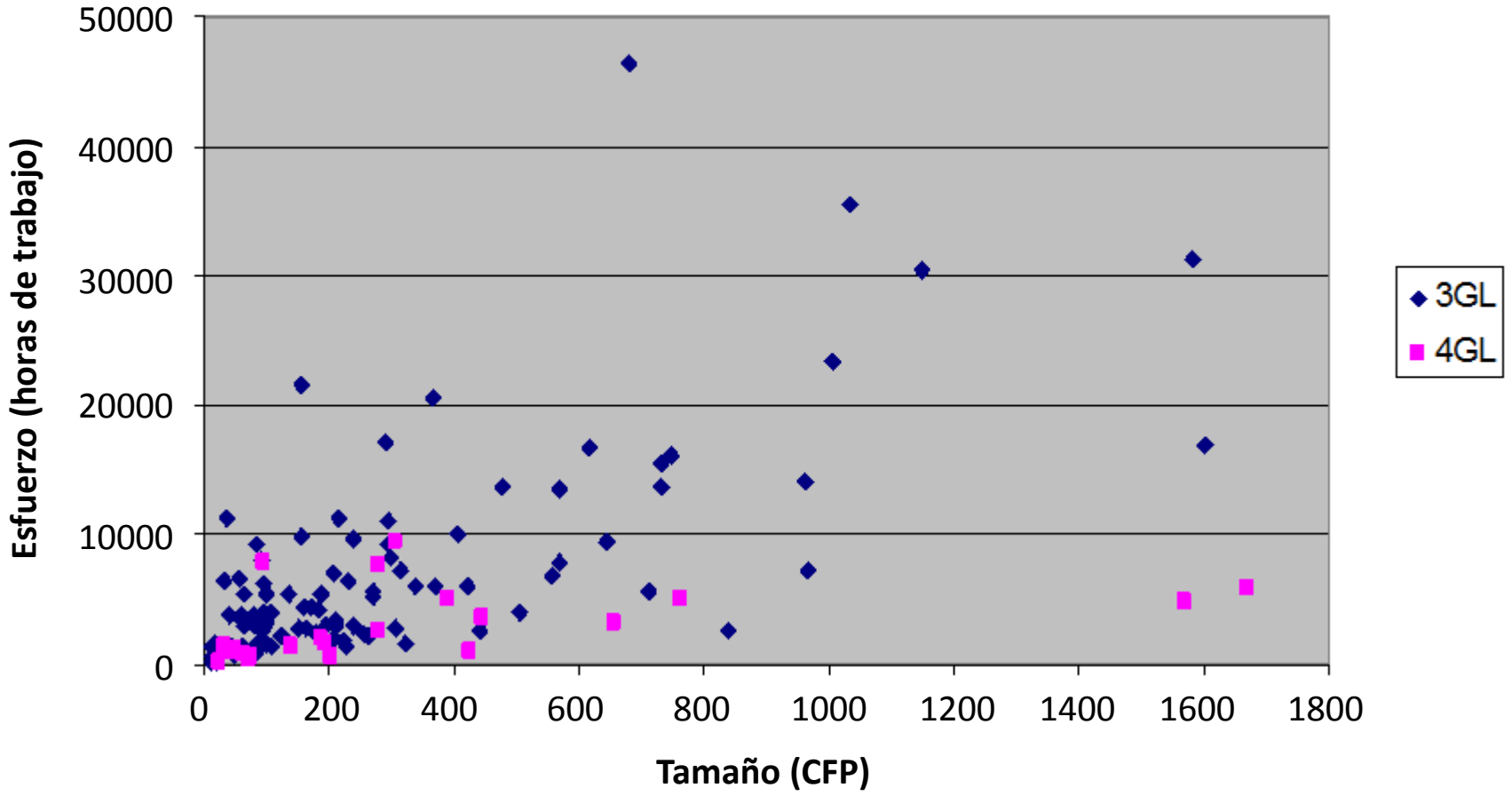
Software de Aplicaciones de Negocio

- 272 proyectos de aplicaciones empresariales
  - Sector bancario = 163 proyectos
  - Gobierno y Administración Pública = 27
  - Seguros = 24
  - Ingeniería = 14
  - Médico y Salud = 6
  - Comercio minorista y mayorista = 5

- Estos 272 proyectos:
  - 129 nuevos proyectos de desarrollo,
  - 137 proyectos de mejora y
  - 6 proyectos de re-desarrollo.

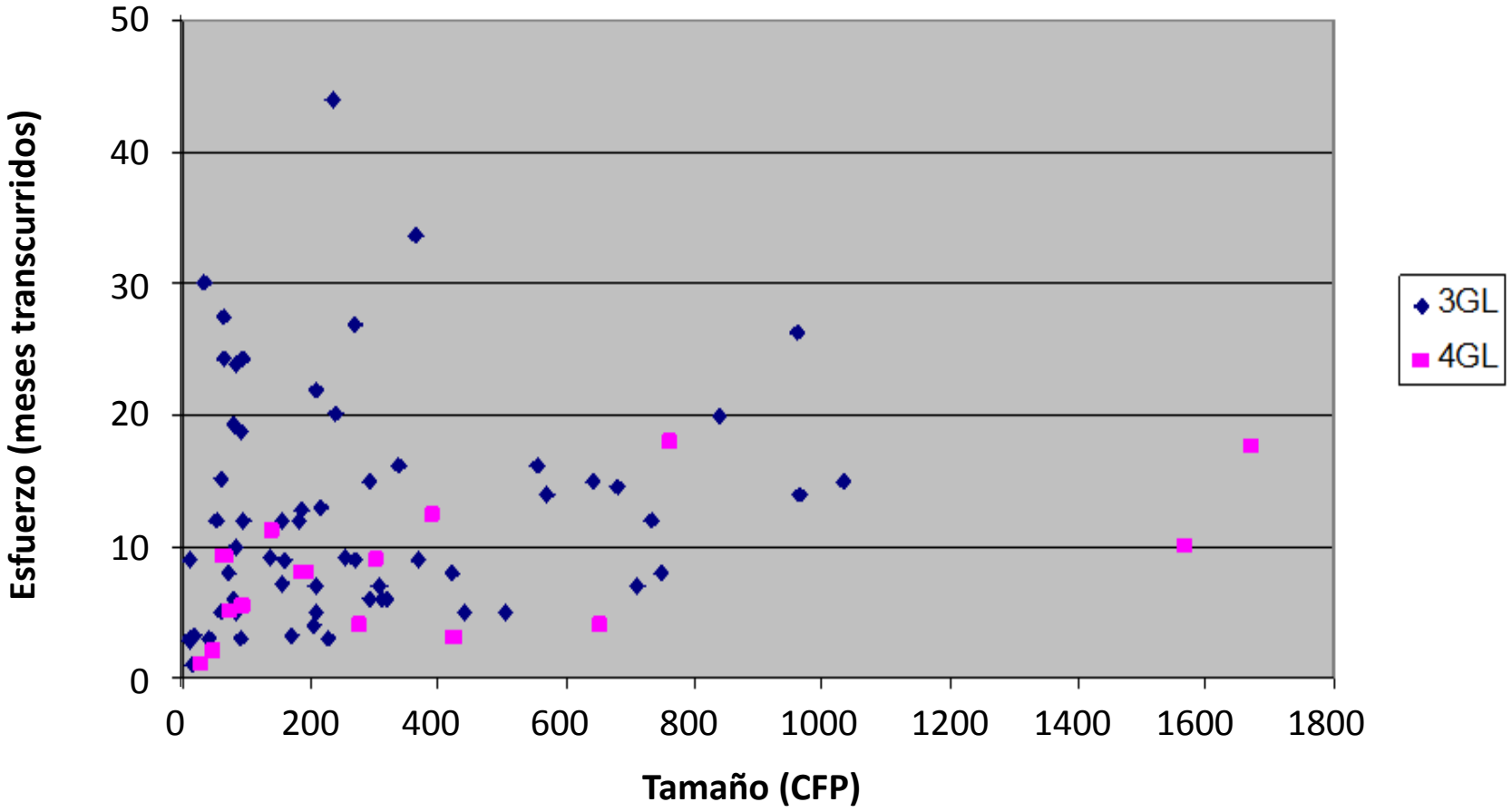
Facultad de Ciencias

Negocio - Nuevo desarrollo: Esfuerzo vs Tamaño

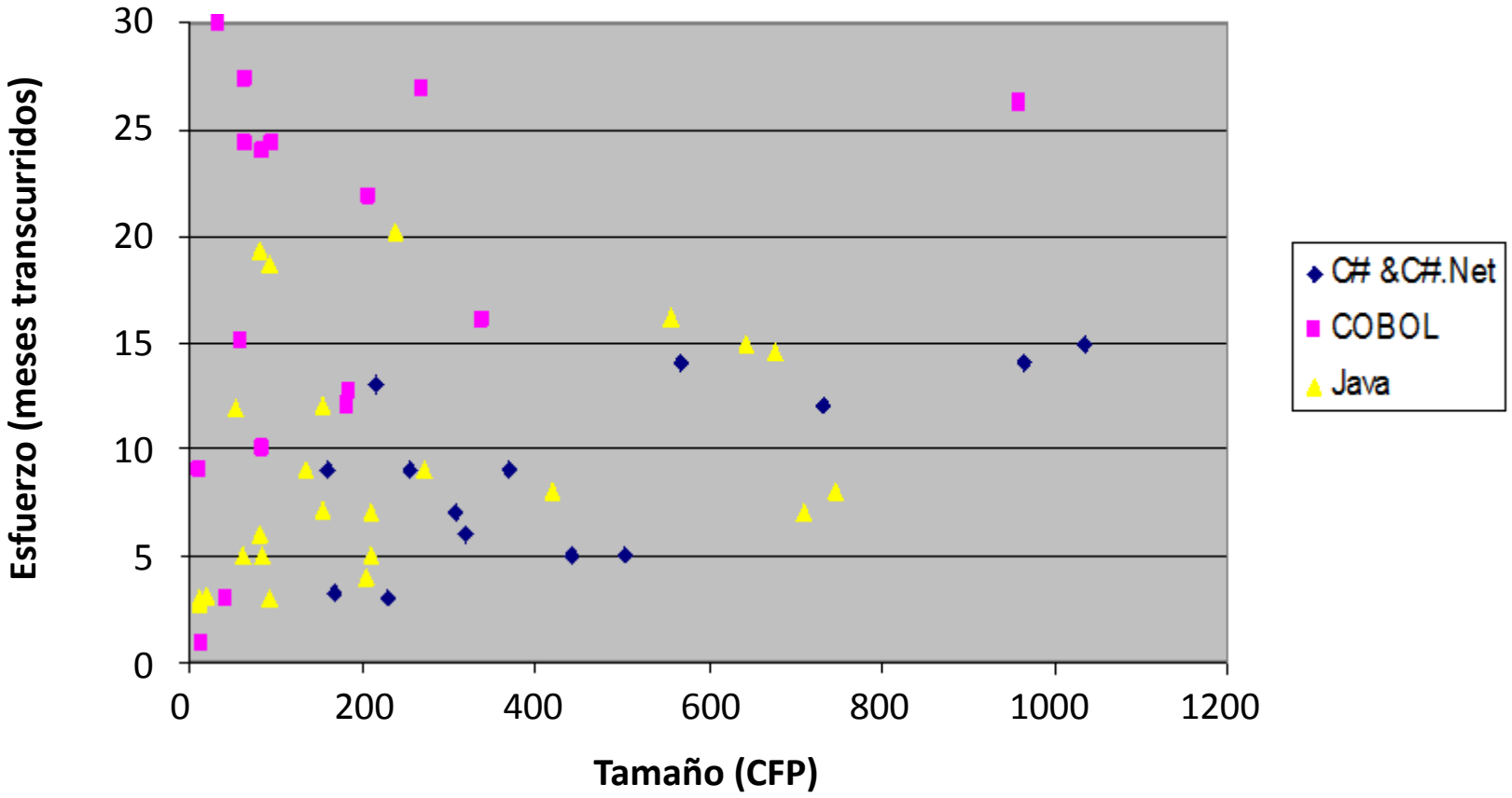


Facultad de Ciencias

Negocio - Nuevo desarrollo: Duración vs Tamaño



Negocio - Nuevo desarrollo: Duración vs Tamaño



Facultad de Ciencias

Contacto

- Jesús Iván Saavedra Martínez
- jism@ciencias.unam.mx

Métricas de Software

Ejemplos de Aproximación de Tamaño Funcional

Jesús Iván Saavedra Martínez

Octubre 2017

Facultad de Ciencias

Agenda

I. Conjunto de Datos 1 para calibración de métodos de aproximación

II. Calibración con métodos de aproximación AFP, AUC, FSC, ESB

III. Aproximación del Caso de uso: Iniciar sesión y Administración de avisos de venta de vehículos

IV. Aproximación del Caso de estudio: Compra-Venta de Autos

Facultad de Ciencias

Conjunto de Datos 1

CU	PF	CFP v4.0.1
1	1	7
	2	2
	3	3
	4	7
2	5	14
	6	5
	7	4
3	8	10
	9	17
4	10	6
5	11	7
	12	6
6	13	10
	14	10
	15	5
7	16	9
8	17	5
	18	9
	19	7
	20	8
9	21	4
	22	8
	23	8
10	24	8
	25	9

CU	PF	CFP v4.0.1
11	26	6
	27	6
	28	4
12	29	2
	30	5
	31	5
13	32	6
	33	6
	34	7
14	35	10
15	36	11
	37	2
	38	12
	39	6
16	40	3
	41	23
	42	5
17	43	5
	44	6
	45	6
	46	4
	47	3
18	48	4
	49	7
	50	5

CU	PF	CFP v4.0.1
19	51	9
	52	3
	53	13
	54	5
	55	12
	56	5
	57	6
	58	4
	59	7
	60	6
20	61	2
	62	5
	63	5
	64	5
	65	6
21	66	6
	67	12
	68	7
22	69	3
	70	5
	71	6
	72	7
	73	3
23	74	4
	75	8
	76	7

CU	PF	CFP v4.0.1
24	77	4
	78	6
25	79	15
	80	12
	81	16
	82	5
26	83	14
	84	10
27	85	7
	86	15
	87	15
	88	3
	89	8
	90	8
	91	20
28	92	25
	93	8
29	94	15
	95	2
	96	7
	97	5
30	98	14
	99	3
	100	15

Facultad de Ciencias

Agenda

I. Conjunto de Datos 1 para calibración de métodos de aproximación

II. Calibración con métodos de aproximación AFP, AUC, FSC, ESB

III. Aproximación del Caso de uso: Iniciar sesión y Administración de avisos de venta de vehículos

IV. Aproximación del Caso de estudio: Compra-Venta de Autos

Calibración de métodos de aproximación

En este ejemplo se realizará la calibración local de los métodos de aproximación AFP, AUC, FSC, ESB.

- Average Functional Process (AFP) = *CFP / #PF*

No. de PF	CFP (v4.0.1)	AFP (CFP v4.0.1 x FP)
100	760	**7.6**

- Average Use Case (AUC) = *(#PF / #CU) x AFP*

No. de CU	No. de PF	Promedio PF x CU	AFP (CFP v4.0.1 x FP)	AUC (CFP v4.0.1 x CU)
30	100	3.33	7.6	**25.33**

o bien AUC = *CFP / #CU*

No. de CU	CFP (v4.0.1)	AUC (CFP v4.0.1 x CU)
30	760	**25.33**

Facultad de Ciencias

Calibración de métodos de aproximación

- Fixed Size Classification (FSC)

Banda	CFP (v4.0.1)
Pequeño	5
Mediano	10
Grande	15
Muy Grande	20

- Equal Size Bands (ESB)

Banda	Total de CFP (v4.0.1)	Número de Procesos Funcionales	Promedio CFP (v4.0.1)
Pequeño	188	44	4.3
Mediano	188	27	7
Grande	194	18	10.8
Muy Grande	190	11	17.3

Facultad de Ciencias

Agenda

I. Conjunto de Datos 1 para calibración de métodos de aproximación

II. Calibración con métodos de aproximación AFP, AUC, FSC, ESB

III. Aproximación del Caso de uso: Iniciar sesión y Administración de avisos de venta de vehículos

IV. Aproximación del Caso de estudio: Compra-Venta de Autos

Aproximación a nivel de casos de uso

En este ejemplo se realizará la aproximación de tamaño funcional a nivel de casos de uso utilizando el método AUC calibrado y el método EPCU a 44.

- Aproximación por AUC

Caso de Uso	CFP v4.0.1
Iniciar Sesión	**25.33**
Administración de avisos de venta de vehículos	**25.33**

- Aproximación por EPCU a 44

Caso de Uso	EPCU a 44				CFP v4.0.1
	Tamaño del CU		Nivel de presencia de objetos de interés		
	Clasificación	Valor	Clasificación	Valor	
Iniciar Sesión	Pequeño	1.5	Pocos	1	**2.85**
Administración de avisos de venta de vehículos	Grande	3.9	Promedio	3	**26.36**

Resumen por casos de uso

La siguiente tabla muestra el resultado de la aproximación de AUC y EPCU a 44 en comparación con la medición COSMIC.

- Aproximación por casos de uso

Proceso Funcional	AUC	EPCU 44	COSMIC
Iniciar Sesión	25.33	**2.85**	**3**
Administración de avisos de venta de vehículos	25.33	**26.36**	**26**
TOTAL	50.66	***29.21***	***29***

La tabla muestra en negritas el valor de la aproximación más cercano al valor medido utilizando el método COSMIC.

Aproximación a nivel de procesos funcionales

En este ejemplo se realizará la aproximación de tamaño funcional a nivel de procesos funcionales utilizando los métodos AFP, FSC, ESB calibrados y el método EPCU a 16.4.

- Aproximación por AFP

Proceso Funcional	CFP v4.0.1
Iniciar Sesión	**7.6**
Buscar aviso de venta	**7.6**
Registrar aviso de venta	**7.6**
Actualizar aviso de venta	**7.6**
Cancelar aviso de venta	**7.6**

Facultad de Ciencias

Aproximación a nivel de procesos funcionales

- Aproximación por FSC

Proceso Funcional	Clasificación	CFP v4.0.1
Iniciar Sesión	Pequeño	**5**
Buscar aviso de venta	Pequeño	**5**
Registrar aviso de venta	Mediano	**10**
Actualizar aviso de venta	Mediano	**10**
Cancelar aviso de venta	Pequeño	**5**

- Aproximación por ESB

Proceso Funcional	Clasificación	CFP v4.0.1
Iniciar Sesión	Pequeño	**4.3**
Buscar aviso de venta	Pequeño	**4.3**
Registrar aviso de venta	Mediano	**7**
Actualizar aviso de venta	Mediano	**7**
Cancelar aviso de venta	Pequeño	**4.3**

Facultad de Ciencias

Aproximación a nivel de procesos funcionales

- Aproximación por EPCU a 16.4

Caso de Uso	EPCU a 16.4				CFP v4.0.1
	Tamaño del CU		Nivel de presencia de objetos de interés		
	Clasificación	Valor	Clasificación	Valor	
Iniciar Sesión	Pequeño	1.5	Pocos	1	**2.62**
Buscar aviso de venta	Pequeño	1.8	Promedio	2.2	**6.76**
Registrar aviso de venta	Mediano	2.7	Promedio	2.2	**8.95**
Actualizar aviso de venta	Mediano	2.3	Pocos	1.7	**6.89**
Cancelar aviso de venta	Pequeño	1.5	Pocos	1.4	**3.49**

Resumen por procesos funcionales

La siguiente tabla muestra el resultado de la aproximación de AFP, FSC, ESB y EPCU a 16.4 en comparación con la medición COSMIC.

- Aproximación por procesos funcionales

Proceso Funcional	AFP	FSC	ESB	EPCU 16.4	COSMIC
Iniciar Sesión	7.6	5	4.3	**2.62**	**3**
Buscar aviso de venta	7.6	5	4.3	**6.76**	**6**
Registrar aviso de venta	7.6	10	7	**8.95**	**9**
Actualizar aviso de venta	7.6	10	**7**	6.89	**7**
Cancelar aviso de venta	7.6	5	**4.3**	3.49	**4**
TOTAL	38	35	26.9	***28.71***	***29***

La tabla muestra en negritas el valor de la aproximación más cercano al valor medido utilizando el método COSMIC.

Facultad de Ciencias

Agenda

I. Conjunto de Datos 1 para calibración de métodos de aproximación

II. Calibración con métodos de aproximación AFP, AUC, FSC, ESB

III. Aproximación del Caso de uso: Iniciar sesión y Administración de avisos de venta de vehículos

IV. Aproximación del Caso de estudio: Compra-Venta de Autos

Aproximación a nivel de casos de uso

Aproximación de tamaño funcional a nivel de casos de uso utilizando el método AUC calibrado y el método EPCU a 44.

- Aproximación por AUC

Caso de Uso	CFP v4.0.1
Compra-Venta de Autos	**25.33**

- Aproximación por EPCU a 44

Caso de Uso	EPCU a 44				CFP v4.0.1
	Tamaño del CU		Nivel de presencia de objetos de interés		
	Clasificación	Valor	Clasificación	Valor	
Compra-Venta de Autos	Grande	3.2	Muchos	3.2	**30.29**

Resumen por casos de uso

La siguiente tabla muestra el resultado de la aproximación de AUC y EPCU a 44 en comparación con la medición COSMIC.

- Aproximación por casos de uso

Proceso Funcional	AUC	EPCU 44	COSMIC
Compra-Venta de Autos	**25.33**	30.29	**25**

La tabla muestra en negritas el valor de la aproximación más cercano al valor medido utilizando el método COSMIC.

Facultad de Ciencias

Aproximación a nivel de procesos funcionales

Aproximación de tamaño funcional a nivel de procesos funcionales utilizando los métodos AFP, FSC, ESB calibrados y el método EPCU a 16.4.

- Aproximación por AFP

Proceso Funcional	AFP
Consultar lista de vehículos	7.6
Ver detalle de vehículo	7.6
Comprar Vehículo	7.6

- Aproximación por FSC y ESB

Proceso Funcional	Clasificación	FSC	ESB
Consultar lista de vehículos	Mediano	10	7
Ver detalle de vehículo	Mediano	10	7
Comprar Vehículo	Pequeño	5	4.3

Facultad de Ciencias

Aproximación a nivel de procesos funcionales

- Aproximación por EPCU a 16.4

Caso de Uso	EPCU a 16.4				CFP v4.0.1
	Tamaño del CU		Nivel de presencia de objetos de interés		
	Clasificación	Valor	Clasificación	Valor	
Consultar lista de vehículos	Promedio	2.5	Promedio	2.7	**10.89**
Ver detalle de vehículo	Promedio	2.2	Promedio	2.2	**7.54**
Comprar Vehículo	Pequeño	1.7	Promedio	2.4	**7.29**

Facultad de Ciencias

Resumen por procesos funcionales

La siguiente tabla muestra el resultado de la aproximación de AFP, FSC, ESB y EPCU a 16.4 en comparación con la medición COSMIC.

- Aproximación por procesos funcionales

Proceso Funcional	AFP	FSC	ESB	EPCU 16.4	COSMIC
Consultar lista de vehículos	7.6	10	7	**10.89**	**15**
Ver detalle de vehículo	7.6	10	**7**	7.54	**6**
Comprar Vehículo	7.6	5	**4.3**	7.29	**4**
Total	22.8	***25***	18.3	25.72	***25***

La tabla muestra en negritas el valor de la aproximación más cercano al valor medido utilizando el método COSMIC.

Facultad de Ciencias

Contacto

- Jesús Iván Saavedra Martínez
- jism@ciencias.unam.mx

Métricas de Software

Estimación de Software

Jesús Iván Saavedra Martínez

Octubre 2017

Facultad de Ciencias

Agenda

I. Introducción

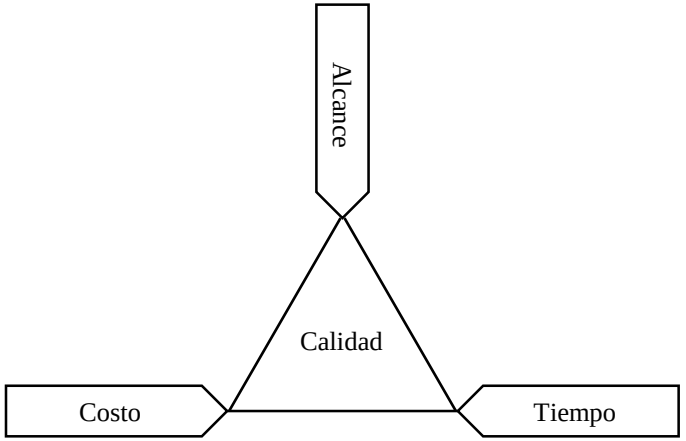
II. Clasificación de estimación

III. Estimación: arte o ingeniería?

IV. Las fases en la estimación

Introducción

- Todos los proyectos de software se realizan bajo restricciones.



Triángulo de hierro

- Todas las organizaciones que desarrollan software deben de estimar proyectos de desarrollo de software para poder manejar apropiadamente sus recursos.

¿Para qué estimamos?

- Morgenshtern (2007) menciona los siguientes usos de la estimación de proyectos:
  - selección de proyectos
  - administración de personal
  - planeación
  - monitoreo y control
  - evaluación de desempeño del equipo

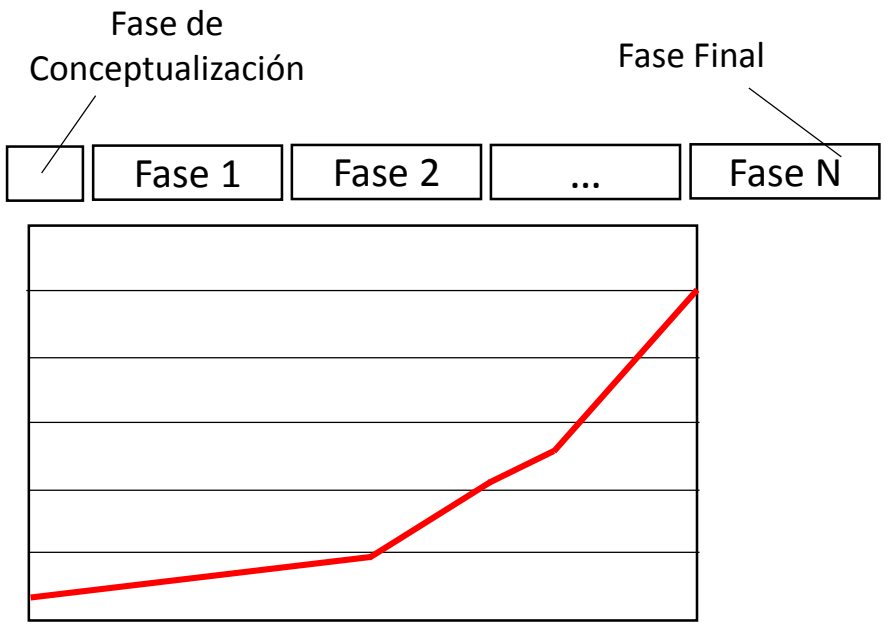
Importancia de la estimación

- "De acuerdo con un testimonio de la Government Accountability Office el pasado septiembre, **si se establecieran bases de referencia más realistas de los requisitos, costo, calendario y riesgo durante las fases de planificación del proyecto, se podrían evitar casi la mitad de los programas de TI cancelados o sobre estimados. Eso ahorraría $ 5.5 mil millones anualmente**, según un estudio realizado por Price Systems LLC, una empresa de software y consultoría en Mount Laurel, N.J., EE.UU. El estudio considera que 104 ejecutivos de TI del gobierno "

*"*Off Base Insufficient expertise in setting baselines hits U.S federal IT budgets where it hurts", PM NETWORK, March 2007 / VOLUME 21

Inician los problemas

- La estimación es especialmente importante en etapas tempranas de un proyecto de desarrollo de software, cuándo el software es conceptualizado, esto es, en la etapa de factibilidad; cuándo la información relevante acerca del proyecto es:
  - Todavía vaga o ambigua
  - Imprecisa
  - Expresada típicamente con valores lingüísticos



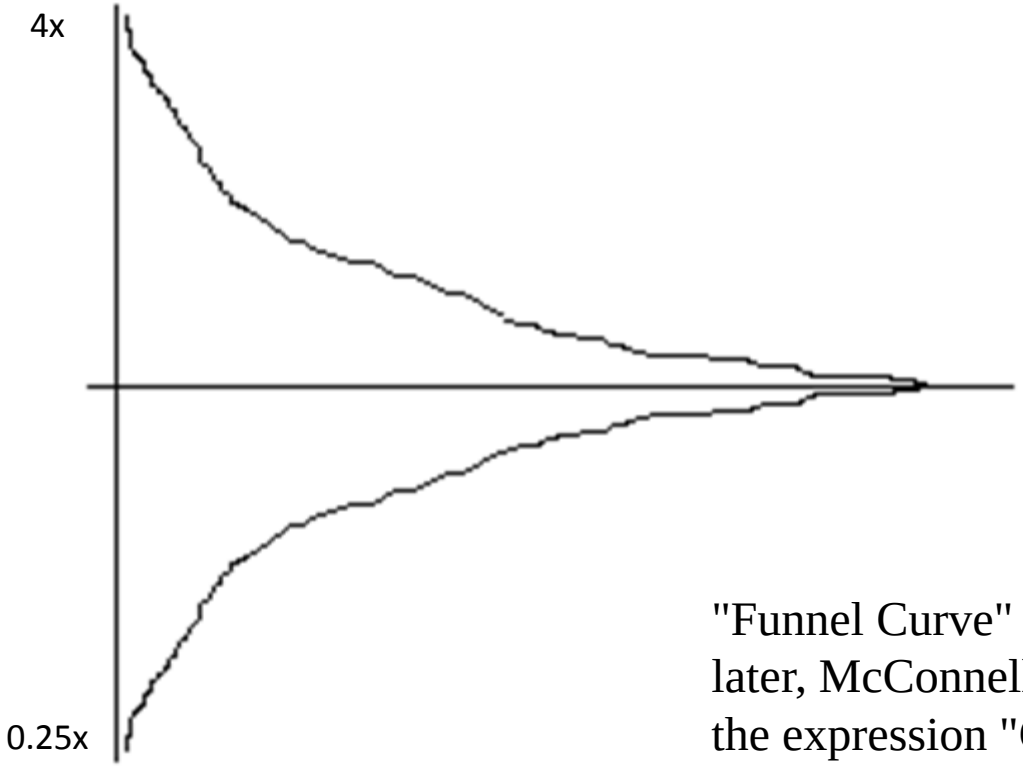
Problemas

- El problema con las características del software descritas de manera lingüística, es que la experiencia está directamente involucrada en el proceso de medición.
- El manejo de la imprecisión y la incertidumbre se realiza frecuentemente en un esquema ad-hoc, en lugar de utilizar técnicas relevantes y adecuadas.

- "Las estimaciones son, en realidad, supuestos respecto al rendimiento futuro basado en el conocimiento disponible. Como tal, su precisión se ve afectada por el grado de incertidumbre respecto a la tarea de estimar. La incertidumbre está asociada, entre otras cosas, con definiciones de requisitos, elección de soluciones tecnológicas, innovación de necesidades y características del cliente " (Morgenshtern, 2007).

Cono de incertidumbre

Variabilidad en la Estimación del Alcance del Proyecto



"Funnel Curve" (Boehm, 1981); later, McConnell (2006) has used the expression "Cone of Uncertainty“

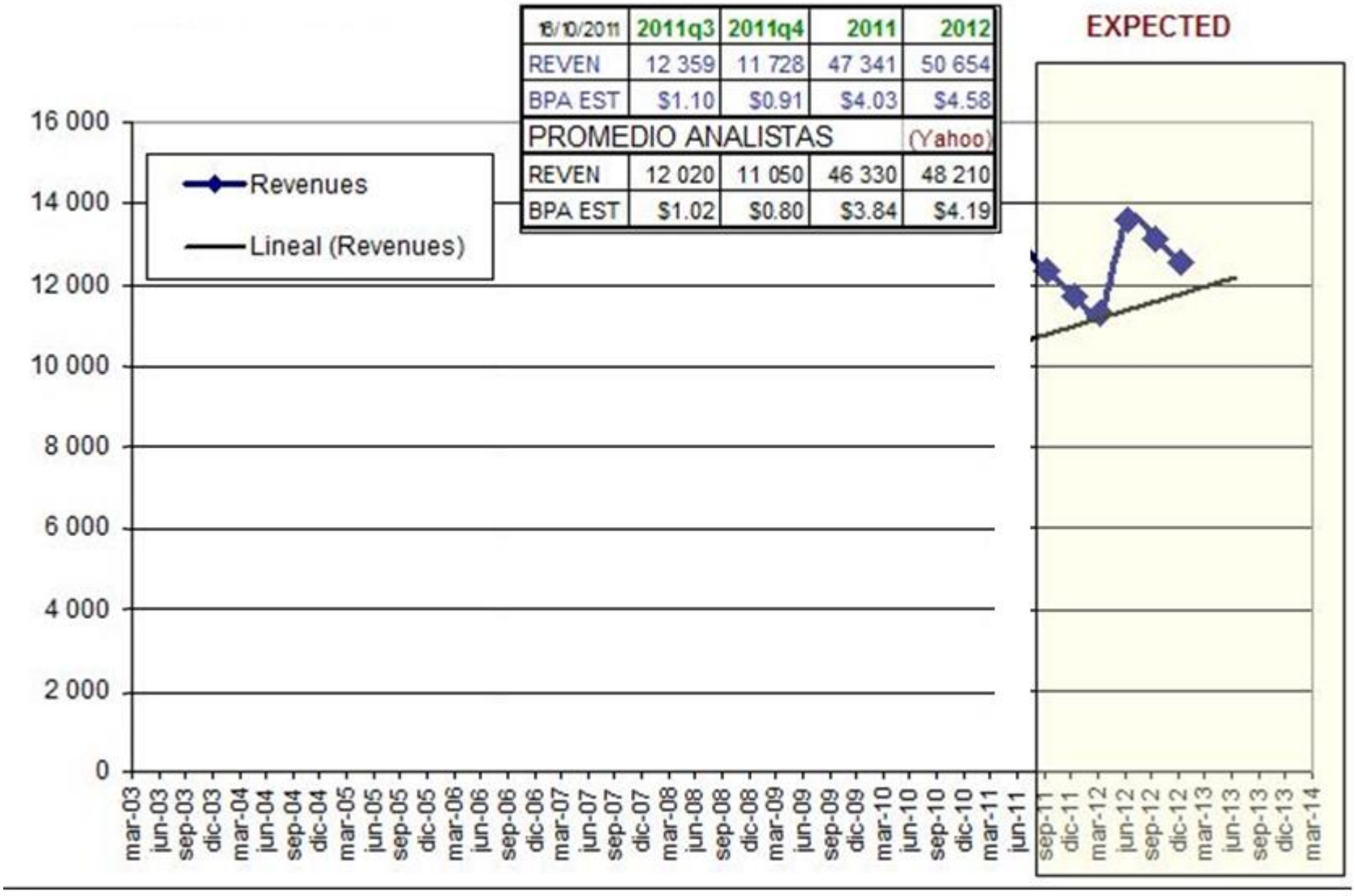
[-25%, 400%]

Estimación de software

- “Una estimación es una predicción que es igualmente probable que esté por encima o por debajo del resultado real.” **T. DeMarco**

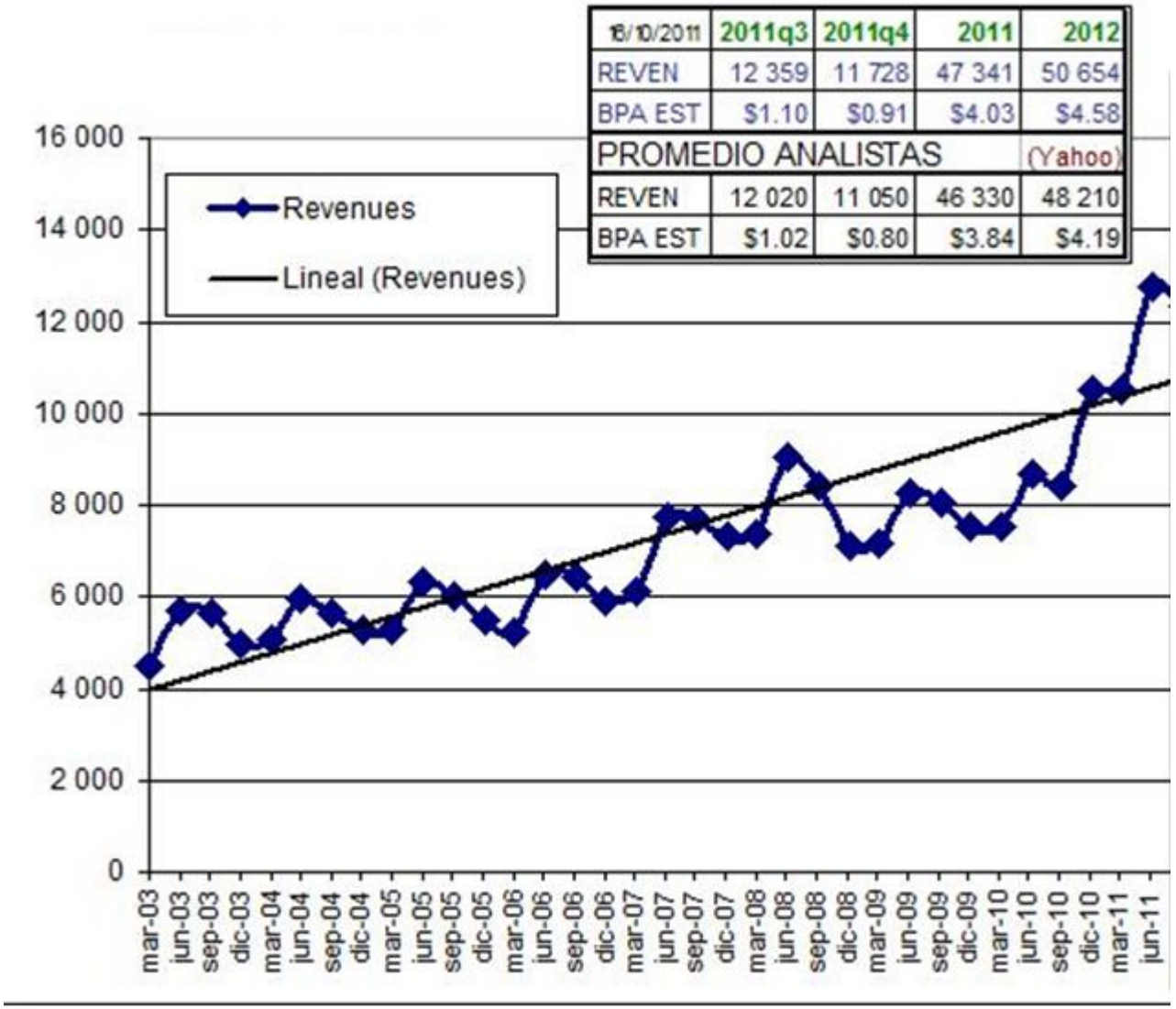
- La estimación es **especialmente importante en etapas tempranas del proyecto** ya que proporciona entradas creíbles para hacer decisiones de negocio.

Medición => Estimación (Modelos predictivos)



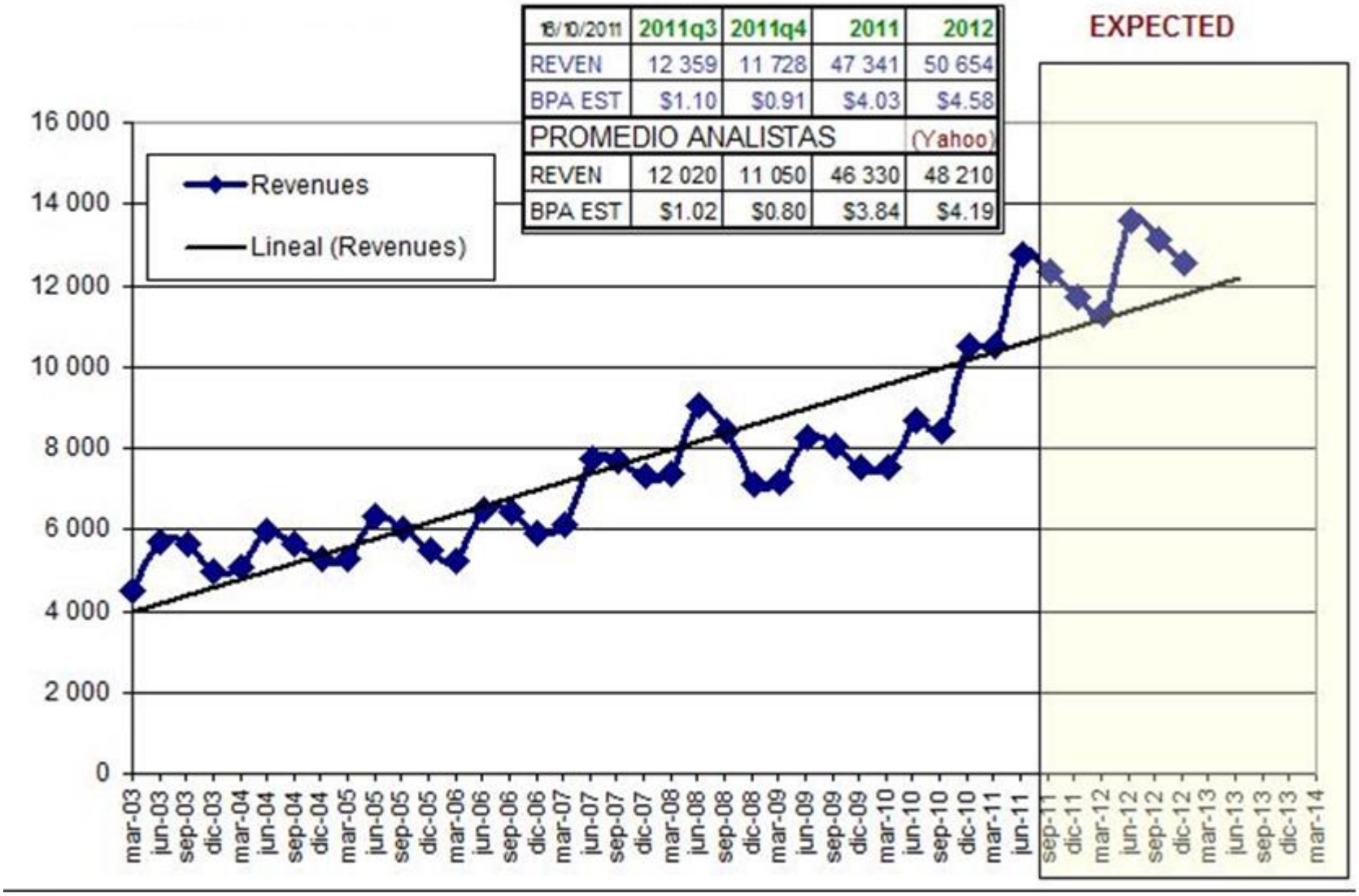
Medición => Estimación (Modelos predictivos)

18/10/2011	2011q3	2011q4	2011	2012
REVEN	12 359	11 728	47 341	50 654
BPA EST	$1.10	$0.91	$4.03	$4.58
PROMEDIO ANALISTAS				(Yahoo)
REVEN	12 020	11 050	46 330	48 210
BPA EST	$1.02	$0.80	$3.84	$4.19



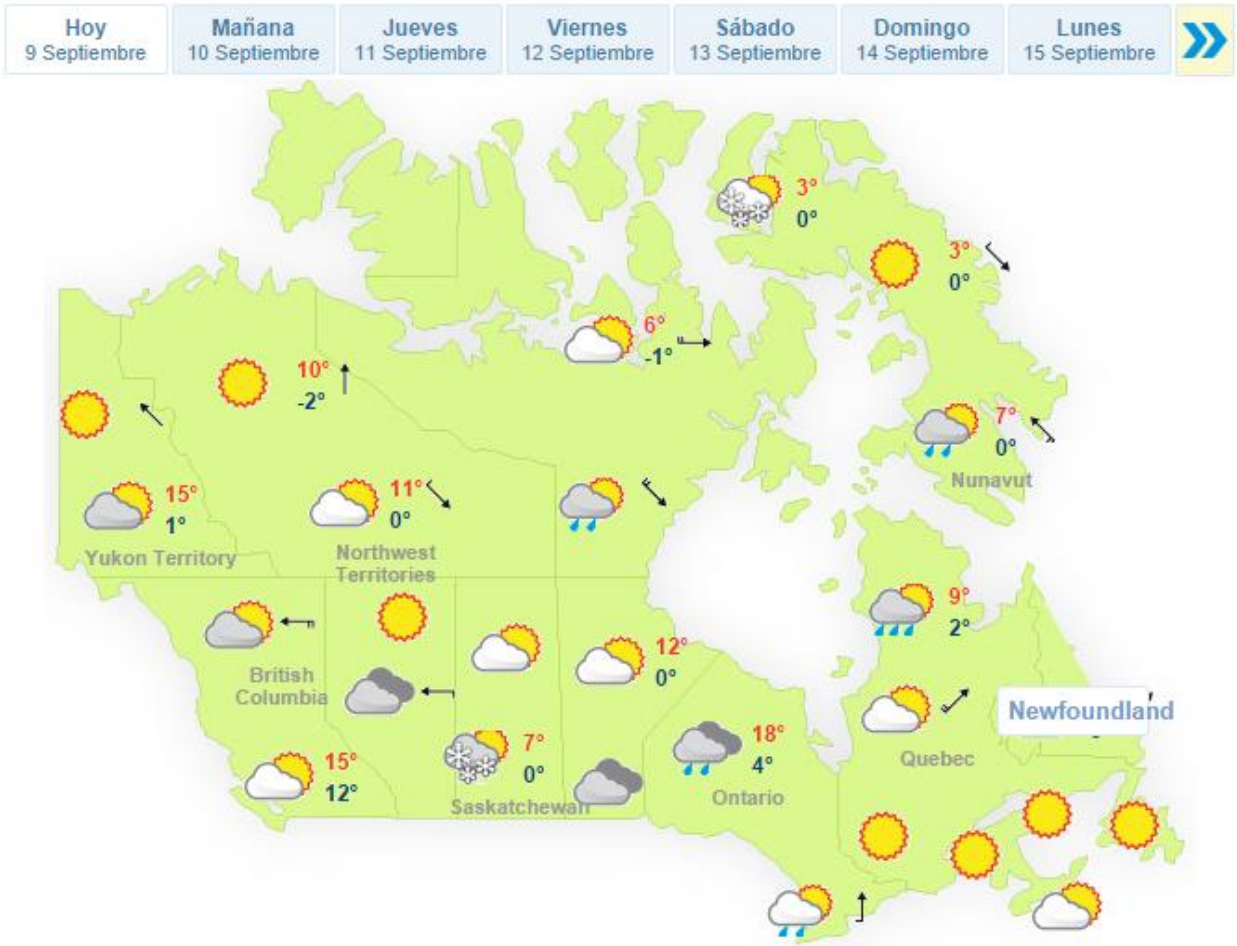
Medición => Estimación (Modelos predictivos)

18/10/2011	2011q3	2011q4	2011	2012
REVEN	12 359	11 728	47 341	50 654
BPA EST	$1.10	$0.91	$4.03	$4.58
PROMEDIO ANALISTAS				(Yahoo)
REVEN	12 020	11 050	46 330	48 210
BPA EST	$1.02	$0.80	$3.84	$4.19

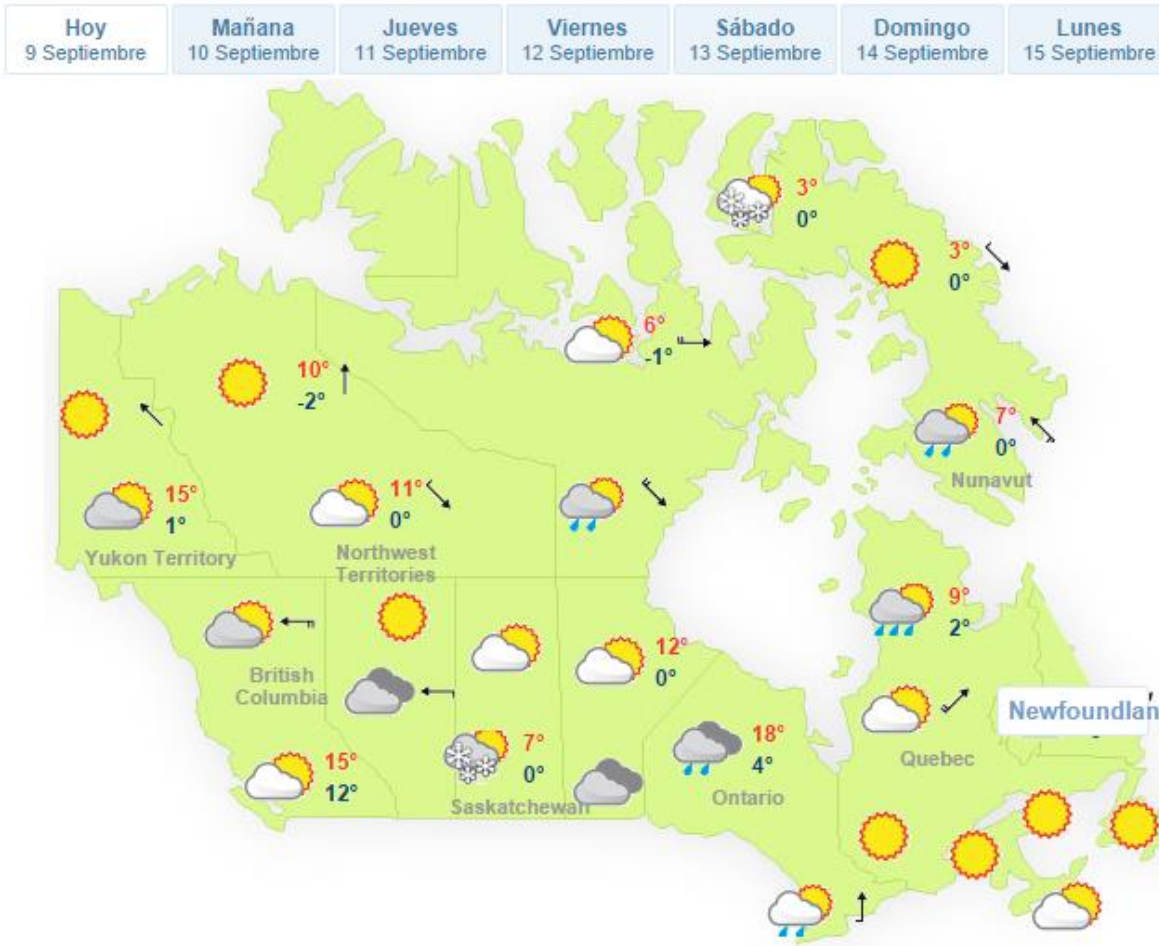


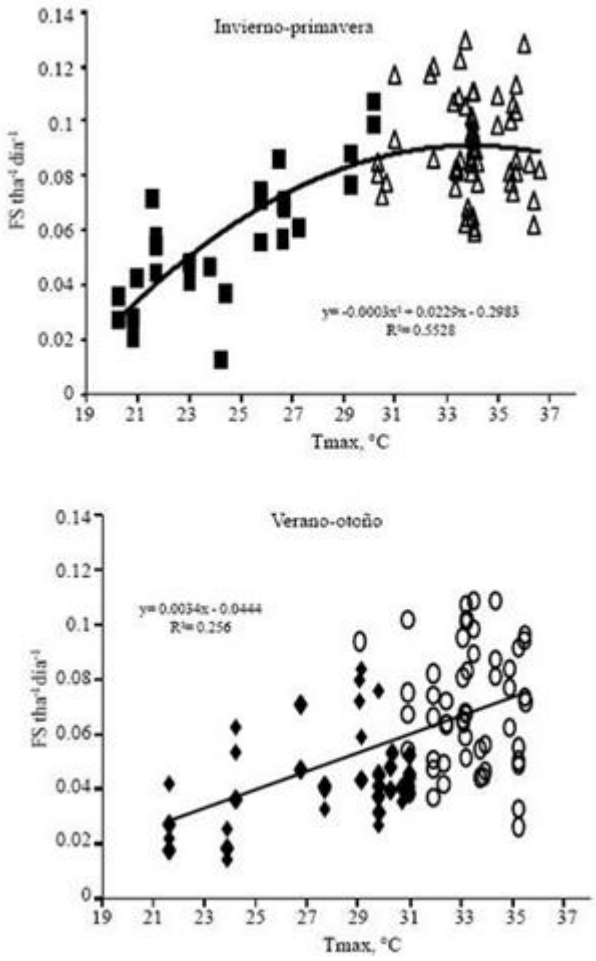
Facultad de Ciencias

Medición => Estimación (Modelos predictivos)



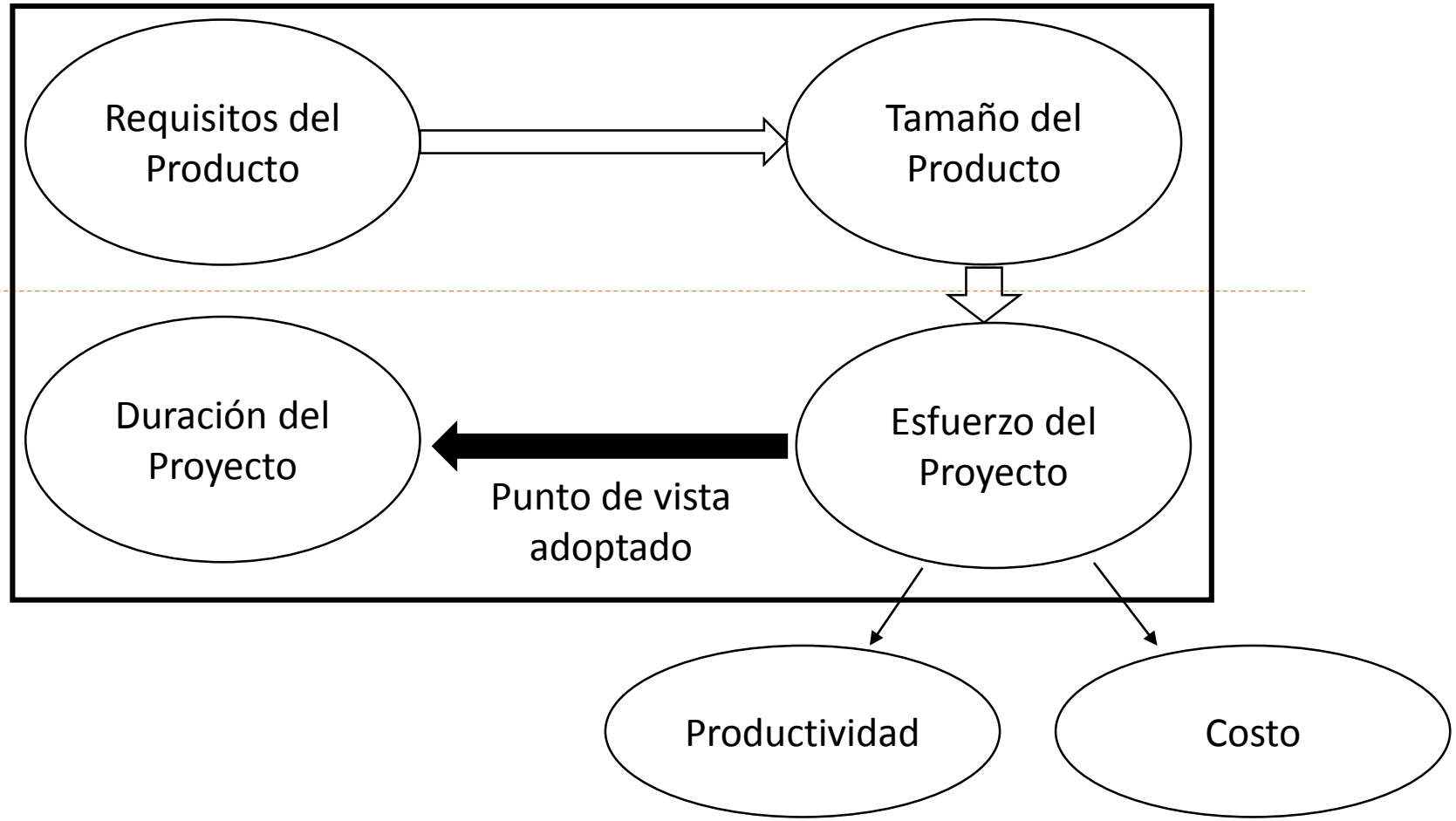
Medición => Estimación (Modelos predictivos)





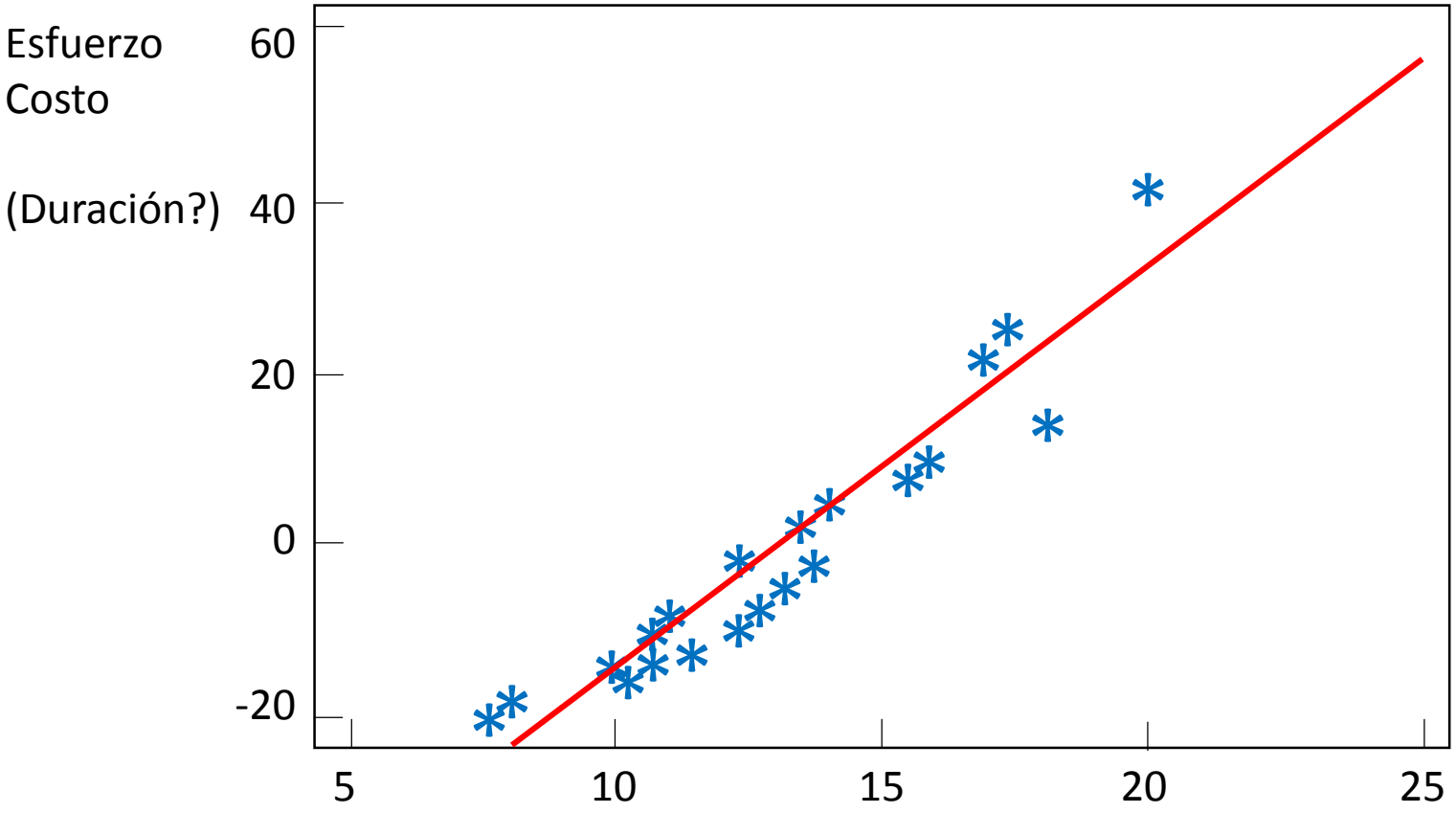
Pasos en la estimación

Pierre Bourque (May 2007)



Facultad de Ciencias

¿Qué estimar?



Facultad de Ciencias

Agenda

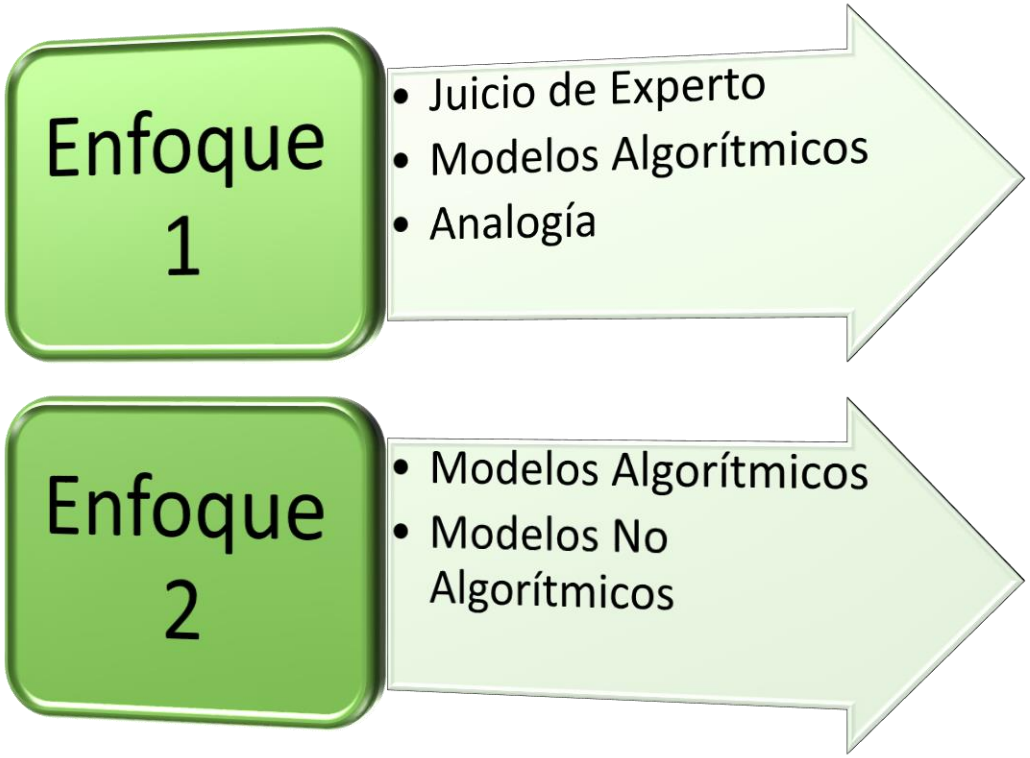
I. Introducción

II. Clasificación de estimación

III. Estimación: arte o ingeniería?

IV. Las fases en la estimación

Clasificación de Estimación (Fundamentos)



Modelos Algorítmicos

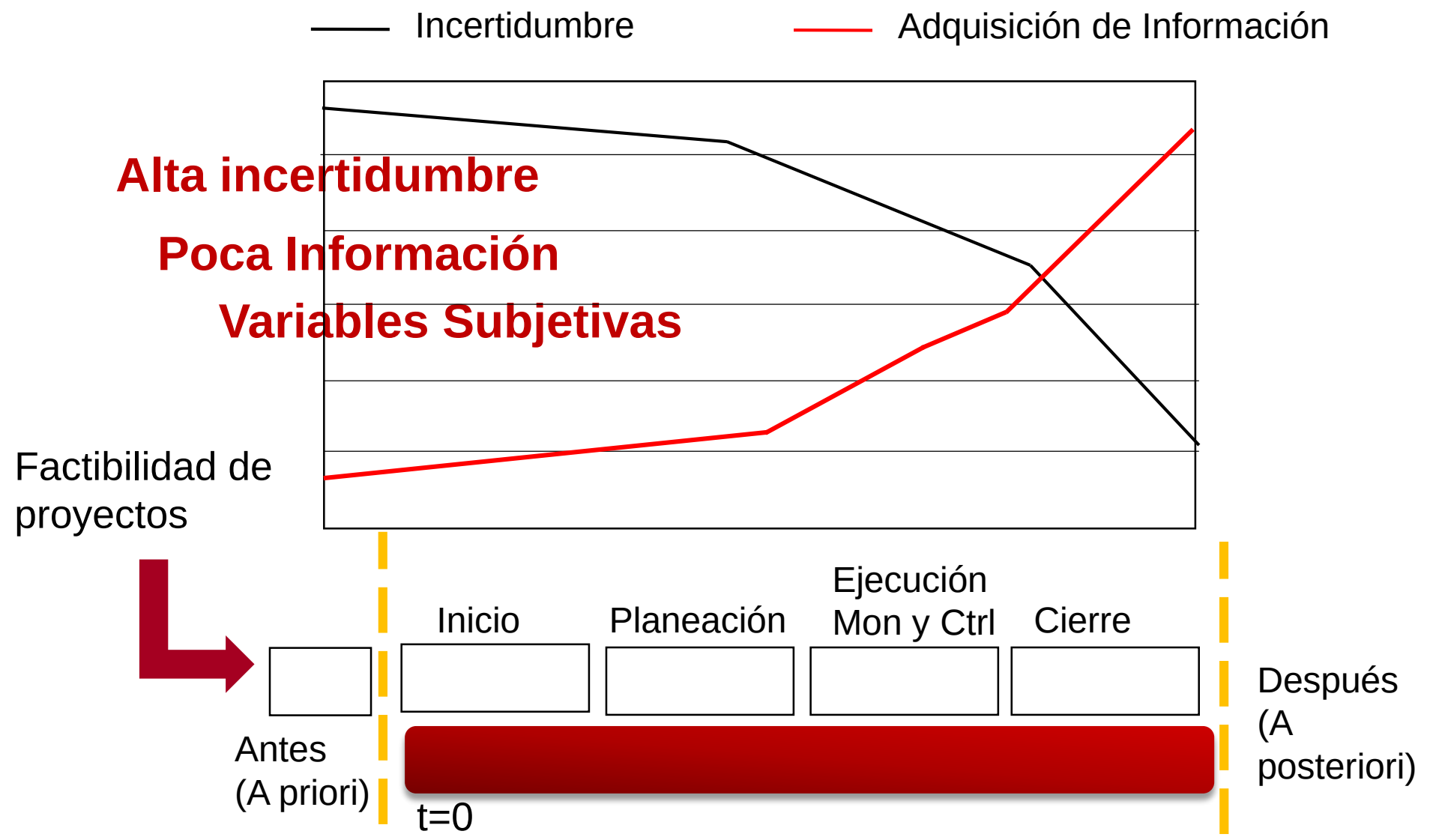
- Derivados básicamente de datos de proyectos terminados.
- La forma de la función de estimación es predeterminada:
- Effort = α x sizeβ, α representa el coeficiente de productividad y β representa el coeficiente de economías o diseconomías de escala.
- Estos modelos necesitan ser ajustados localmente.
- Necesitan datos históricos.

Modelos NO Algorítmicos

- Buscan modelar más adecuadamente la complejidad en las relaciones entre las variables independientes y la variable dependiente.
- El conocimiento requerido para implementarlos, así como el costo de implementarlos es usualmente alto.
- No hay herramientas sencillas para usuarios.
- Se conocen más en la academia que en la industria.

- EJEMPLOS: CBR, RN

Clasificación de Estimación (Momento de realización)



Estimando Software "A posteriori"

- CARACTERÍSTICAS
  - Se conoce toda la información.
  - No hay incertidumbre.

- MÉTODOS UTILIZADOS
  - Métodos algorítmicos basados en análisis numéricos o estadísticos, como por ejemplo los basados en Function Points (IFPUG), Use Case Points (UCP), COCOMO, PMI, **COSMIC**, para estimar esfuerzo, costo, duración.

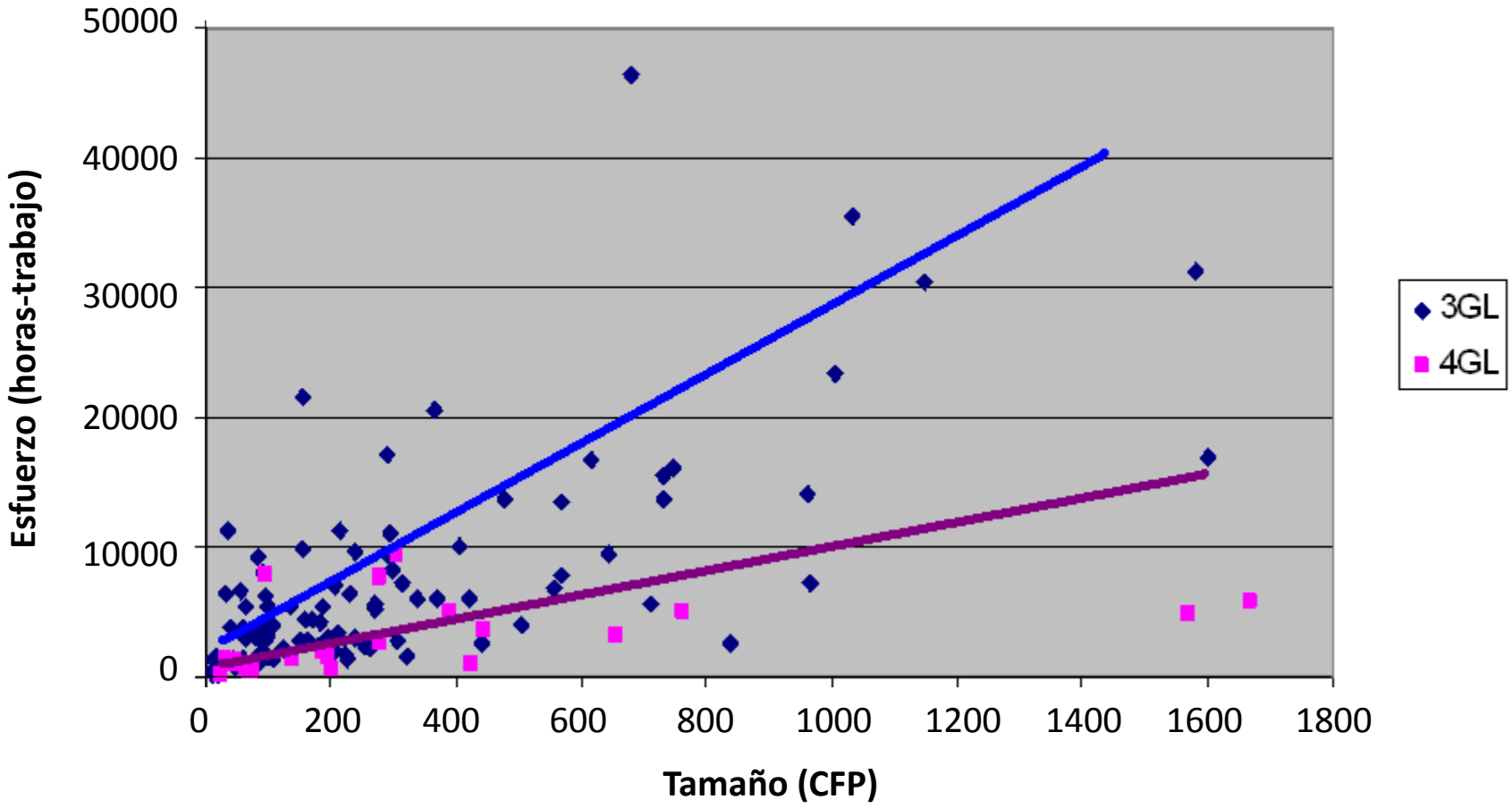
Estimando Software “A posteriori”

- VENTAJAS
  - Fácil interpretación.

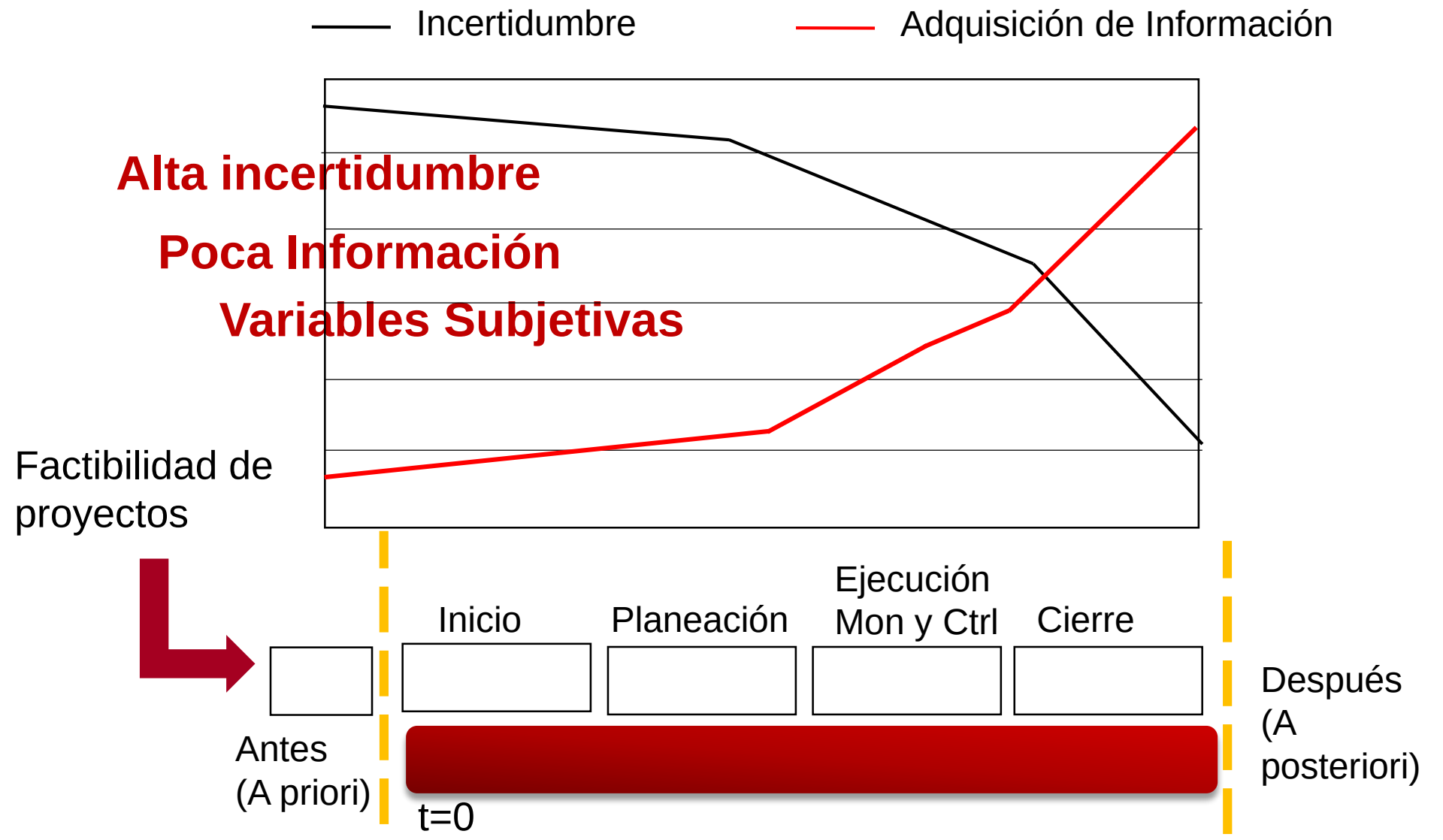
- DESVENTAJAS
  - Requieren información **oportuna**, **confiable** y **precisa.** Lo que incrementa la infraestructura de adquisición de información y en consecuencia el costo.

  - Requiere datos históricos**!!!!!!!**.

Estimación de Proyectos



Clasificación de Estimación (Momento de realización)



Estimando Software “A priori”

- CARACTERÍSTICAS
  - Se tiene un ambiente de alta incertidumbre.
  - Nivel de abstracción de necesidad muy alto.
  - Usualmente se tiene un ventana de tiempo o de recursos limitados.
  - La mayoría de las variables que se conocen **son cualitativas**.
- MÉTODOS UTILIZADOS
  - Modelos “A posteriori”
  - **Juicio de Expertos**

Estimando Software “A priori”

- DESVENTAJAS
  - La experiencia le pertenece al experto no a la organización.
  - La experiencia no se puede replicar sistemáticamente.

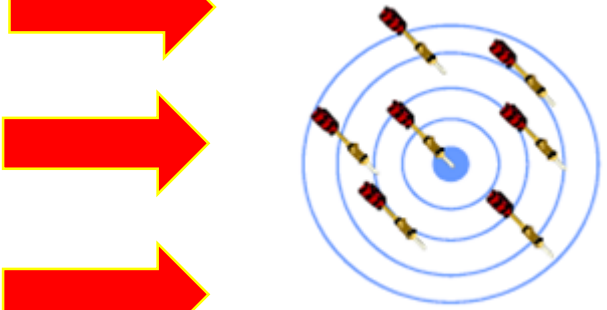
- VENTAJAS
  - La gente de negocios la mayoría del tiempo se basa en información incompleta para tomar decisiones. Los tomadores de decisiones pueden hacer juicios de experto valiosos sobre la información incompleta.
  - Manejo de incertidumbre a través de **precisión de significado**.
  - No se necesita historia.

Facultad de Ciencias

Agenda

I. Introducción
II. Clasificación de estimación
III. Estimación: arte o ingeniería?
IV. Las fases en la estimación

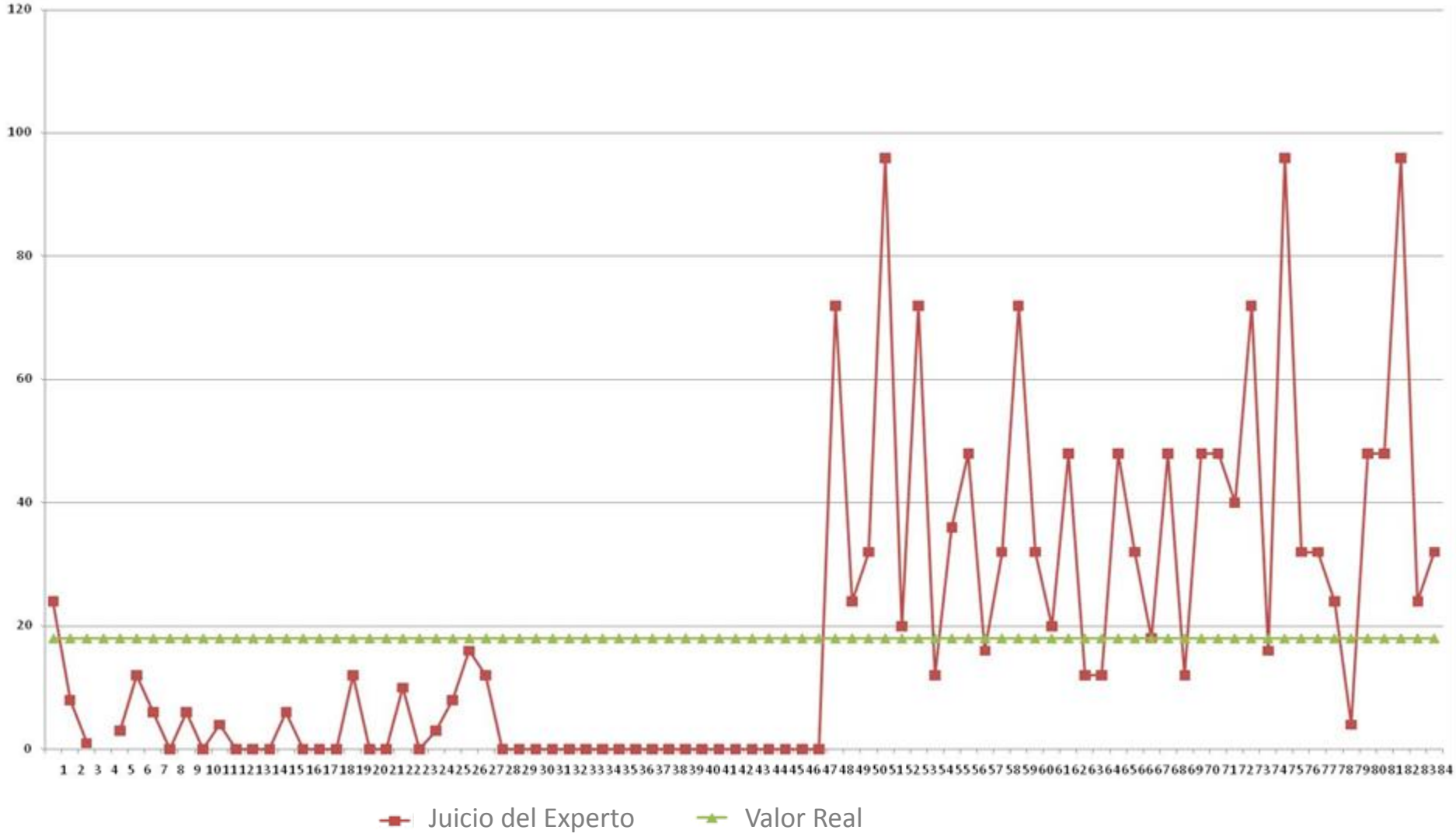
Enfoque más utilizado para estimar: Juicio de Experto (Discrecionalidad)



Generador de Estimados

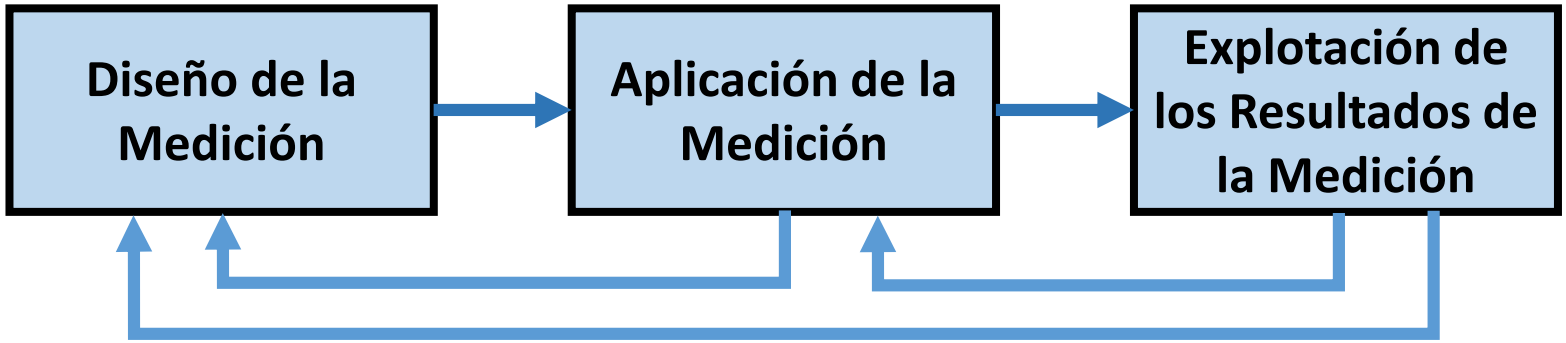
Facultad de Ciencias

Enfoque más utilizado para estimar: Juicio de Experto (ejemplo)



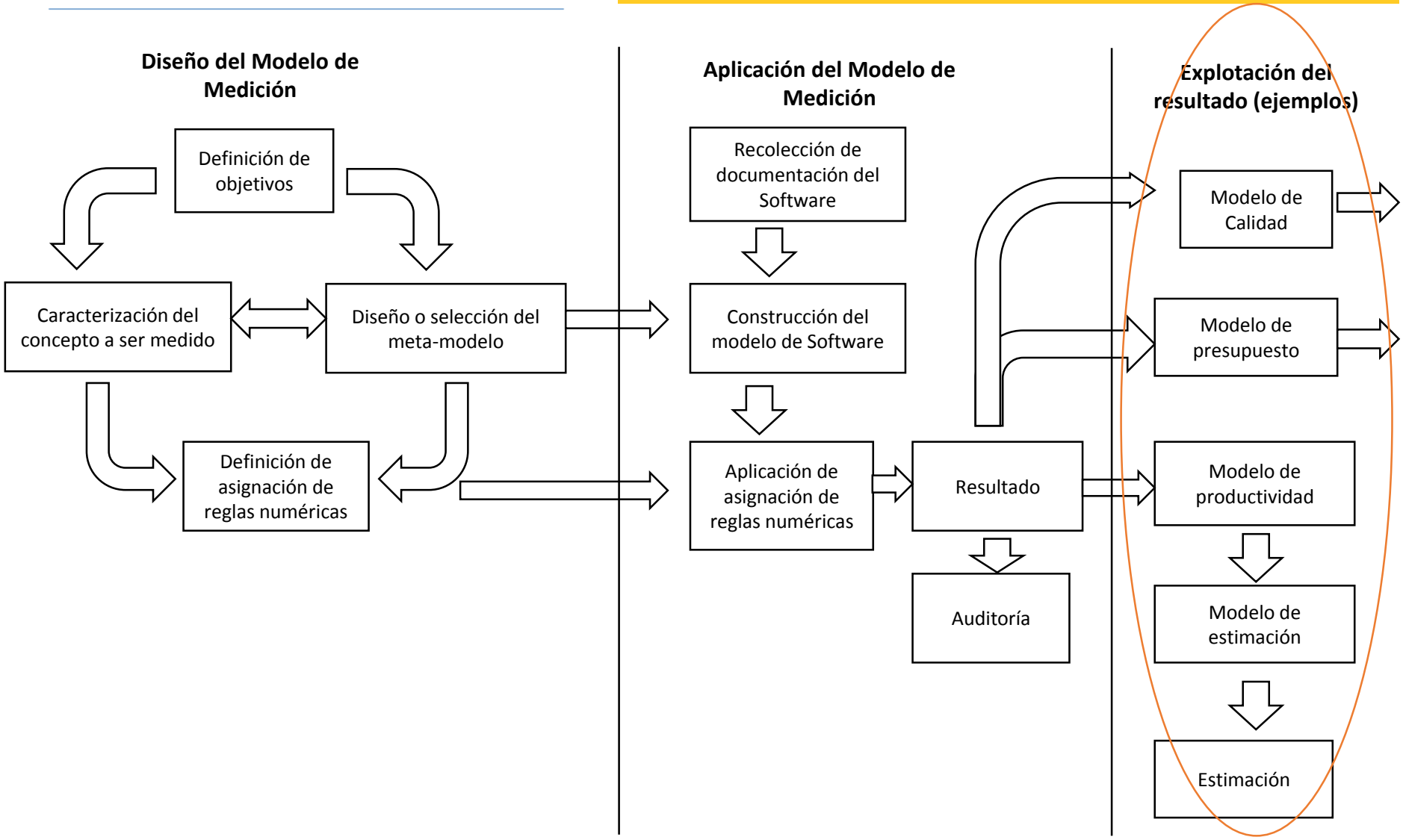
Facultad de Ciencias

¿Por qué es complejo estimar software?



1. Diseño	2. Aplicación	3. Explotación
Verificación del diseño de un método de medición (con respecto al objetivo y el principio de medición)	Verificación del los procedimientos de medición, incluyendo los instrumentos o pasos de medición y los resultados obtenidos.	Verificación de la calidad de los modelos y la relación entre múltiples resultados de medición

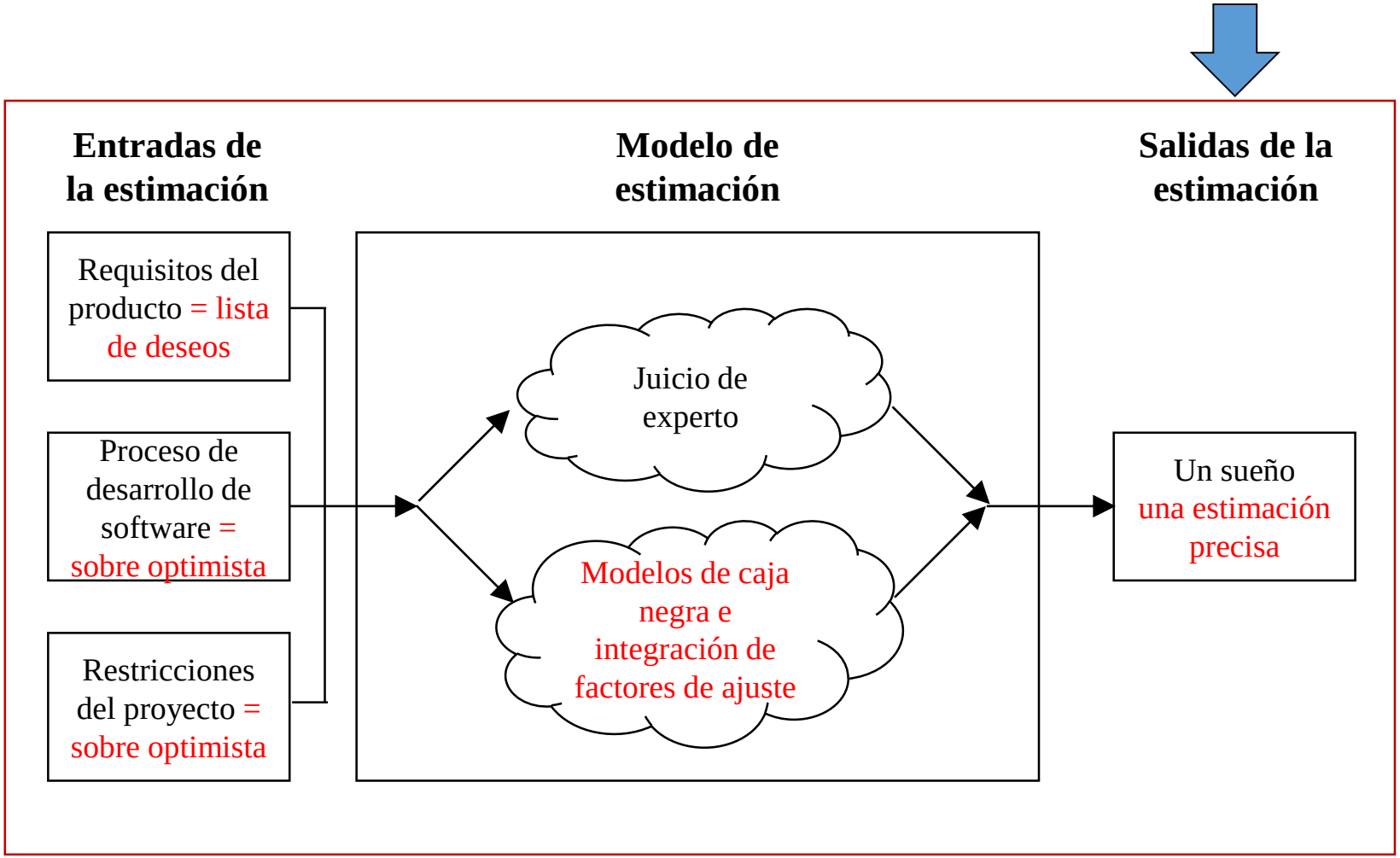
¿Por qué es complejo estimar software?



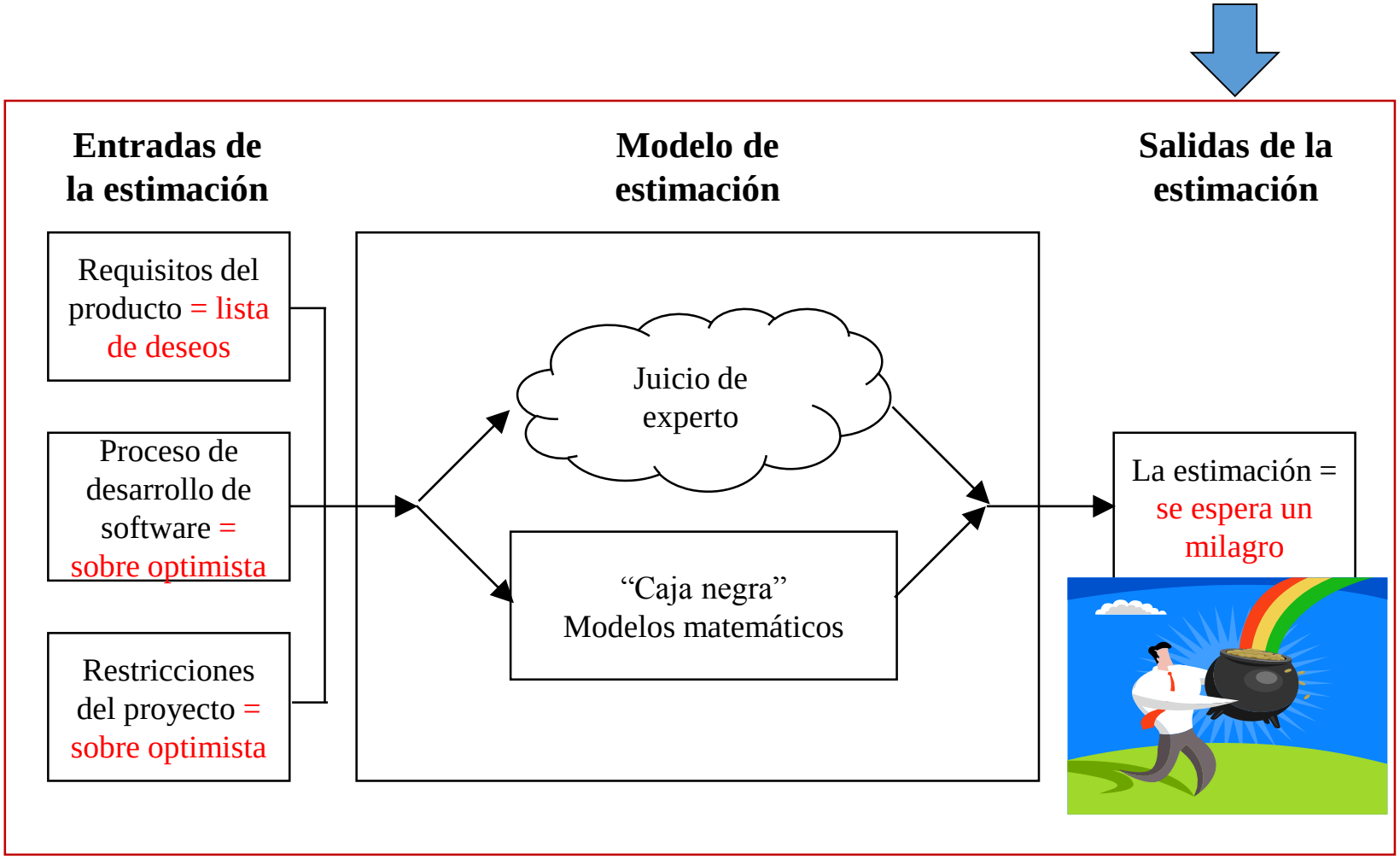
Técnicas de estimación en la literatura

- Regresión de mínimos cuadrados (Least Squares Regression)
- Regresión robusta (Robust Regression)
- Redes Neuronales (Neural networks)
- Lógica Difusa (Fuzzy Systems)
- Razonamiento Basado en Casos (Case-Based Reasoning)

Estimación


Entradas de la estimación
Modelo de estimación
Salidas de la estimación
Requisitos del producto = lista de deseos
Proceso de desarrollo de software = sobre optimista
Restricciones del proyecto = sobre optimista
Juicio de experto
Modelos de caja negra e integración de factores de ajuste
Un sueño una estimación precisa

Estimación


Entradas de la estimación
Modelo de estimación
Salidas de la estimación
Requisitos del producto = lista de deseos
Proceso de desarrollo de software = sobre optimista
Restricciones del proyecto = sobre optimista
Juicio de experto
"Caja negra" Modelos matemáticos
La estimación = se espera un milagro

Resultados de la estimación!

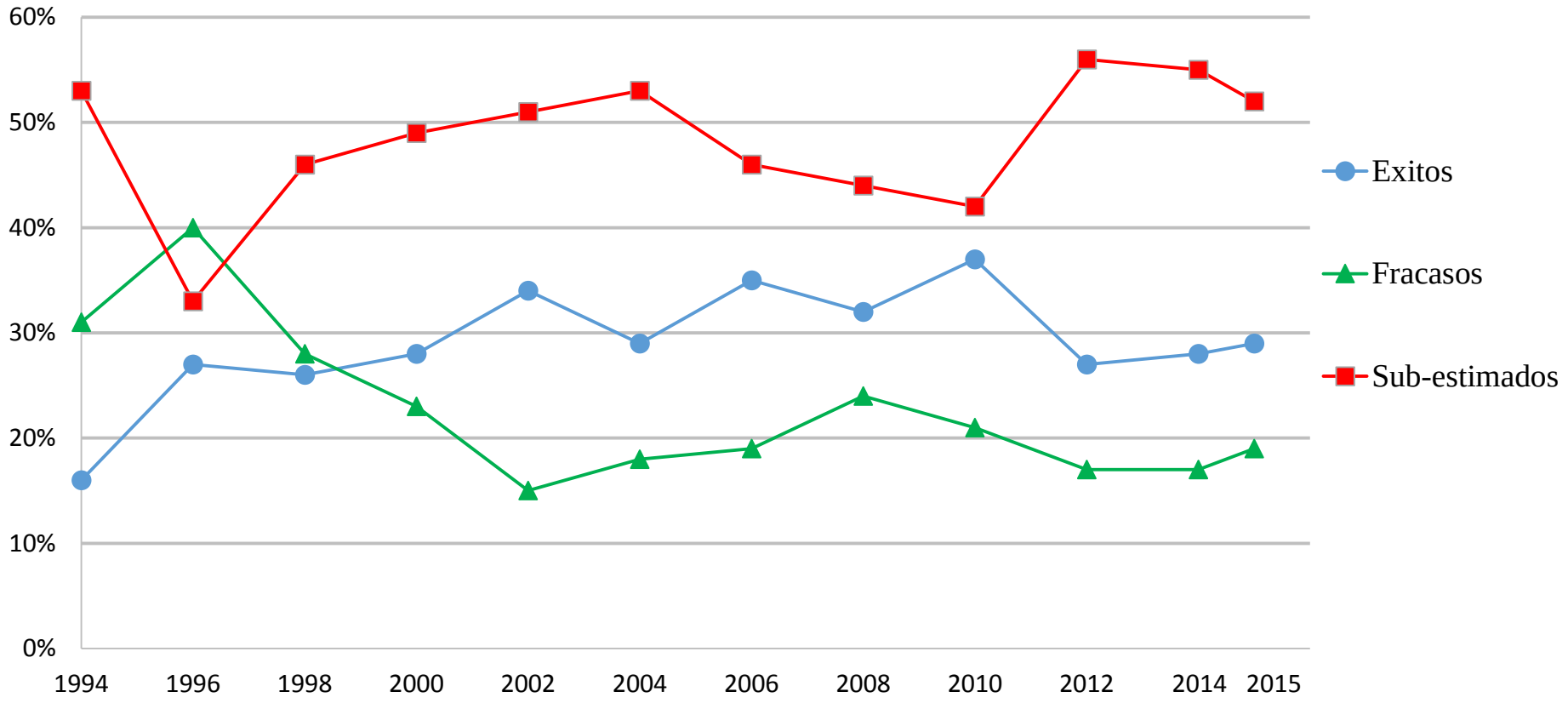
The ***'feel-good'*** → dead end!

→ **Quick & Easy…**

→

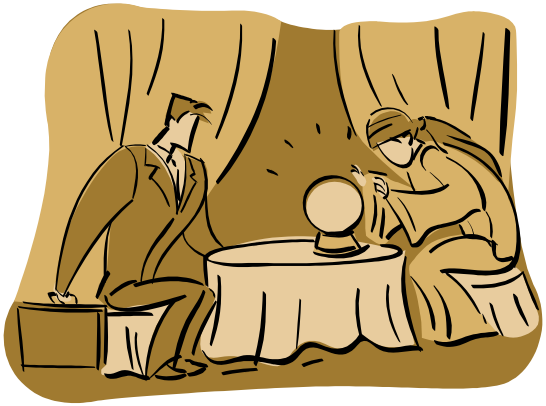
Facultad de Ciencias

Éxito de los proyectos



Tendencias de éxito de proyectos de acuerdo a Standish Group

Estimación (Software)



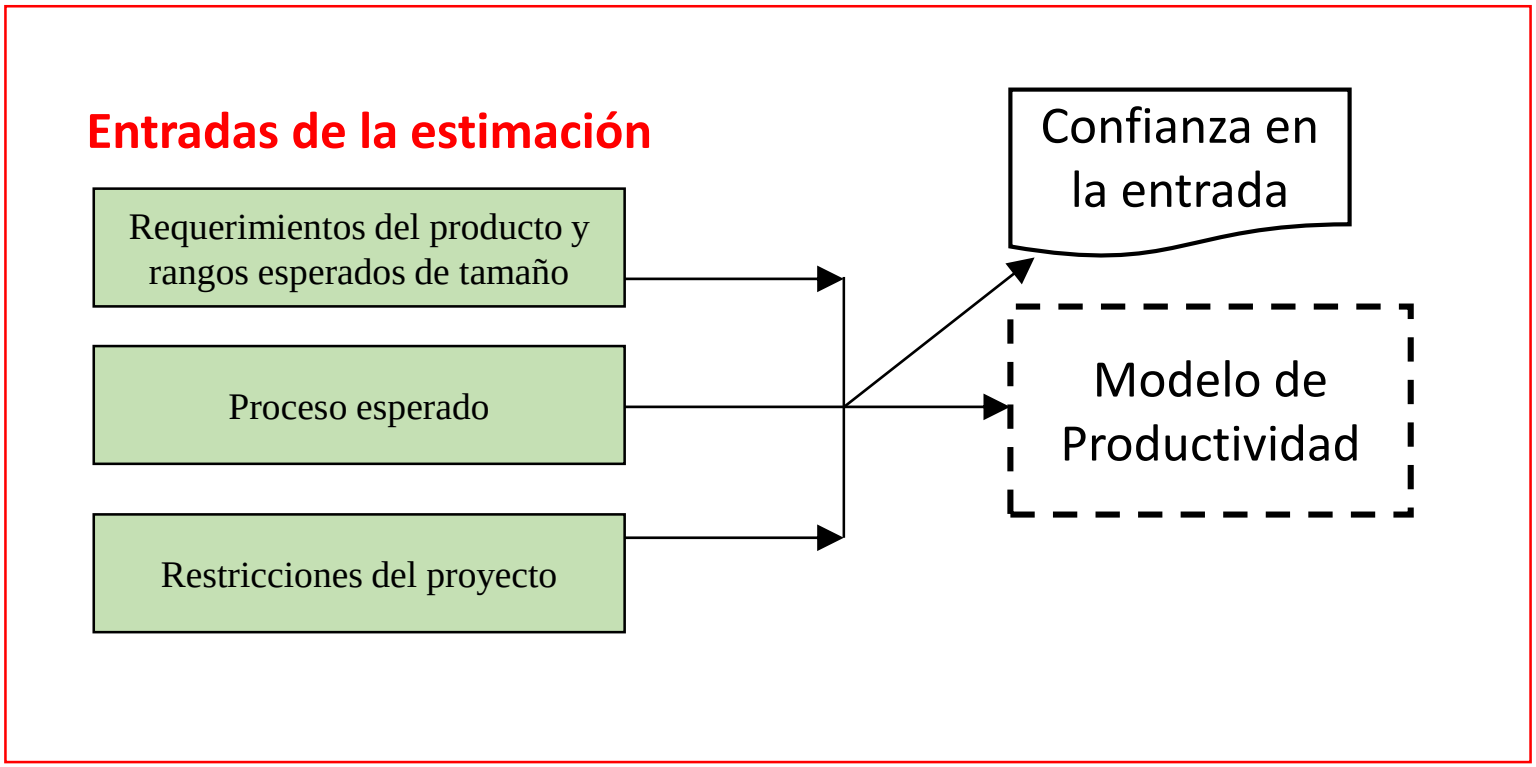
O?

Facultad de Ciencias

Agenda

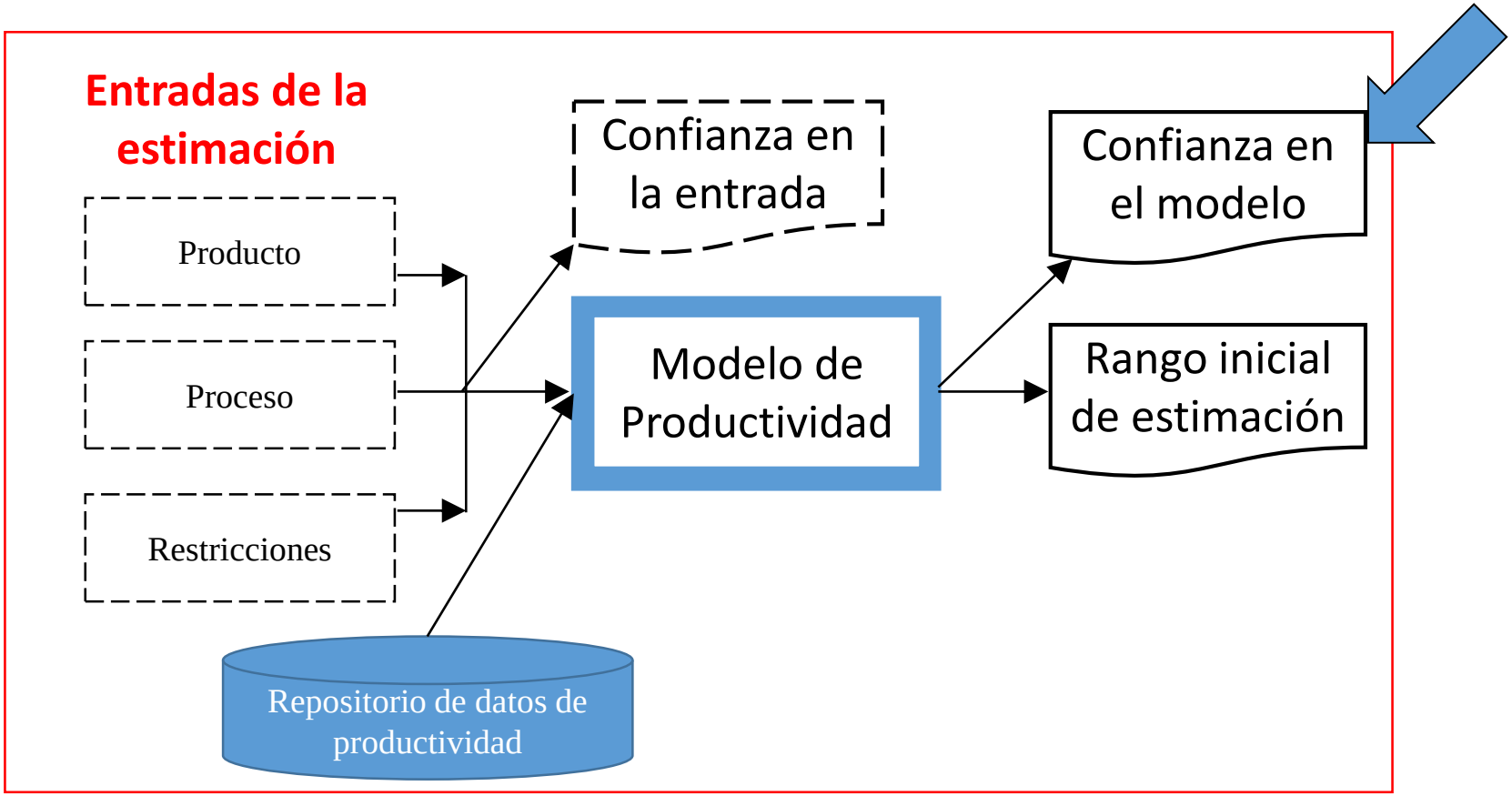
I. Introducción
II. Clasificación de estimación
III. Estimación: arte o ingeniería?
IV. Las fases en la estimación

Fases de la estimación



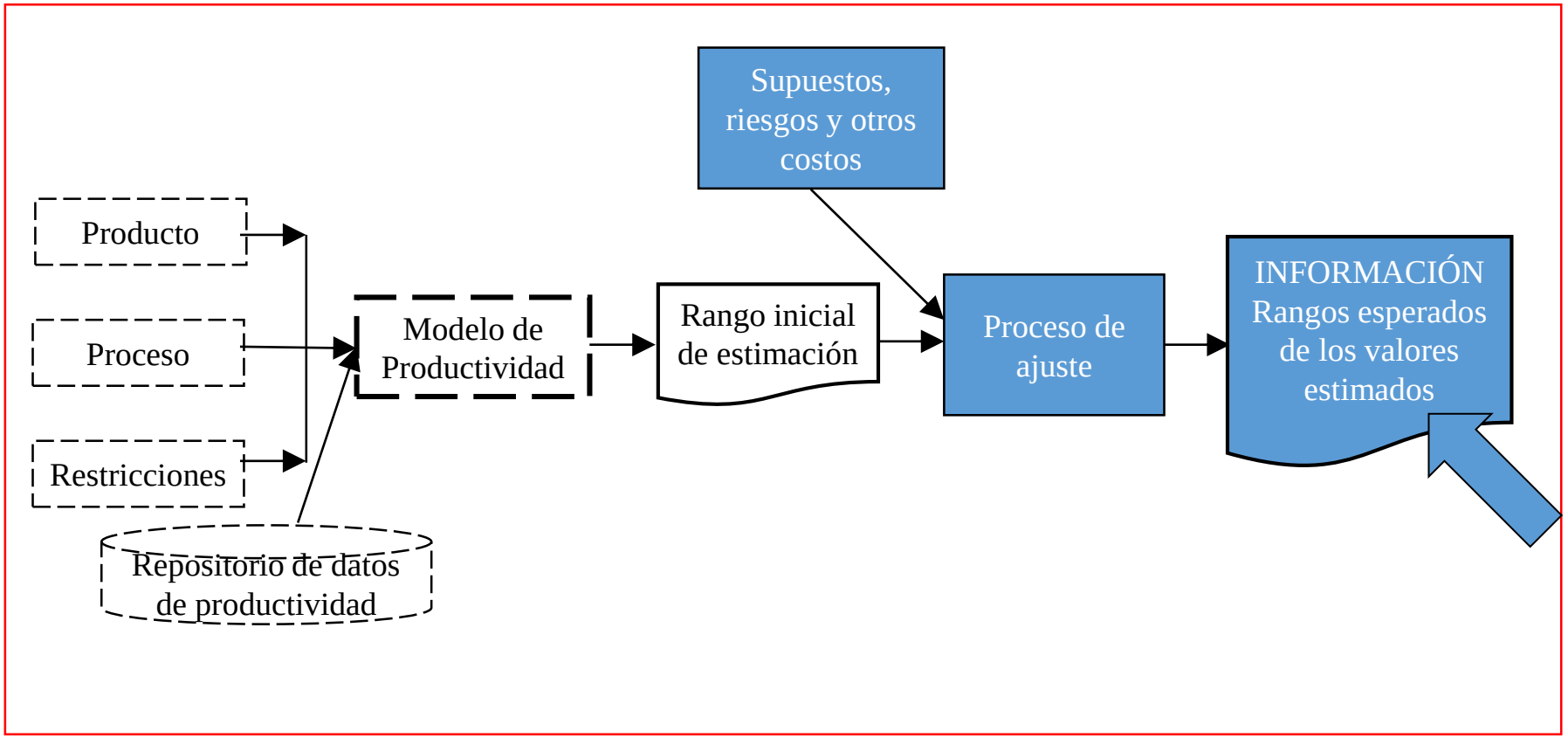
Colección de entradas para el proceso de estimación

Fases de la estimación



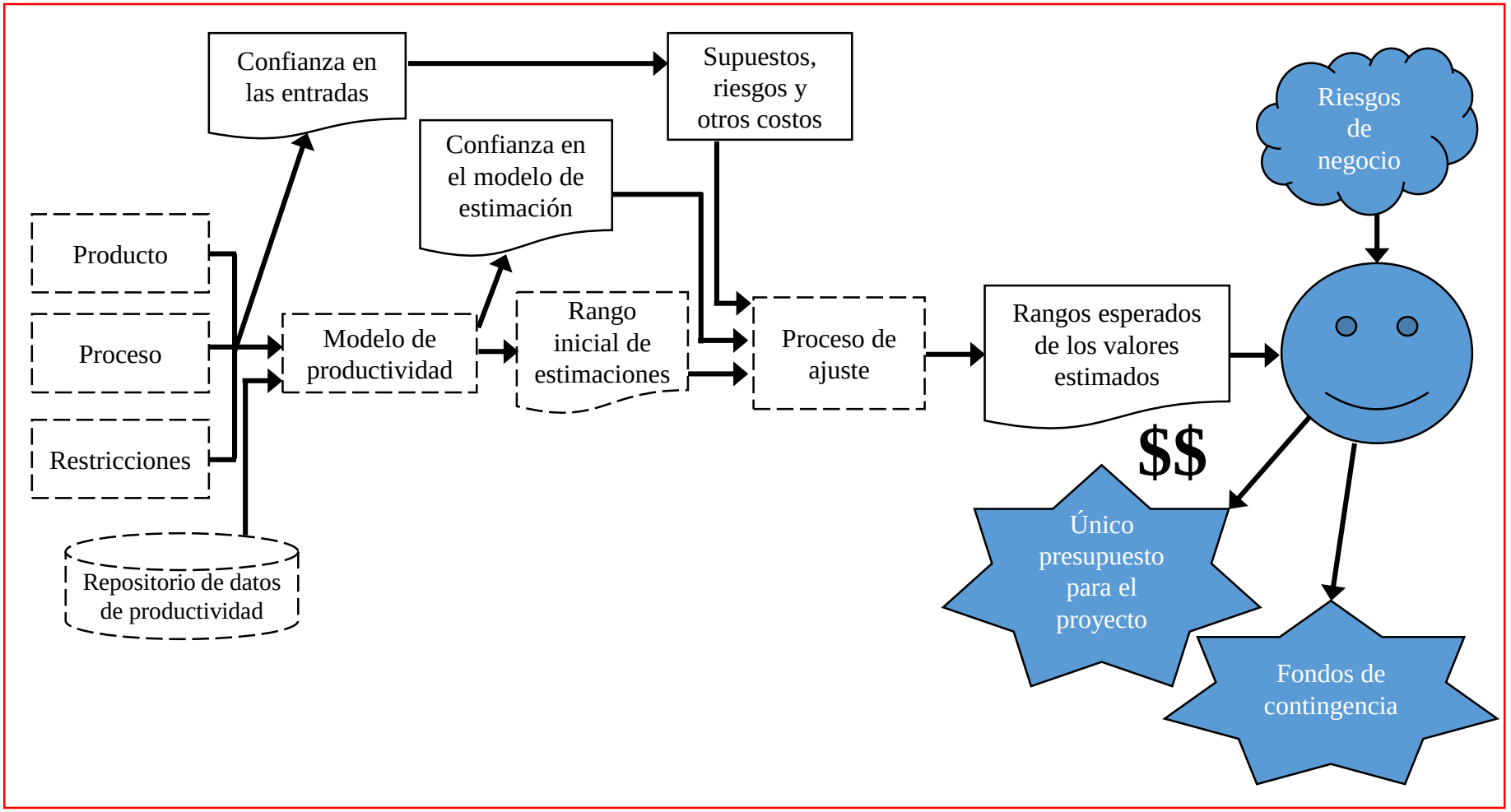
Fase B. Ejecución del modelo de productividad

Fases de la estimación



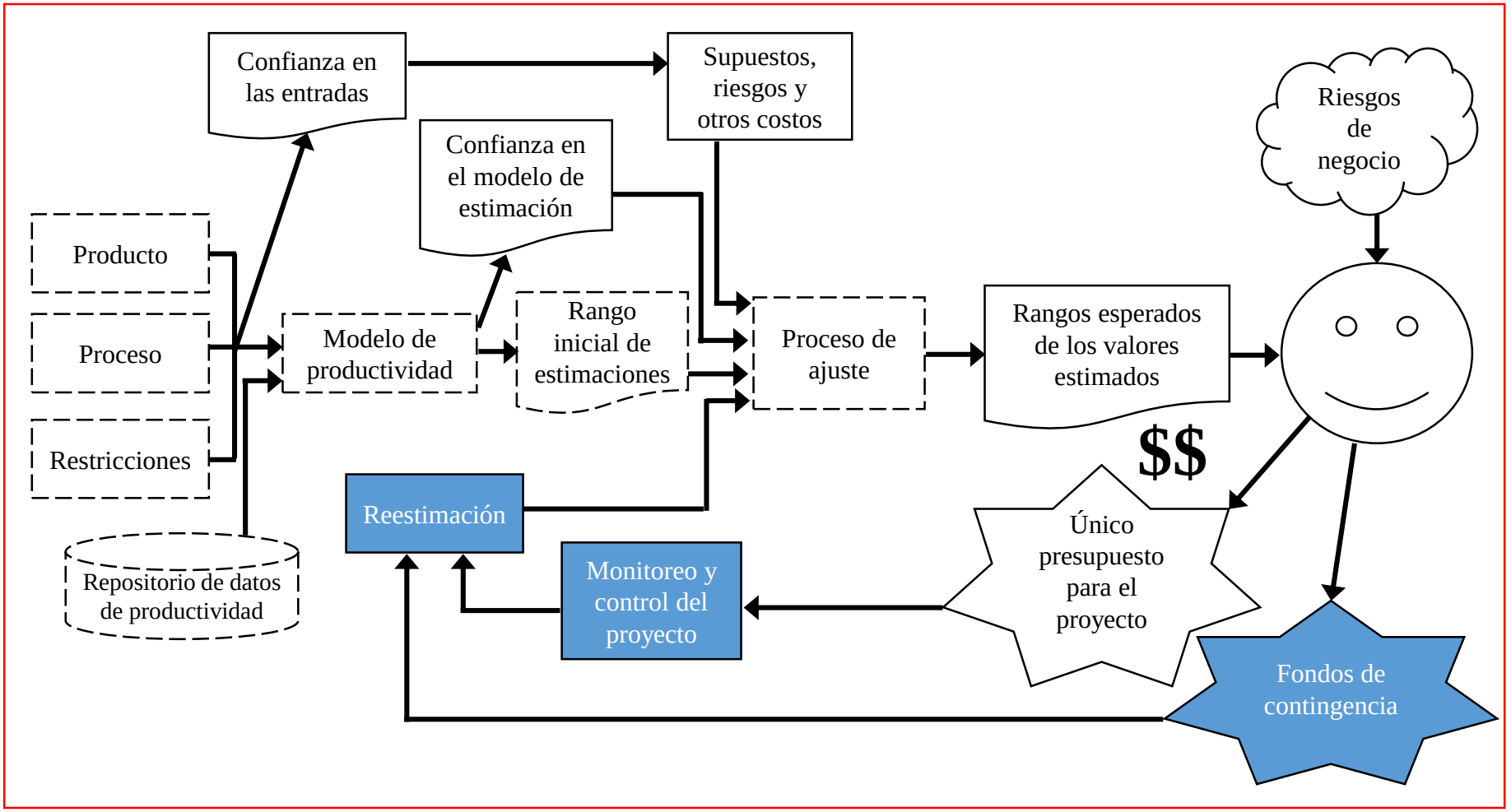
Fase C. El proceso de ajuste

Fases de la estimación



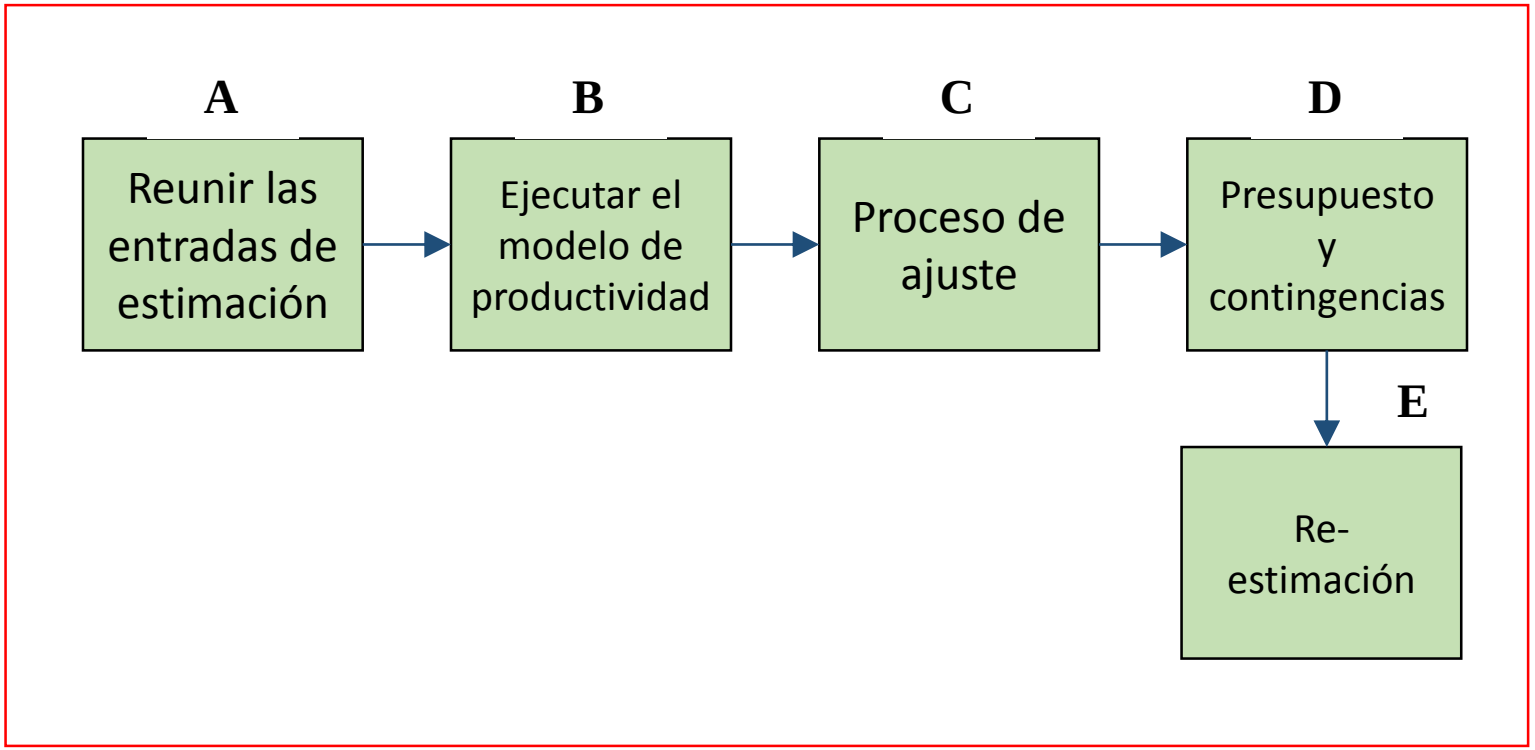
Fase D. Decisión sobre el presupuesto

Fases de la estimación



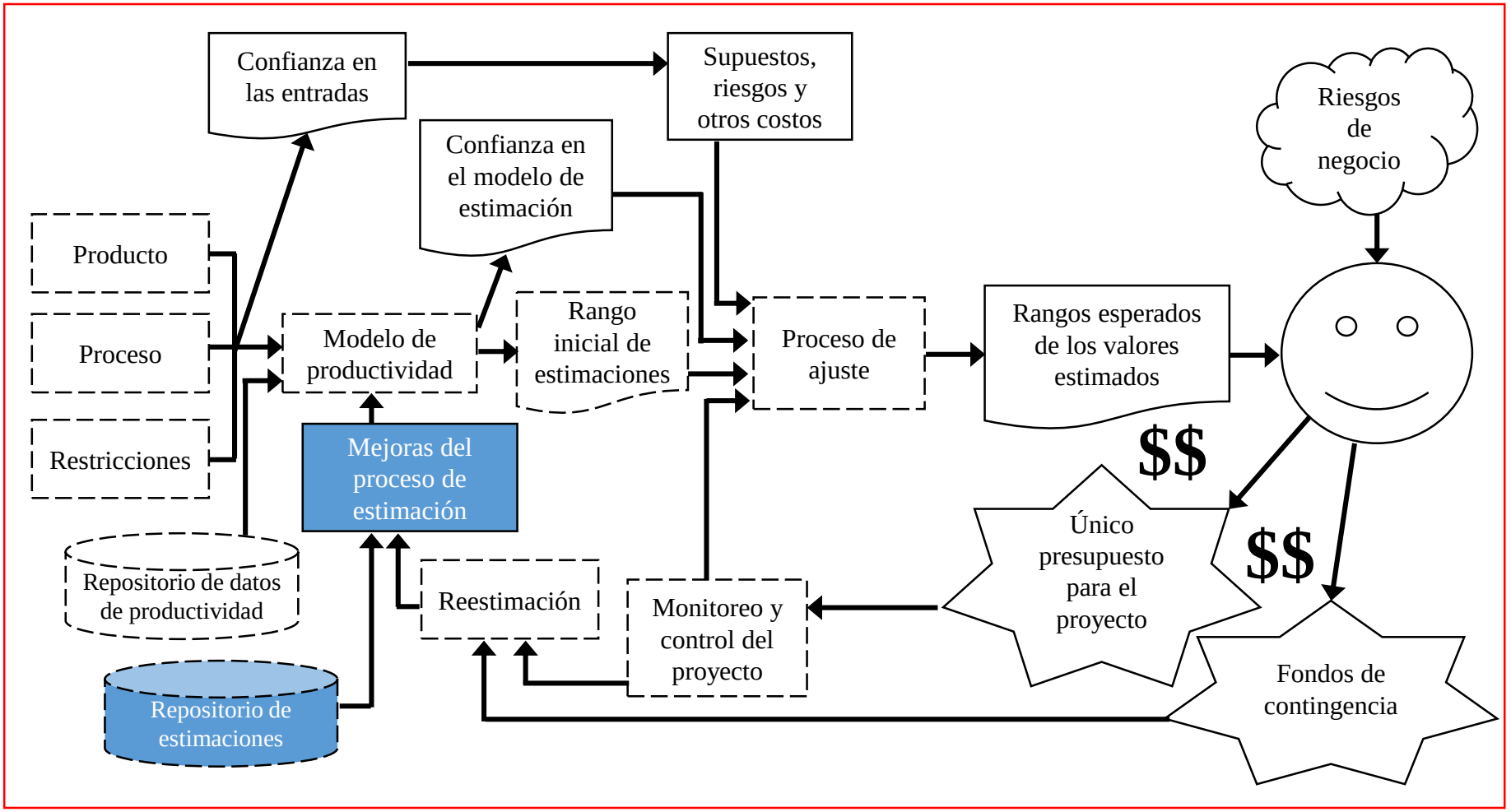
Fase E. Re-estimación

Fases de la estimación



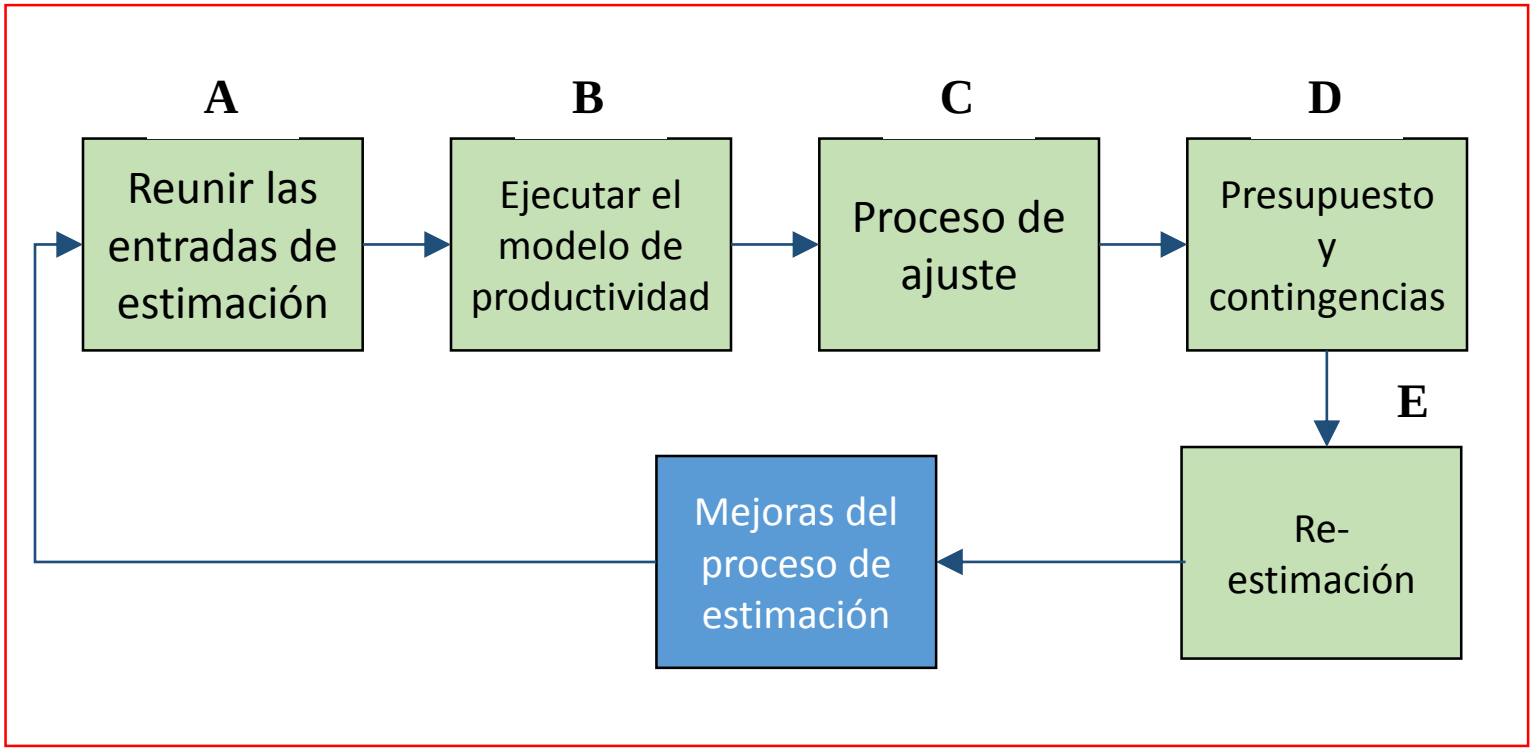
El proceso de estimación

Fases de la estimación



Fase G. Mejoras del proceso de estimación

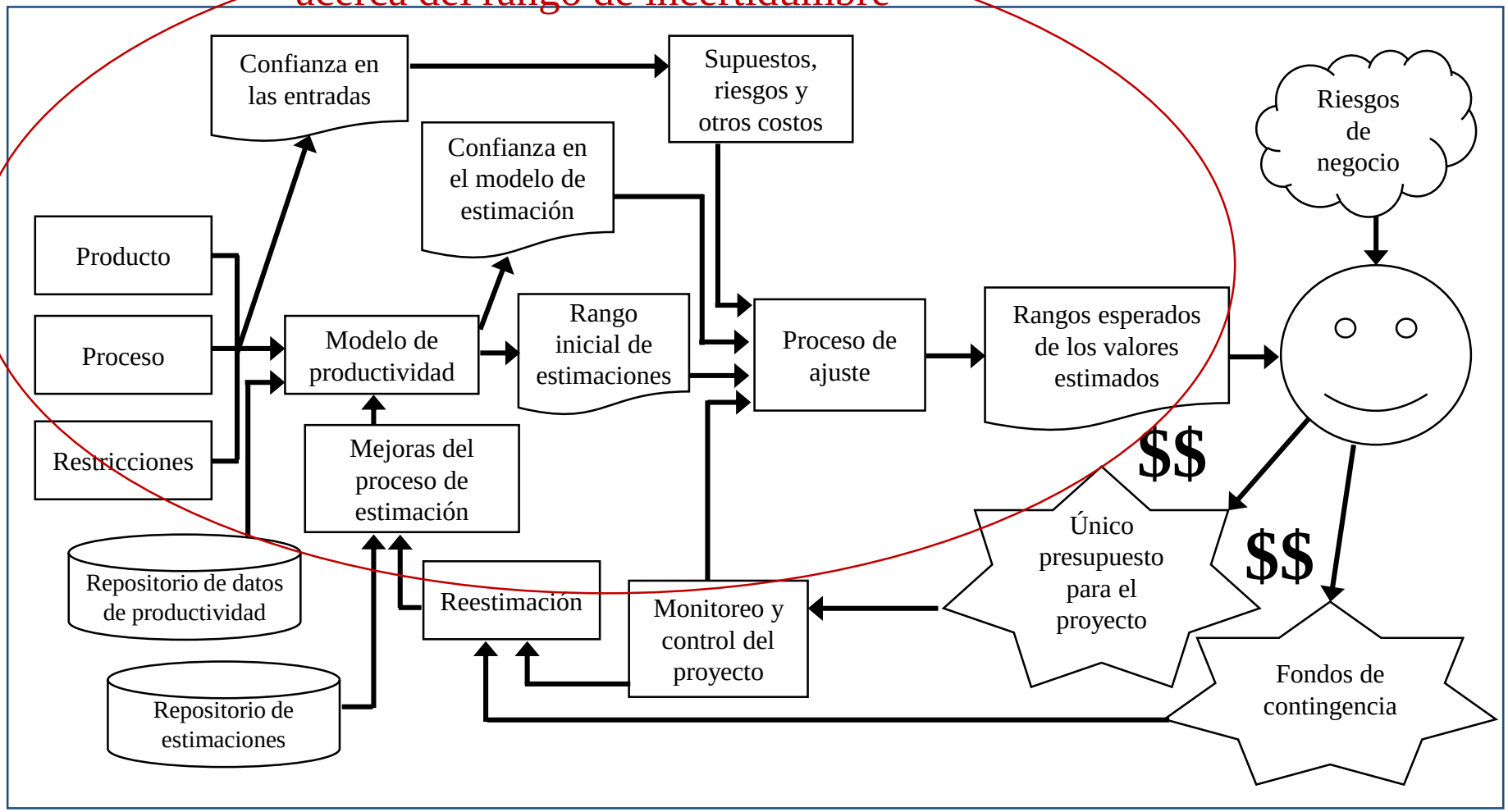
Fases de la estimación



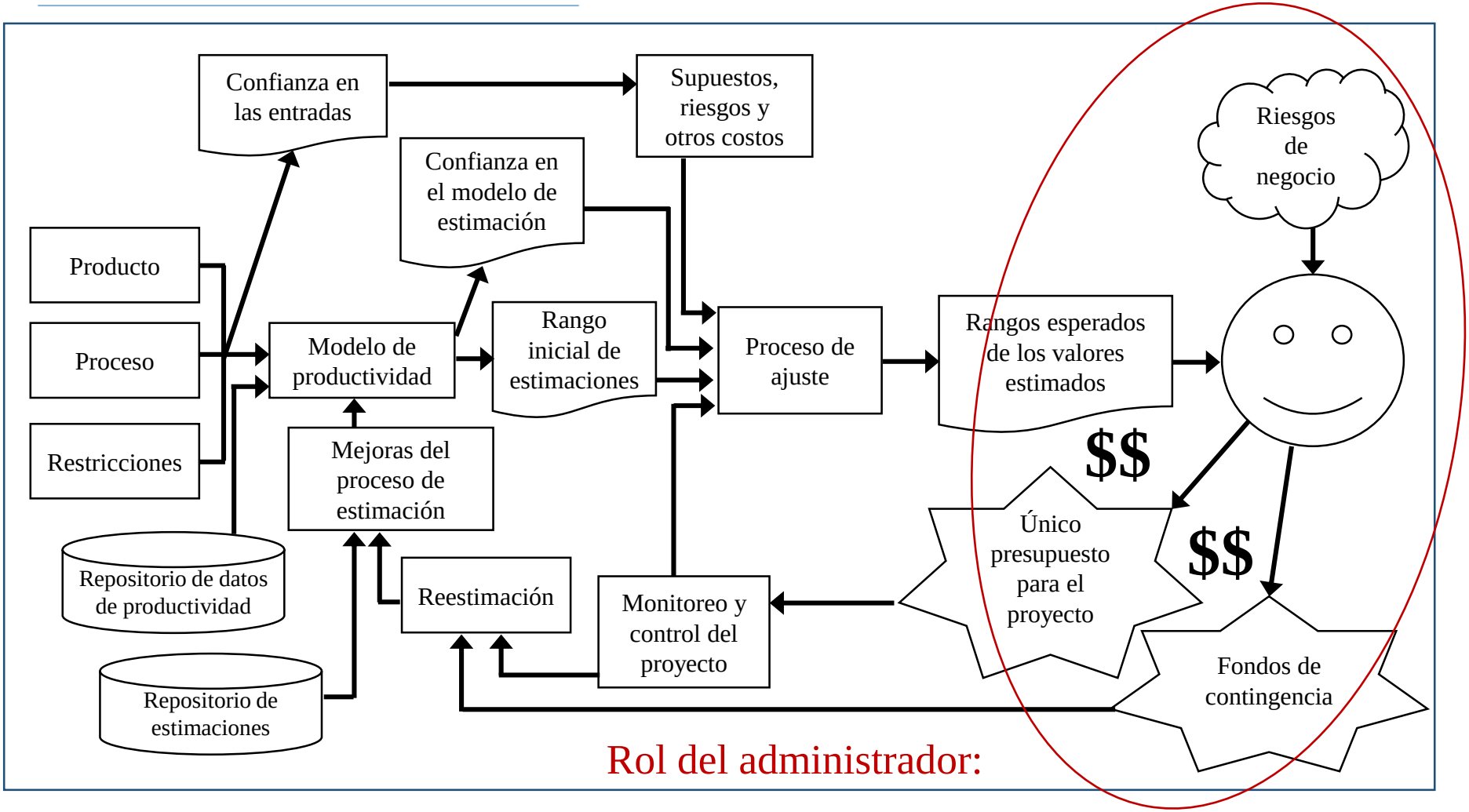
Fase F. Ciclo de retroalimentación de la estimación

Fases de la estimación

Rol del estimador: Proporcionar información acerca del rango de incertidumbre



Fases de la estimación



Rol del administrador:

elegir un número y administrar el riesgo

Facultad de Ciencias

Contacto

- Jesús Iván Saavedra Martínez
- jism@ciencias.unam.mx

Métricas de Software

Modelos de Estimación de Software

Jesús Iván Saavedra Martínez

Octubre 2017

Agenda

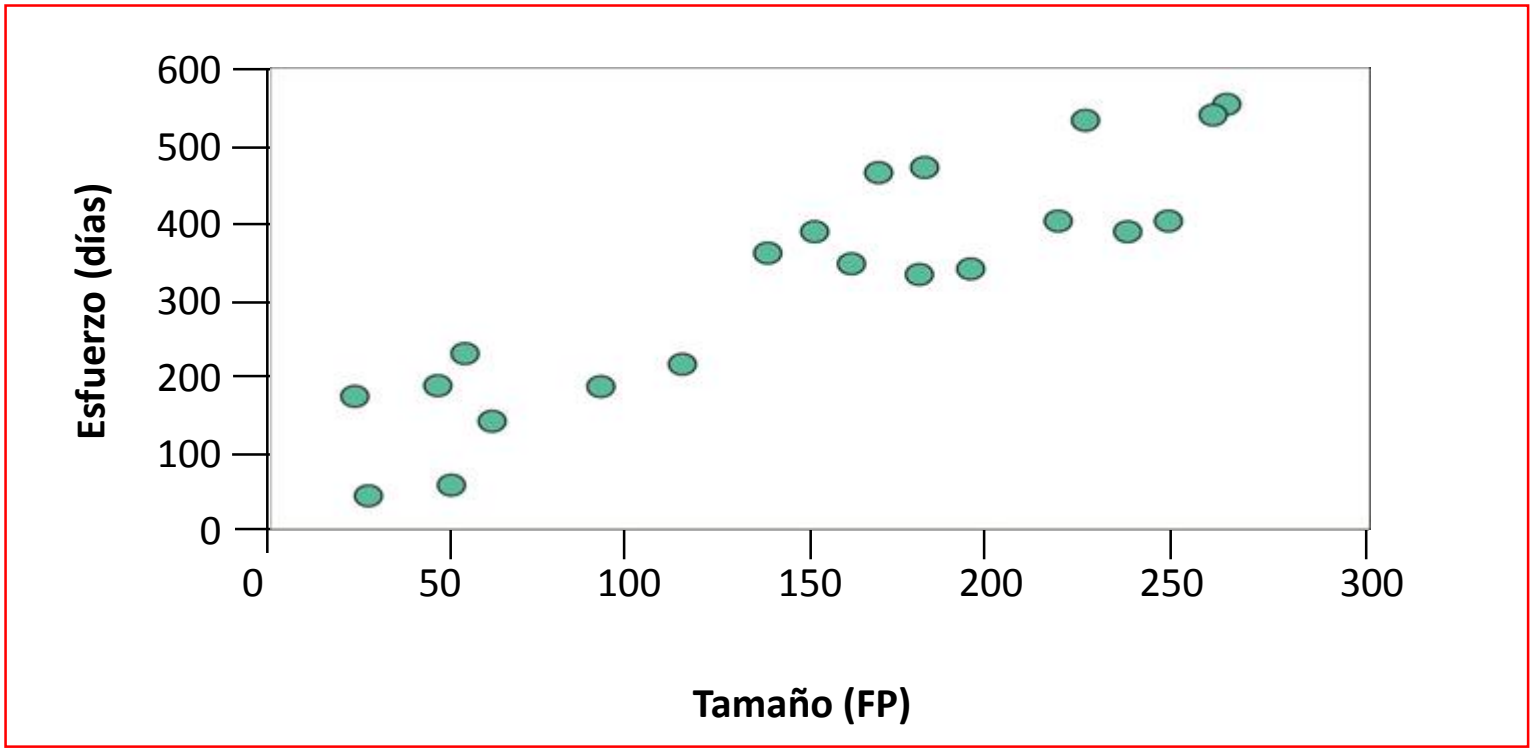
I. Conceptos económicos para modelos de estimación

II. Criterios de calidad para modelos de estimación

III. Conjunto de datos para modelos de estimación

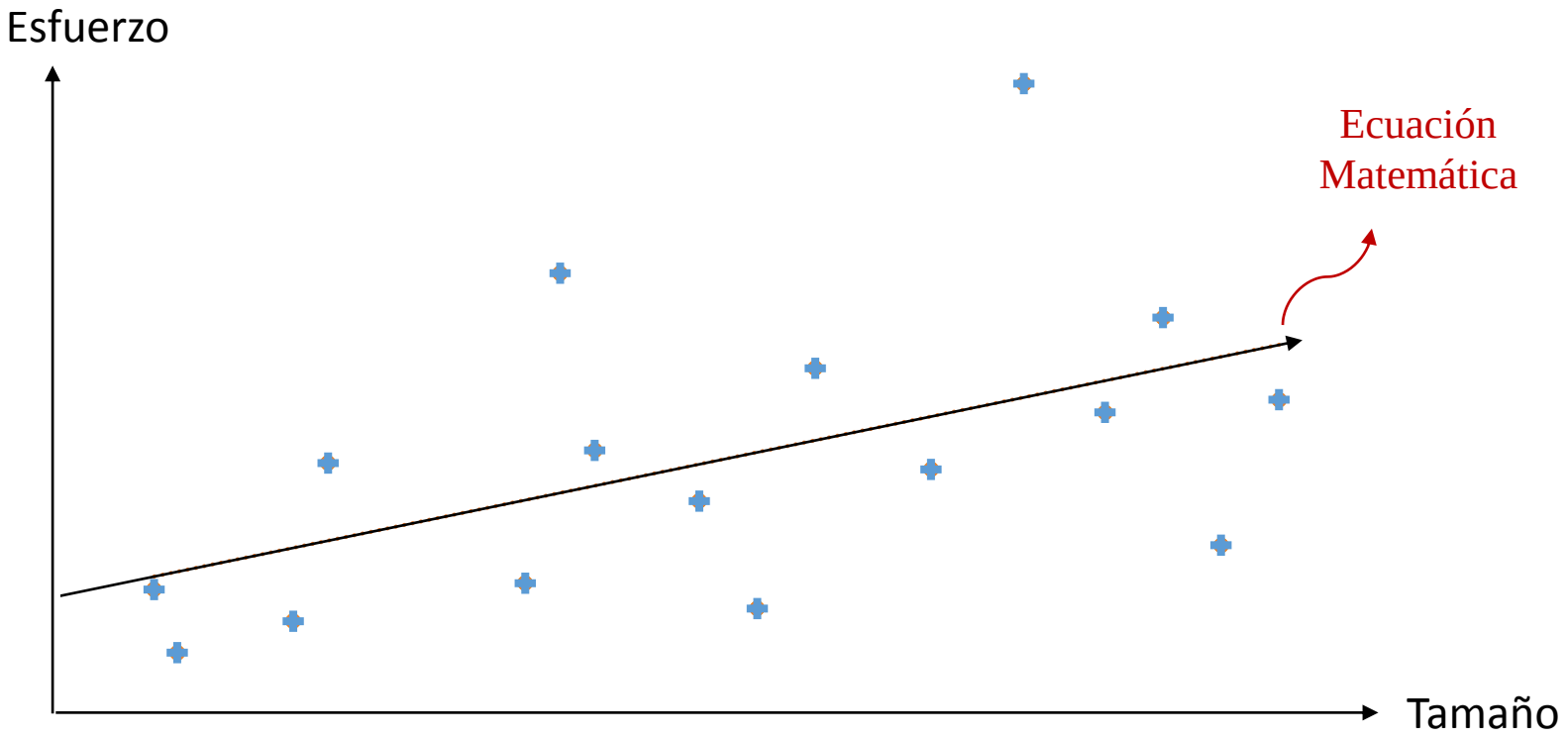
IV. Ejemplo de generación de un modelo de estimación

Conceptos económicos para modelos de estimación



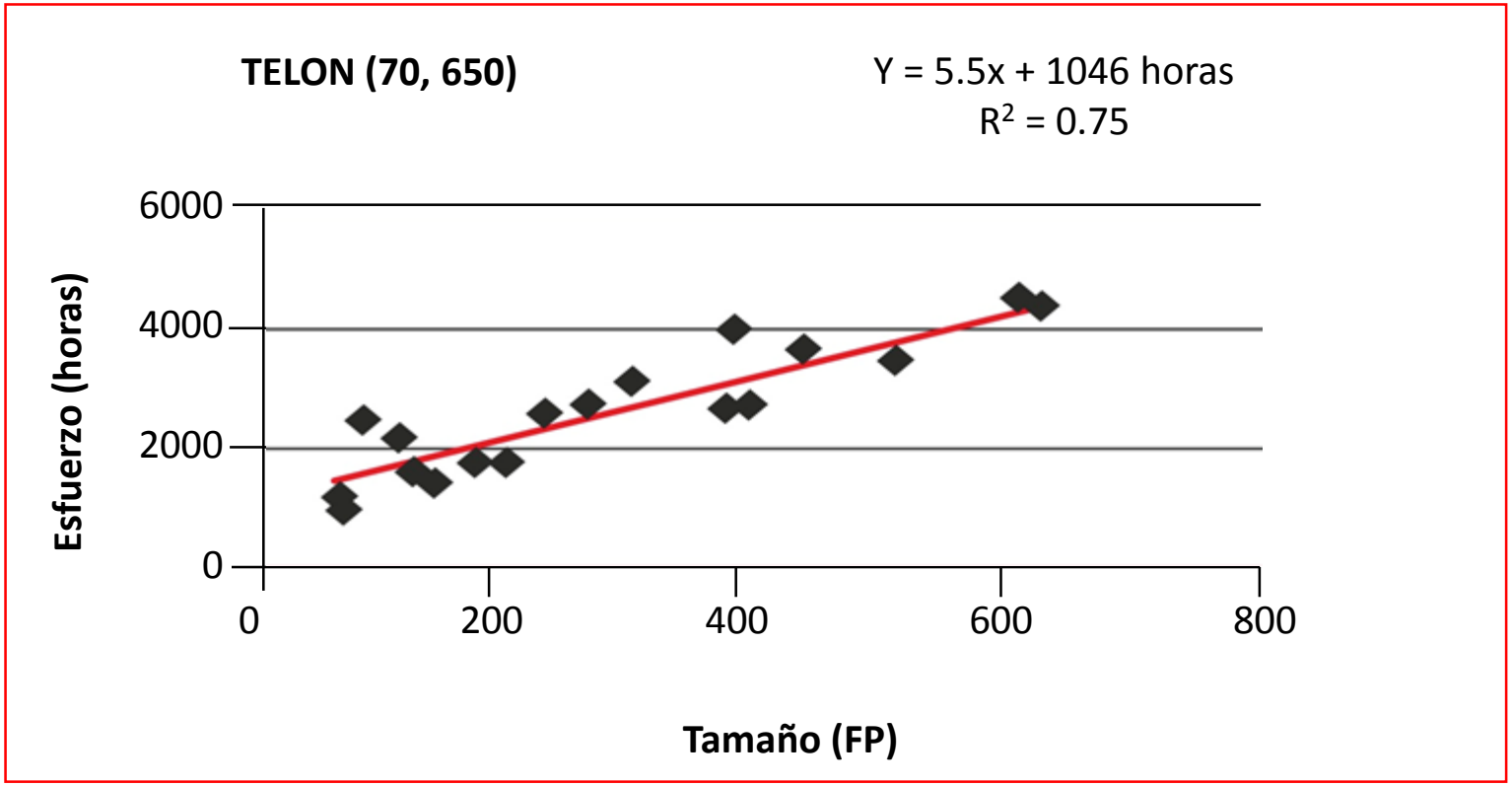
Conjunto de datos homogéneo de 21 proyectos (Abran 1994)

Conceptos económicos para modelos de estimación



Un modelo de productividad con una variable independiente

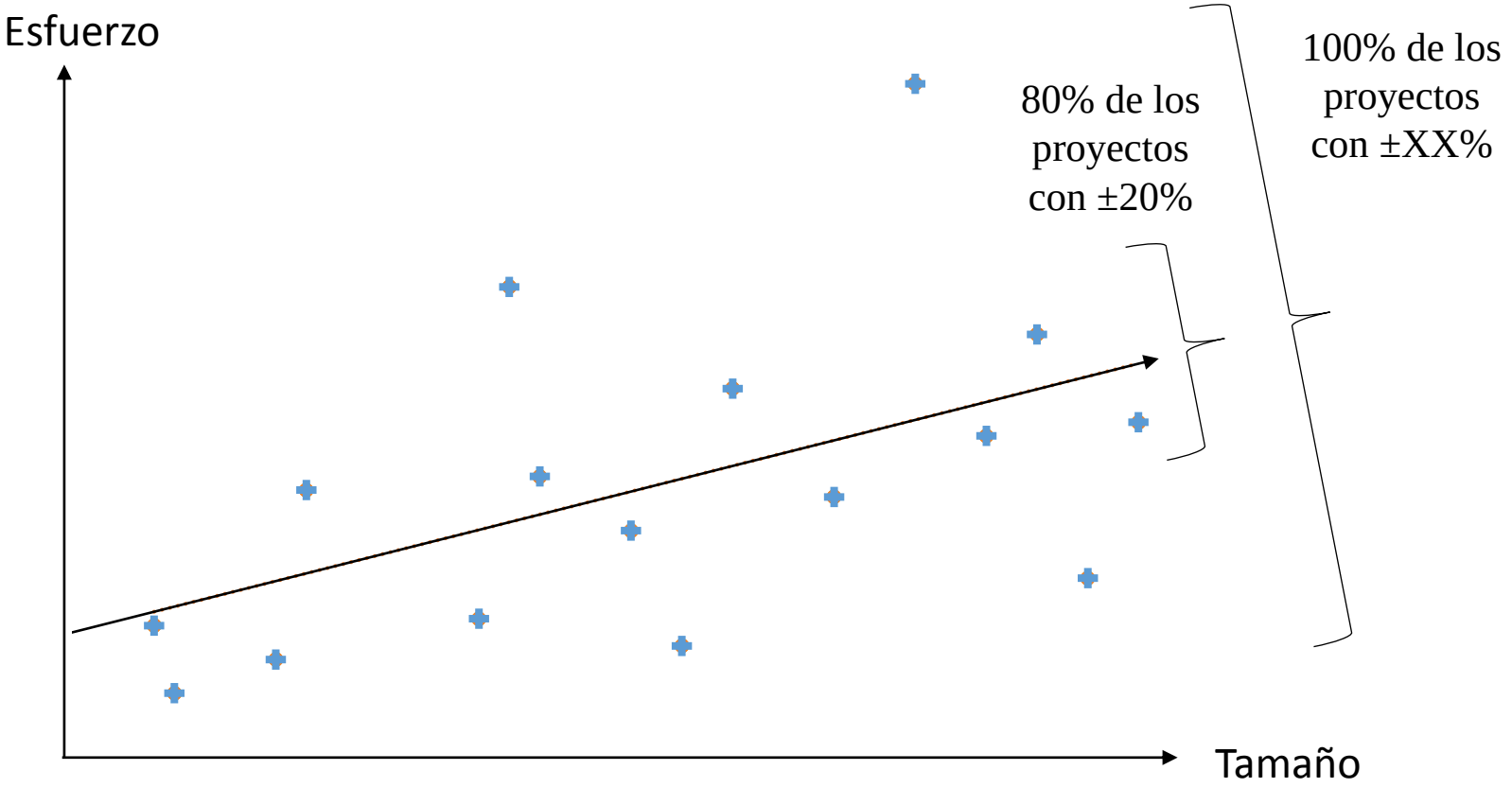
Conceptos económicos para modelos de estimación



El conjunto de datos TELON en el lanzamiento de ISBSG 1999

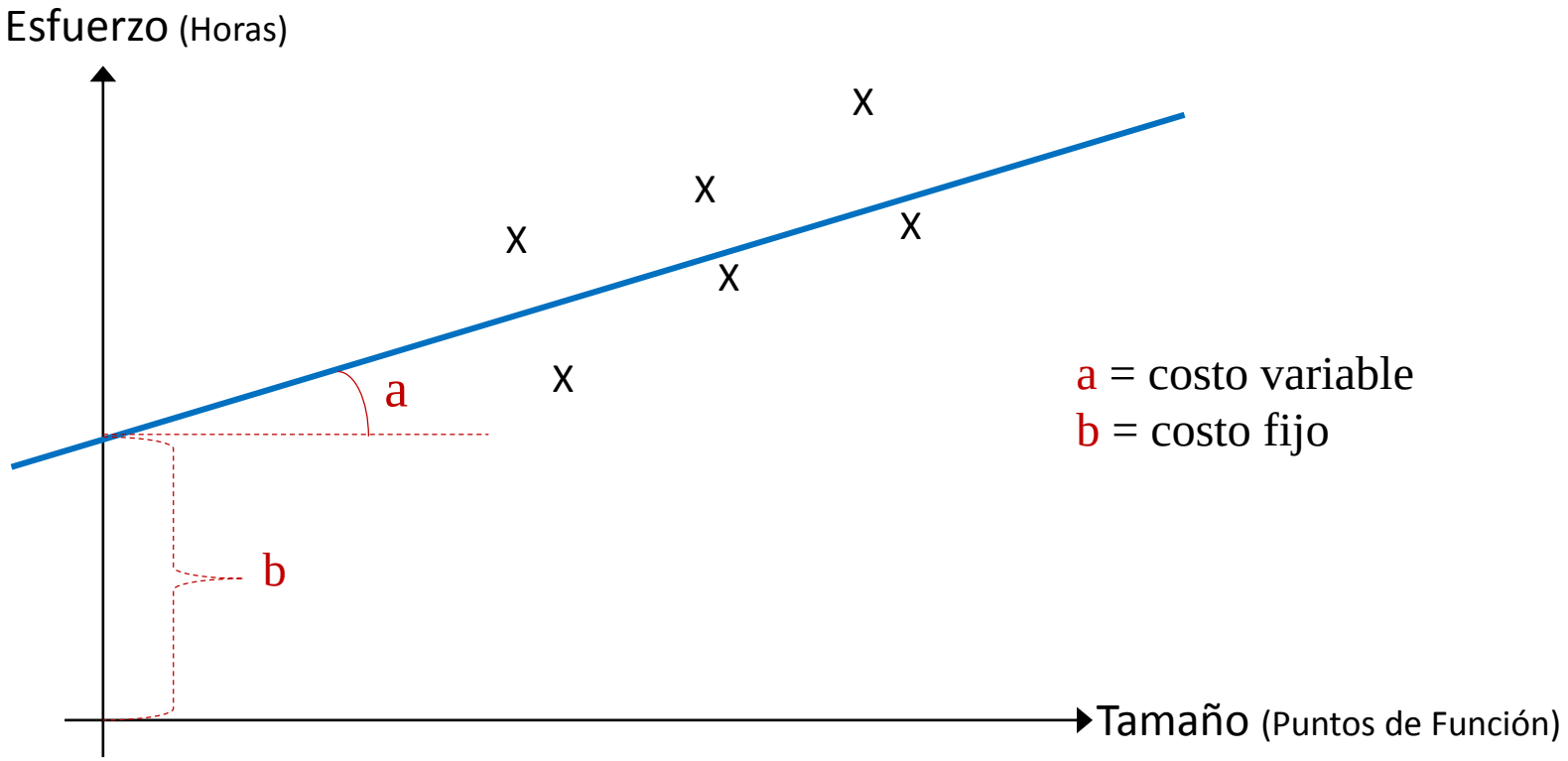
(Abran, Ndiaye, Bourque, 2007)

Conceptos económicos para modelos de estimación



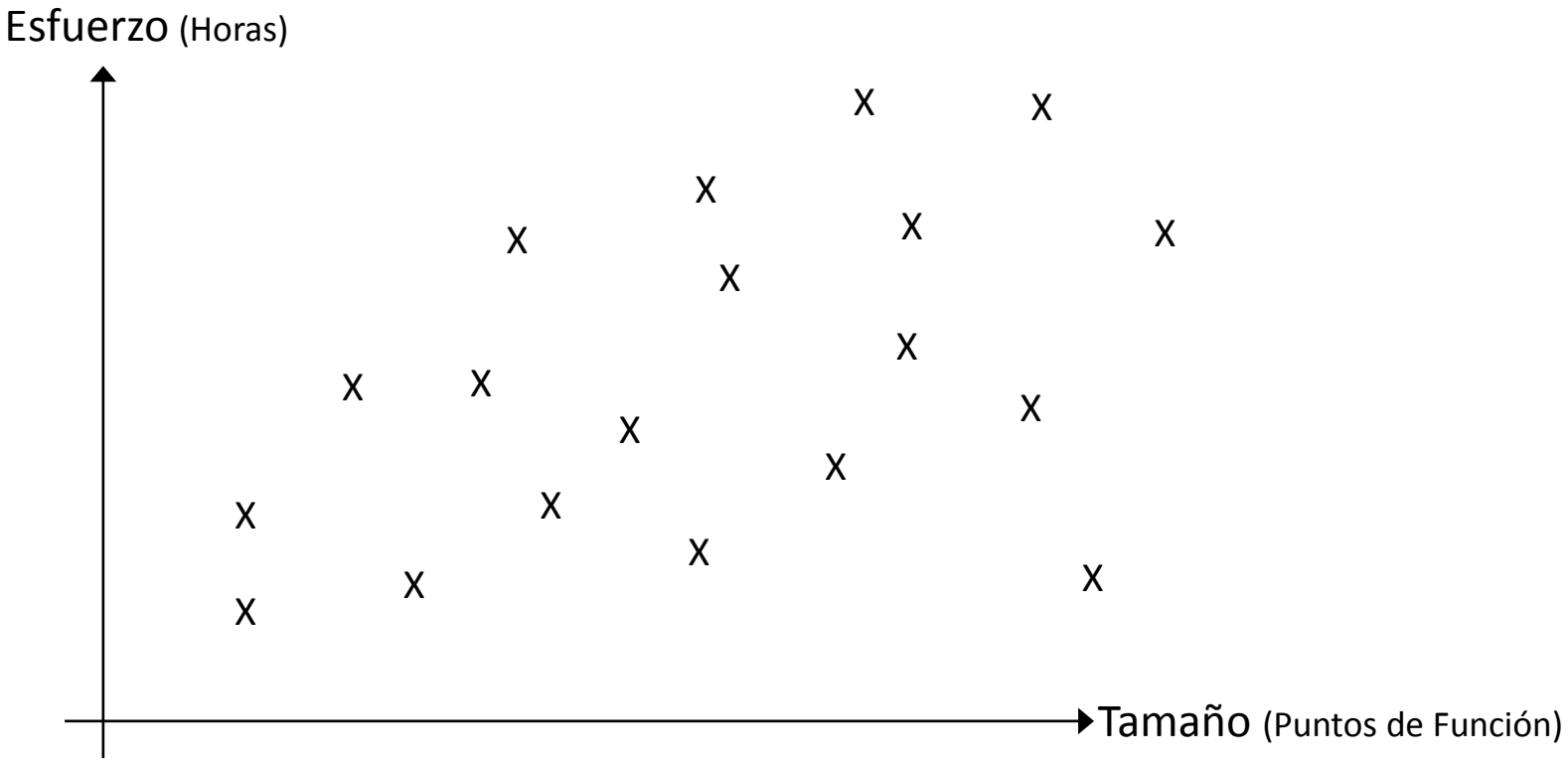
Una precisión del modelo de productividad

Conceptos económicos para modelos de estimación



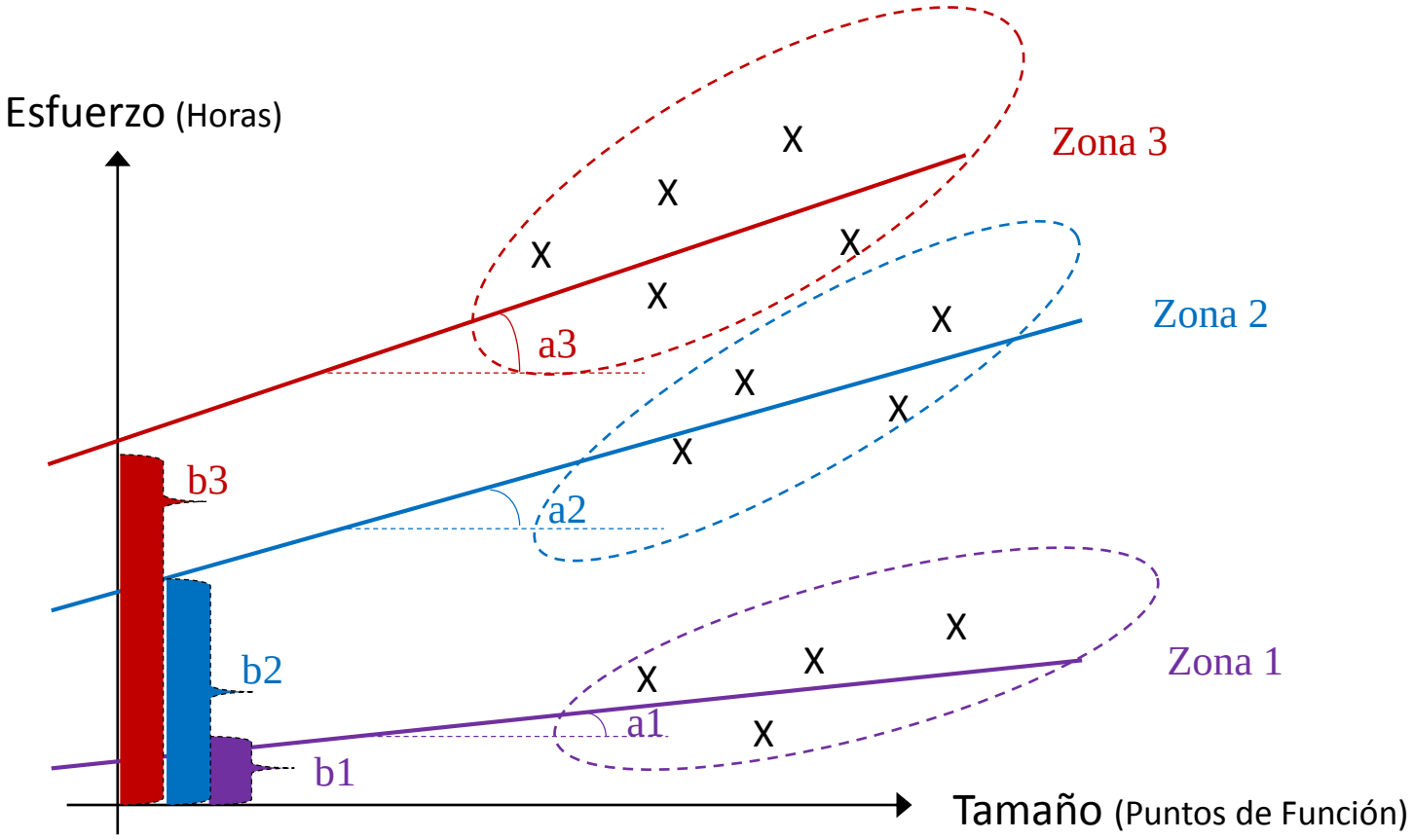
Modelo con costo variable y costo fijo

Conceptos económicos para modelos de estimación



Conjunto de datos cuneiforme en Ingeniería de Software

Conceptos económicos para modelos de estimación



Conjunto de datos cuneiforme con 3 subconjuntos con economías/deseconomías de escala

Facultad de Ciencias

Agenda

I. Conceptos económicos para modelos de estimación

II. Criterios de calidad para modelos de estimación

III. Conjunto de datos para modelos de estimación

IV. Ejemplo de generación de un modelo de estimación

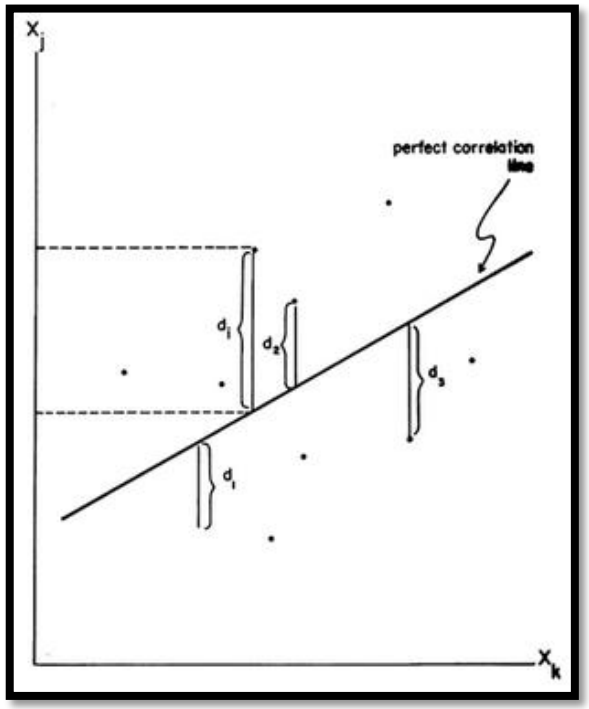
Criterios de Calidad par Modelos de Estimación

- Magnitude of Relative Error (MRE)

$$MRE = \left| \frac{Actual - Estimado}{Actual} \right|$$

$$\%MRE = 100\ x \left| \frac{Actual - Estimado}{Actual} \right|$$

Mean Magnitude of Relative Error (MMRE)
Median Magnitude of Relative Error (MdMRE)



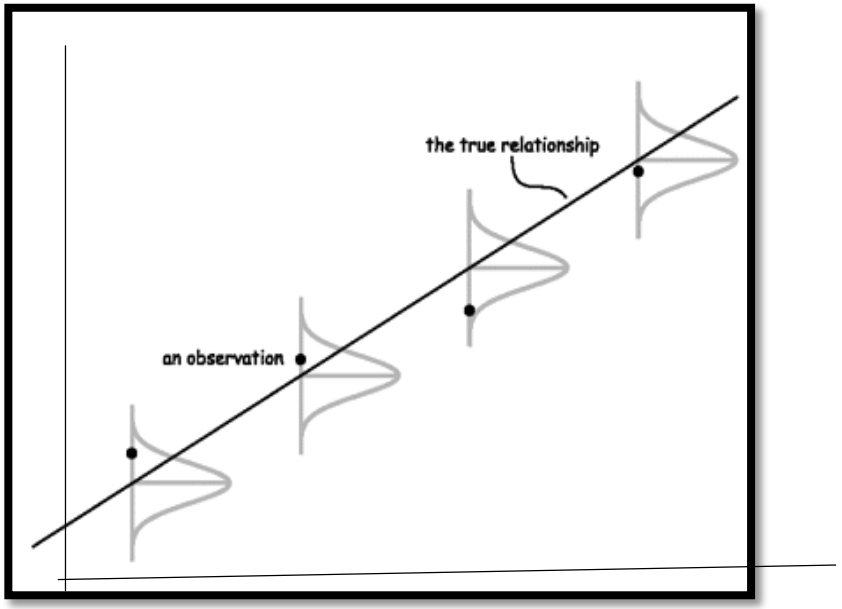
Criterios de Calidad par Modelos de Estimación

- Standard Deviation of MRE (SDMRE defined as the root of the mean square error (RMS))

$$RMS = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (Actual\ i - Estimado\ i)^2}$$

- Relative RMS

$$\overline{RMS} = \frac{RMS}{\frac{1}{n} \sum_{i=1}^{n} Actual\ i}$$



Nivel de predicción

- Prediction level Pred

$$Pred(l) = \frac{K}{N}$$

- Ejemplo *Pred(25%)* = 21% (18 de 84 proyectos tuvieron un MRE <= 25%)

Facultad de Ciencias

Agenda

I. Conceptos económicos para modelos de estimación
II. Criterios de calidad para modelos de estimación
III. Conjunto de datos para modelos de estimación
IV. Ejemplo de generación de un modelo de estimación

Facultad de Ciencias

Conjunto de datos para modelos de estimación

ID de Proyecto	Tamaño Funcional CFP v4.0.1	Esfuerzo HH
1	171	1,763
2	216	2,009
3	48	403
4	569	4,498
5	229	1,669
6	354	3,436
7	154	1,497
8	181	2,190
9	433	4,592
10	288	1,900
11	101	1,216
12	455	3,318
13	143	1,460
14	509	4,731
15	749	5,646
16	183	1,116
17	112	1,235
18	607	4,918
19	186	2,083
20	368	2,321

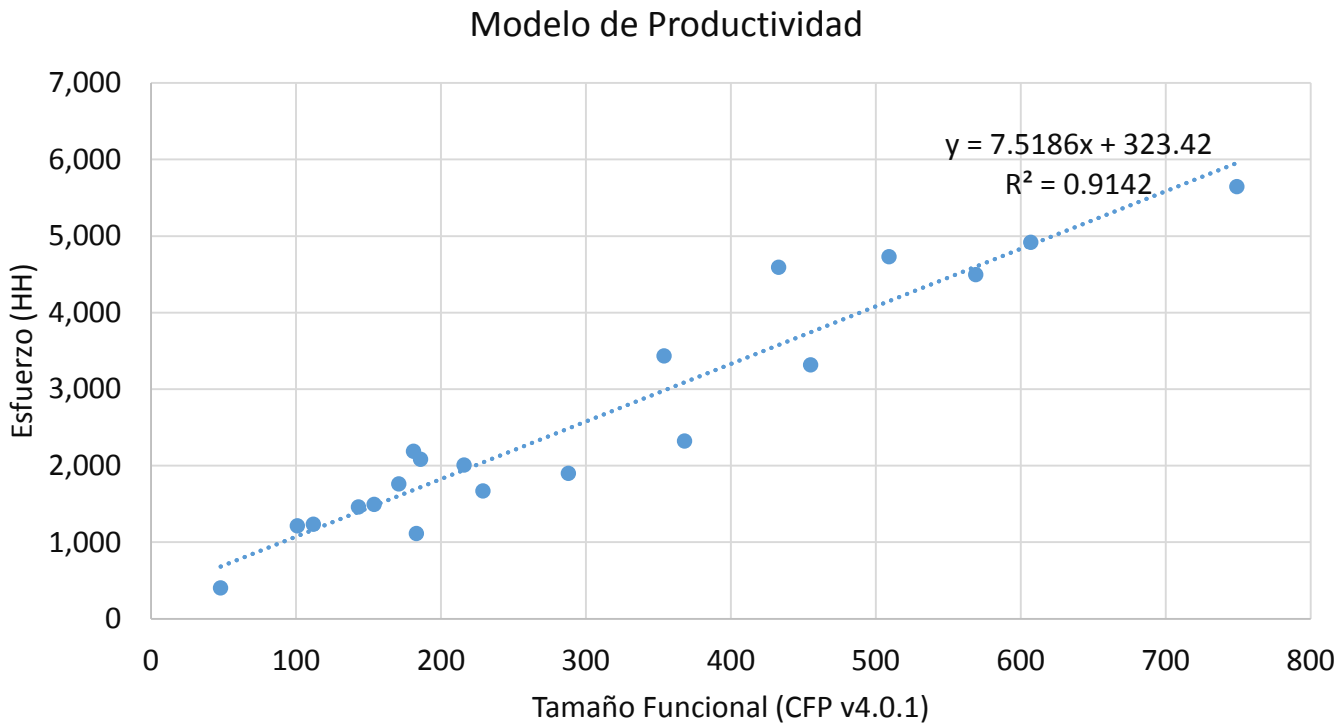
ID de Proyecto	Tamaño Funcional CFP v4.0.1	Esfuerzo HH
21	283	3,202
22	213	1,473
23	67	798
24	83	522
25	49	488
26	124	954
27	467	4,718
28	41	464
29	420	3,821
30	188	1,428
31	464	4,224
32	545	4,686
33	195	1,973
34	384	3,187
35	596	4,467
36	234	1,498
37	158	1,514
38	319	2,140
39	81	682
40	373	2,611

Agenda

I. Conceptos económicos para modelos de estimación
II. Criterios de calidad para modelos de estimación
III. Conjunto de datos para modelos de estimación
IV. Ejemplo de generación de un modelo de estimación

Modelo de Productividad

- Para la generación del modelo de productividad con regresión lineal se utilizará el 50% de la muestra (los primeros 20 proyectos), el otro 50% se utilizará para evaluar la calidad del modelo.



Evaluación de criterios de calidad

- Una vez generado el modelo de productividad, el otro 50% se utilizará para evaluar la calidad del modelo.

ID de Proyecto	CFP v4.0.1	Esfuerzo Real HH	Esfuerzo Estimado HH	MRE
21	283	3,202	2,453	23%
22	213	1,473	1,928	31%
23	67	798	823	3%
24	83	522	946	81%
25	49	488	694	42%
26	124	954	1,254	31%
27	467	4,718	3,835	19%
28	41	464	632	36%
29	420	3,821	3,480	9%
30	188	1,428	1,736	22%
31	464	4,224	3,813	10%
32	545	4,686	4,419	6%
33	195	1,973	1,792	9%
34	384	3,187	3,210	1%
35	596	4,467	4,801	7%
36	234	1,498	2,083	39%
37	158	1,514	1,509	0.004%
38	319	2,140	2,724	27%
39	81	682	933	37%
40	373	2,611	3,127	20%
			MMRE	**22.65%**
			SDMRE	**19.24%**
			PRED(25%)	**60%**

Modelo de Estimación

- Para obtener el *esfuerzo estimado* de cada proyecto utilizando el modelo de estimación, solo hay que sustituir en la ecuación de la recta la 'x' por el tamaño funcional y tener en cuenta los criterios de calidad del modelo de estimación.
  - Ecuación de la recta: ***y = 7.5182x + 322.93***

- Criterios de calidad:
  - ✓ ***MMRE = 22.65%***
  - ✓ ***SDMRE = 19.24***
  - ✓ ***PRED(25%) = 60%***

Facultad de Ciencias

Contacto

- Jesús Iván Saavedra Martínez
- jism@ciencias.unam.mx

Métricas de Software

Evaluación de Desempeño de Proyectos

Jesús Iván Saavedra Martínez

Octubre 2017

Facultad de Ciencias

Agenda

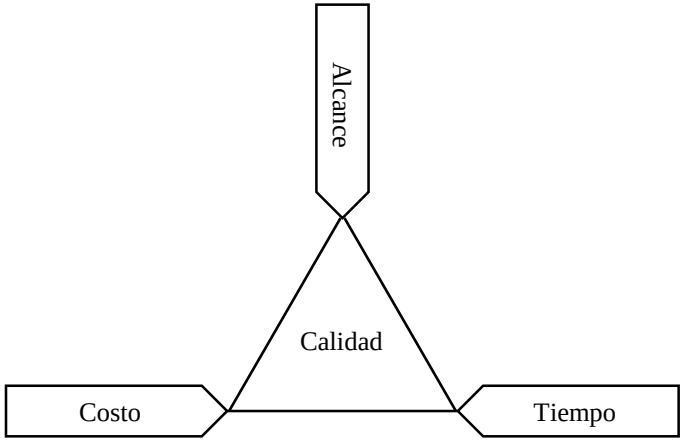
I. Earned Value Management

II. Earned Schedule

III. Earned Scope Management

Propuesta

- Todos los proyectos de software se realizan bajo restricciones.



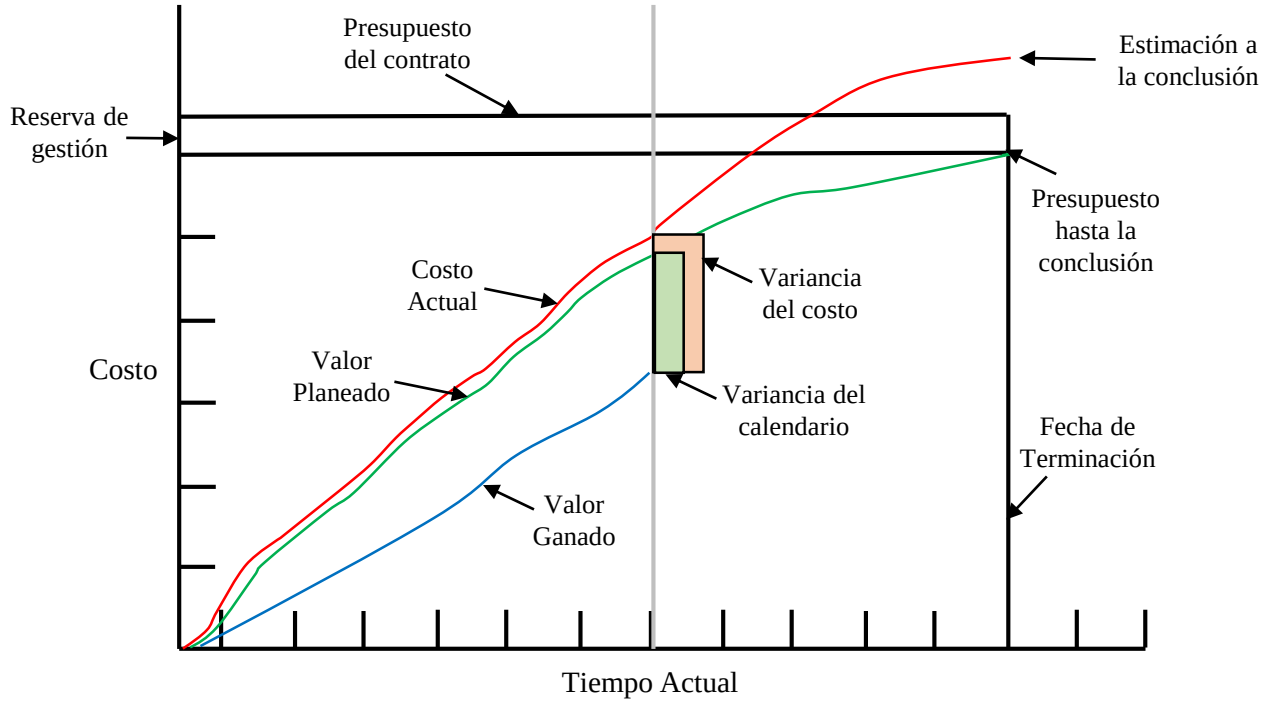
Triángulo de hierro

- Todas las organizaciones que desarrollan software deben de estimar proyectos de desarrollo de software para poder administrar apropiadamente sus recursos, para estimar entonces y administrar necesitamos medir.

Enfoque PMI

Control Cost: Inputs, Tools & Techniques, and Outputs

Inputs	Tools & Techniques	Outputs
1 Project management plan 2 Project founding requirements 3 Work performance data 4. Organizational process assets	1 Earned value management 2 Forecasting 3 To-complete performance index (TCPI) 4 Performance reviews 5 Project management software 6 Reserve analysis	1 Work performance information 2 Cost forecasts 3 Change requests 4 Project management plan updates 5 Project documents updates 6 Organizational process assets updates



Facultad de Ciencias

Earned Value Management (EVM)

Termología y Definición del Valor Ganado	Fórmula
Valor Planeado (Planned Value - PV) = Costo Presupuestado del Trabajo Planificado (Budgeted Cost of Work Scheduled - BCWS)	PV = BCWS
Costo Actual (Actual Cost - AC) = Costo Real del Trabajo Realizado (Actual Cost of Work Performed - ACWP)	AC = ACWP
Presupuesto hasta la Conclusión (Budget At Completion - BAC)	BAC = Presupuesto hasta la conclusión
Valor Ganado (Earned Value - EV)	EV = BAC (% completado)
Variación del Costo (Cost Variance - CV)	CV = EV –AC
Variación del Calendario (Schedule Variance - SV)	SV = EV – PV
Índice de Desempeño del Costo (Cost Performance Index - CPI)	CPI = EV / AC
Índice de Desempeño del Calendario (Schedule Performance Index - SPI)	SPI = EV / PV
Estimación a la Conclusión (Estimate AT Completion - EAC)	EAC = BAC / CPI
Estimación hasta la Conclusión (Estimate To Complete - ETC)	ETC = EAC –AC
Variación hasta la Conclusión (Variance At Completion - VAC)	VAC = BAC -EAC

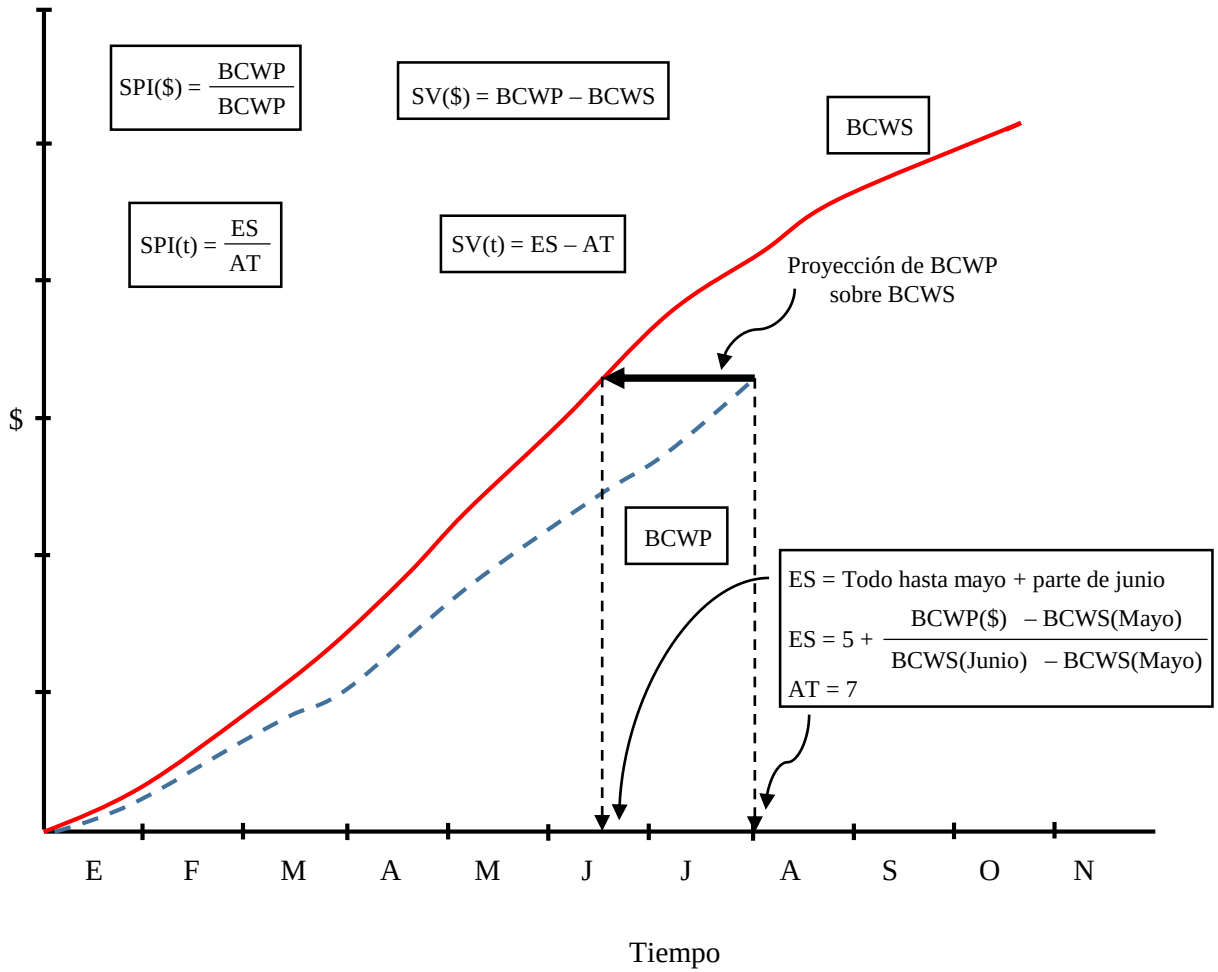
Agenda

I. Earned Value Management

II. Earned Schedule

III. Earned Scope Management

Earned Schedule (ES)



Earned Schedule (ES)

Métricas	Calendario Ganado (Earned Schedule - ES)	ES
	Tiempo Actual (Actual Time - AT)	AT
	Duración Planeada (Planned Duration - PD)	PD
Indicadores	Variación de Calendario (Schedule Variance - SV)	SV(t) = ES – AT
	Índice de Desempeño del Calendario (Schedule Performance Index - SPI)	SPI(t) = ES / AT
	Índice de Desempeño del Calendario hasta la Conclusión (To Complete Schedule Performance Index - TSPI)	TSPI = (PD - ES) / (PD - AT)
Predictores	Estimado Independiente hasta la Conclusión (Independent Estimate at Completion - IEAC)	IEAC(t) = PD / SPI(t)
	Variación hasta la Conclusión (Variance At Completion - VAC)	VAC(t) = PD – IEAC(t)

Facultad de Ciencias

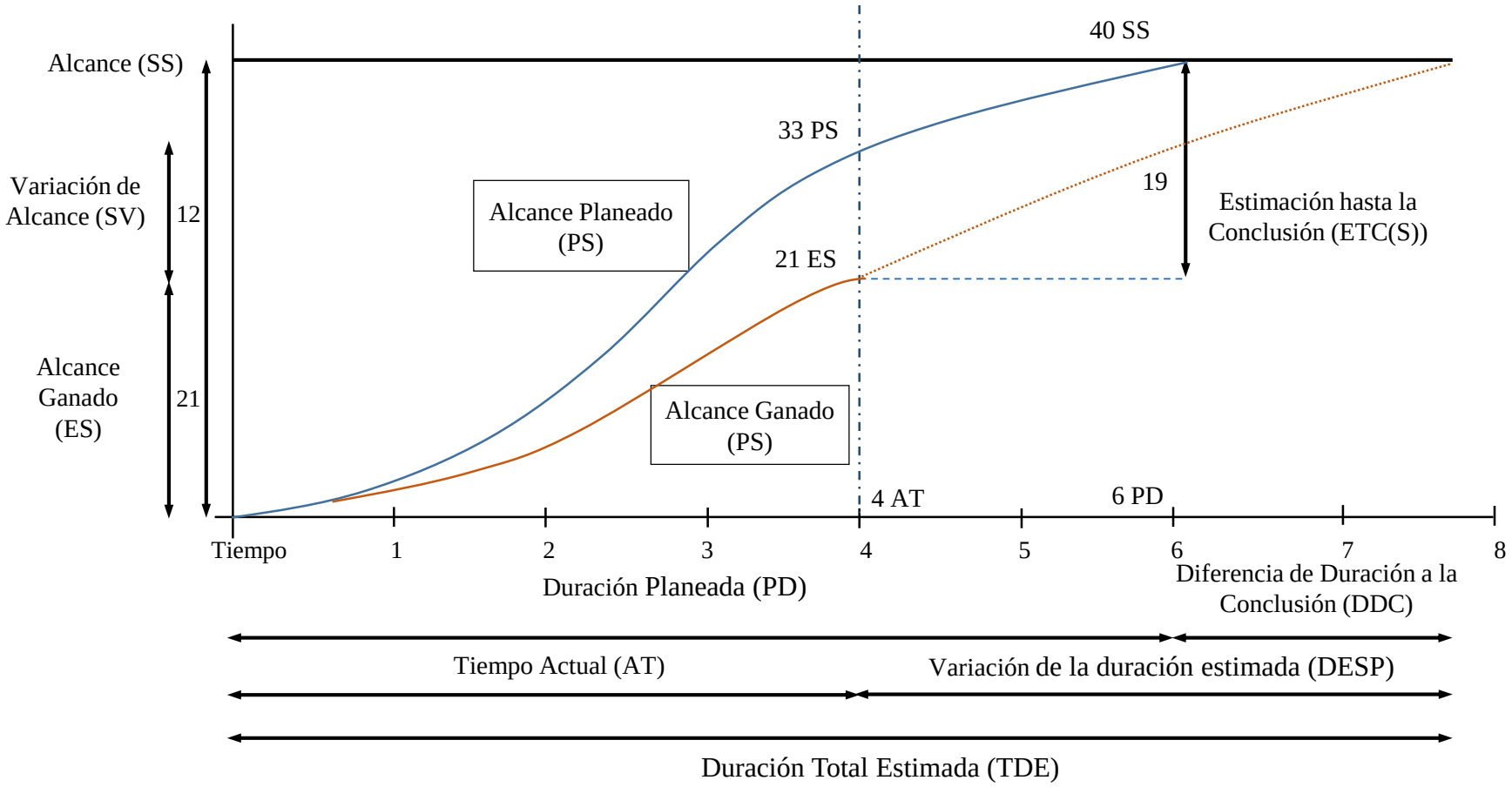
Agenda

I. Earned Value Management

II. Earned Schedule

III. Earned Scope Management

Earned Scope Management (ESM)



Earned Scope Management (ESM)

Datos de Entrada	Tiempo Actual (Actual Time - AT)	AT = Número de periodos ejecutados
	Duración Planeada (Planned Duration - PD)	PD = Duración planeada del proyecto
	Alcance del proyecto (SS)	SS = Lo que se ha planeado realizar
	Alcance Ganado (Earned Scope - ES)	Si el %C es un dato de entrada entonces: ES = SS (% completado)
	Alcance Planeado (Planned Scope (PS)	PS = Lo que se ha planeado hacer
	% completado (%C)	Si el ES es un dato de entrada. %C = ES / SS
	Recursos Humanos usados en el Proyecto (Project Human Resources - PHR)	PHR = Recursos humanos usados en el proyecto

Earned Scope Management (ESM)

Estado del Alcance	Variación de Alcance (Scope Variance - SV)	SV = ES – PS
	Índice de Desempeño del Alcance (Scope Performance Index - SPI)	SPI = ES / PS
	Productividad por Recurso (Productivity by Resource - PR)	PR = ES / PHR
	Productividad hasta la Conclusión (Productivity to Complete - PTC)	PTC = (PSn-1 – PSn) / PHR
	Productividad Promedio por Recurso Humano (Average Productivity by Human Resource - PROAVG)	PROAVG = AVG(PRn)
	Variación de la Productividad (Productivity Variation - PV)	PV = PR - PTC
	Estimación hasta la Conclusión (Estimate To Complete – ETC(s))	ETC(s) = SS - ES

Earned Scope Management (ESM)

Predicción del Alcance	Duración Total Estimada (Total Duration Estimate - TDE)	TDE = AT + ETC(s) / (PR * PHR)
	Variación de la Duración Estimada. (Estimate Duration Variation - DESP)	DESP = TDE - AT
	Diferencia de Duración a la Conclusión (Duration Difference at Completion – DDC)	DDC = PD - TDE
	Productividad Requerida por Recursos definida para Completar el Proyecto como se Planeó (Productivity required by resources defined to complete the scope as was planned - PRTC).	PRTC = (ETC / PHR) / (PD - AT)
	Variación de Recursos para completar el Alcance planeado con la misma Productividad (Resources Variation to Complete Planed Scope by period - RVTC)	RVTC = (-SV) / PR
	Recursos Humanos Necesarios para Completar el Proyecto (Human Resources Need to Complete the Project - RNTC)	RNTC = RVTC + PHR

Facultad de Ciencias

Contacto

- Jesús Iván Saavedra Martínez
- jism@ciencias.unam.mx

Métricas de Software

Ejemplos Evaluación de Desempeño de Proyectos

Jesús Iván Saavedra Martínez

Octubre 2017

Facultad de Ciencias

Agenda

I. Ejemplo Earned Value Management

II. Ejemplo Earned Schedule

III. Ejemplo Earned Scope Management

Ejemplo EVM

- Tenemos un proyecto para ejecutar en 4 semanas y el presupuesto es de $100,000.
- Nos informan que al finalizar la tercera semana se ha completado sólo el 50% del trabajo, de acuerdo al calendario se debía haber realizado el 75%, también que los gastos actuales ascienden a $90,000.

➢ ¿Cuál será el costo total del proyecto?

Facultad de Ciencias

Earned Value Management (EVM)

Termología y Definición del Valor Ganado	Fórmula
Valor Planeado (PV)	75% x $100,000 = $75,000
Costo Actual (AC)	$90,000
Presupuesto hasta la Conclusión (BAC)	$100,000
Valor Ganado (EV)	50% x $100,000 = $50,000
Variación del Costo (CV)	$50,000 - $90,000 = -$40,000
Variación del Calendario (SV)	$50,000 - $75,000 = $25,000
Índice de Desempeño del Costo (CPI)	$50,000 / $90,000 = 0.56
Índice de Desempeño del Calendario (SPI)	$50,000 / $75,000 = 0.67
Estimación a la Conclusión (EAC)	**$180,000**
Estimación hasta la Conclusión (ETC)	$180,000 - $90,000 = $90,000
Variación hasta la Conclusión (VAC)	$100,000 - $180,000 = -$80,000

Agenda

I. Ejemplo Earned Value Management

II. Ejemplo Earned Schedule

III. Ejemplo Earned Scope Management

Ejemplo ES

Calcular el tiempo necesario para terminar el proyecto a partir de la siguiente información obtenida del calendario de un proyecto:

- Duración planeada = 10 meses
- Meses anteriores completados = 7
- Presupuesto autorizado hasta julio = $18,500
- Presupuesto autorizado hasta agosto = $21,350
- Presupuesto gastado hasta agosto = $19,000

Facultad de Ciencias

Ejemplo ES

Métricas	Calendario Ganado (ES)	7 + ($19,000 - $18,500) / ($21,350 - $18,500) = 7 + 0.288 = 7.288
	Tiempo Actual (AT)	8
	Duración Planeada (PD)	10
Indicadores	Variación de Calendario (SV)	7.288 – 8 = -0.712
	Índice de Desempeño del Calendario (SPI)	7.288 / 8 = 0.91 = 91%
	Índice de Desempeño del Calendario hasta la Conclusión (TSPI)	(10 – 7.288) / (10 - 8) = 2.71 / 2 = 1.356
Predictores	Estimado Independiente hasta la Conclusión (IEAC)	10 / 0.91 = **10.98**
	Variación hasta la Conclusión (Variance At Completion - VAC)	10 – 10.98 = -0.98

Facultad de Ciencias

Agenda

I. Ejemplo Earned Value Management

II. Ejemplo Earned Schedule

III. Ejemplo Earned Scope Management

Ejemplo ESM

Datos de Entrada: entradas para los 3 primeros periodos

Elemento	Periodo 1 (mes 1)	Periodo 2 (mes 2)	Periodo 3 (mes 3)	Unidad
Duración Planeada (PD)	10.5	10.5	10.5	(meses)
Tiempo Actual (AT)	1	2	3	(periodo)
Alcance del proyecto (SS)	254	254	254	(CFP)
Alcance Planeado (PS)	24	48	86	(CFP)
Alcance Ganado (ES)	20	40	87	(CFP)
Recursos Humanos usados en el Proyecto (PHR)	11	11	11	(persona)
% completado (%C)	8	16	34	(%)

Ejemplo ESM

Progreso del estado del alcance

Elemento	Fórmula	Periodo 1 (mes 1)	Periodo 2 (mes 2)	Periodo 3 (mes 3)	Unidad
Variación de Alcance (SV)	SV = ES – PS	-4	-8	1	(CFP)
Índice de Desempeño del Alcance (SPI)	SPI = ES / PS	83%	83%	101%	(%)
Productividad por Recurso (PR)	PR = ES / PHR	1.82	1.82	4.27	(CFP/persona)
Productividad hasta la Conclusión (PTC)	PTC = (PSn-1 – PSn) / PHR	2.18	2.18	3.45	(CFP/persona)
Productividad Promedio por Recurso (PROAVG)	PROAVG = AVG(PRn)	1.82	1.82	2.64	(CFP/persona)
Variación de la Productividad (PV)	PV = PR – PTC	-0.36	-0.36	0.82	(CFP/persona)
Estimación hasta la Conclusión (ETC)	ETC(s) = SS - ES	234	214	167	(CFP)

Ejemplo ESM

Predicción del alcance

Elemento	Fórmula	Periodo 1 (mes 1)	Periodo 2 (mes 2)	Periodo 3 (mes 3)	Unidad
Duración Total Estimada (TDE)	TDE = AT + ETC(s) / (PR * PHR)	12.70	12.70	6.55	(meses)
Variación de la Duración Estimada (DESP)	DESP = TDE - AT	11.70	10.70	3.55	(meses)
Diferencia de Duración a la Conclusión (DDC)	DDC = PD - TDE	-2.20	-2.20	3.95	(meses)
Productividad Requerida por Recursos definida para Completar el Proyecto (PRTC)	PRTC = (ETC / PHR) / (PD - AT)	2.24	2.29	2.02	(CFP/persona)
Variación de Recursos para completar el Alcance planeado (RVTC)	RVTC = (-SV) / PR	2.20	4.40	-0.23	(persona)
Recursos Humanos Necesarios para Completar el Proyecto (RNTC)	RNTC = RVTC + PHR	13.20	15.40	10.77	(persona)

Facultad de Ciencias

Contacto

- Jesús Iván Saavedra Martínez
- jism@ciencias.unam.mx